

Jiangxing Wu

Cyber Resilience System Engineering Empowered by Endogenous Security and Safety



Science Press
Beijing



Springer

Cyber Resilience System Engineering Empowered by Endogenous Security and Safety

Jiangxing Wu

Cyber Resilience System Engineering Empowered by Endogenous Security and Safety

 Science Press
Beijing

 Springer

Jiangxing Wu
China National Digital Switching System Engineering &
Technological R&D Center (NDSC)
Zhengzhou, China

ISBN 978-981-97-0115-5 ISBN 978-981-97-0116-2 (eBook)
<https://doi.org/10.1007/978-981-97-0116-2>

Jointly published with Science Press

© Science Press 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publishers, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publishers nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publishers remain neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

If disposing of this product, please recycle the paper.

Preface

The COVID-19 pandemic unseen for a century has had a huge impact on the human society. Fortunately, during this unprecedented and special historical period, the network and information infrastructure has played a decisive role that exceeds the imagination of governments and cannot be overestimated. If people can still accept the epidemic control instruction of isolation and staying at home under the standstill orders, they will never tolerate any failure in information network services. This fully shows that not only under normal circumstances, but also in special historical periods, it is impossible to imagine what social disasters would occur without secure and reliable infrastructure.

Unfortunately, however, the Von Neumann computer architecture, on which the underlying driving (dynamics) paradigm of the current digital ecosystem relies heavily, has its own “genetic defects brought in the womb” (see “Introduction” for details); cyber-physical systems or digital infrastructure, on which the network society is extremely dependent, has inevitable ESS problems or inherent contradictions such as vulnerabilities and backdoors. The network security defense paradigm of “mending the fold after a sheep is lost and learning from mistakes” that has been followed for a long time cannot fundamentally reverse the current situation of serious imbalance between security risks and responsibilities; in addition, the academic and industrial circles have never realized the essence of the problem that “inherent contradictions cannot be eliminated, but can only evolve, transform or reconcile” in cyberspace, which has caused network security threats to become more and more serious, almost to the point of omnipresence and omnipotence. What is even more disturbing is that the digital industry has formed a habitual thinking of “deviation being normal,” and users can only be forced to absurdly accept the products without network security and quality assurance. It seems that it is only natural that all products containing digital elements have “defects that must be patched.” What is even more difficult is that the network security market has strong negative externalities, and users of digital products are forced to bear the high security defense costs in the whole life cycle, which should not be borne by third parties. The difficulty of learning and familiarizing with the complex operation

interface leads to users' inertia of use and limits their freedom of switching to other suppliers. The market monopoly caused by "user stickiness" makes manufacturers stick to the old ways and do not make progress, and the network security field has become "a place where the market mechanism is completely out of order and government intervention must be implemented." Developers or designers of digital products lack the motivation and incentives to take network security seriously; most enterprises "selectively ignore" the network security quality of their products, and even "nearly 50% of manufacturers put their products into the market in spite of knowing they have vulnerabilities" (quoted from the European Union's Cyber Resilience Act). The cyber world needs digital products that are "secure by default out of the box" rather than more add-on or non-integrated network security products.

Due to the unpredictability, extreme uncertainty, and rapid evolution of cybersecurity threats or attacks, people have gradually realized that it is extremely difficult to ensure that digital products have no vulnerability or backdoor, and that the security defense of any part is "not penetrated" at any time in any scenario. In the increasingly complex situation of cybersecurity which is easy to attack but difficult to defend and whose border defense is collapsing, the technical and industrial circles can no longer merely, blindly, and desperately pursue excellent performance while "selectively ignoring" the cybersecurity quality of software and hardware products, and the co-management and co-governance of cyberspace cannot merely rely on the continuous implementation of add-on, plug-in, or built-in security defense measures on the user side or application side to solve the problem of endogenous and uncertain network threats and damages. It is urgent to ensure that the technical goal and quality specification of "mission assurance" with quantifiable design and verifiable measurement can be achieved from the design and manufacturing side of digital products, even if "there are local incidents of network attacks breaking through the bottom line." Undoubtedly, this brand-new guiding ideology and security strategy that the manufacturing side and user side exert their force at the same time to form a "two-wheel drive" effect is the core measure to fundamentally reverse the current strategic disadvantage that cyberspace is easy to attack but difficult to defend. Admittedly, there is no single method that can solve all the security problems of all digital products. However, the only thing that the digital industry cannot do is to continue to selectively disregard this problem. It is in this background that the concept and engineering goal of cyber resilience come into being.

The concept of cyber resilience was originally proposed by the United States in the context of the adjustment of national security strategies and military strategies. With the continuous promotion by the US government, the concept has gradually been recognized by the governments of major countries in the world and the international academic and engineering technology circles. In 2009, Madni et al. proposed in *Towards a Conceptual Framework for Resilience Engineering* that resilience engineering is "building systems that can predict and avoid accidents, survive and recover from disruptions through appropriate learning and adaptation, and restore as much as possible to the state before the disruption." In 2010, Sterbenz et al. defined the concept of cyber resilience in *Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines* as

“the ability of a network to provide and maintain acceptable service levels in the face of various failures and challenges that jeopardize normal operations.” In 2013, the US Presidential Policy Directive No. 21, Cyber Infrastructure Security and Resilience stated that resilience “is the ability to prepare to respond to and adapt to changing conditions, withstand damage and recover quickly therefrom, and resilience also includes the withstanding of intentional attacks, accidental events or naturally occurring threats and the ability to recover from them.” In December 2021, the United States National Institute of Standards and Technology (NIST) officially released *Developing Cyber-resilient Systems: A System Security Engineering Approach*, marking the official release of the world’s first authoritative document on the cyber resilience technology.

NIST defines cyber resilience as the ability of entities possessing network resources to prevent, resist, recover from, and adapt to adverse conditions, stresses, attacks, or damage. In a broad sense, cyber resilience can be applied to a variety of entities such as devices, components, subsystems, systems, systems of systems, public infrastructure, shared services, organizations, industries, regions, and countries. In a narrow sense, cyber resilience comes from the functions and security design of the network equipment itself, as well as various configuration strategies and response mechanisms of the network architecture. It requires the basic equipment, system architecture, operation mechanisms, etc. to have the ability of risk prediction, active resistance, rapid function recovery, dynamic adjustment, and adaptability.

It should be noted that cyber resilience is both similar to and different from traditional cybersecurity. Cybersecurity is defined as “taking necessary measures to prevent attacks, intrusions, interference, destruction, illegal use and accidents on the network, keep the network in a stable and reliable operating state, ensure the integrity, availability and confidentiality of network data, and maximize the protection of network resources from threats in adverse conditions.” Cyber resilience pays more attention to ensuring the basic functions of digital network resources when encountering any potential threats or adverse conditions. It emphasizes that even if some network resources are damaged or captured, the ability to provide basic functions of digital facilities and the ability to recover from and adapt to adverse conditions should be guaranteed. Compared with the traditional concepts of robustness, vulnerability, and persistence, cyber resilience does not stop at the ability to resist stress, but takes into account both the ability to resist stress and the ability to recover. Compared with the classical reliability theory that pays more attention to general high-probability risks, the theory of cyber resilience pays more attention to or focuses more on low-probability extreme threats. In short, “Cyberspace resilience is different from traditional functional resilience in that the latter only focuses on non-confrontational forms of adversity, while the former focuses on confrontation with any potential damage.” It should be pointed out that although toughness and resilience are words similar in meaning in English, there are bigger differences between them in Chinese. Toughness refers to the ability to fight. For example, Mike Tyson, the boxing champion, has a strong ability to fight. The definition of toughness in the sense of materials science may be more accurate, that is, the ability

of materials to absorb energy during plastic deformation and fracture. Resilience refers to the ability to recover after being hit, such as the ability to continue to participate in boxing matches covered by the countdown rule. Resilience defined in the sense of physics refers to the property that an object can recover its original size and shape after deformation. Obviously, the “cyber resilience” we expect should have both of the above capabilities or properties.

At present, cyber resilience engineering provides a method library for dealing with advanced threats. However, the 4 purposes, 8 goals, 26 sub-goals, 49 methods, 5 strategic design principles, 14 structural design principles, and nearly 500 sub-items included in the whole method library are mostly parallel options, and the effectiveness of its engineering method relies heavily on the current cybersecurity defense paradigm based on prior knowledge, or the accumulated experience and cognitive level of designers and project leaders in cybersecurity, lacking theoretical hierarchical relationship analysis and systematic evaluation of gains. So, it is not yet possible to provide guiding suggestions based on system architectures and systematic evaluation of gains. In other words, although cyber resilience engineering as a basic path to “mission assurance” provides a hierarchical method library and rich options at a “managed” level, it lacks a “beads-in-a-string” style core technical architecture at the “manageable” level. In terms of how to effectively combat “potential threats,” especially how to deal with “unknown” internal-external collaborative attacks, it does not provide any “implementable” solutions, nor does it answer the question of how to solve the cyber resilience problem of cyber-physical systems or the functional carrier (or base) of digital facilities using an integrated architecture or in an integrated way.

What needs to be emphasized in particular is that there are three hardcore problems in cyber resilience engineering at present: cyber resilience design based on prior knowledge of cybersecurity theories and methods—a major hard defect of inability to cope with the challenge of unknown or uncertain threats; cyber resilience cannot only be “concrete” without “steel skeleton”—a major hard problem of lacking the support of an integrated architecture; the hard challenge of the lack of quantitative design and evaluation methods for cyber resilience. The current development situation seems that the National Institute of Standards and Technology (NIST) has opened a traditional Chinese medicine shop with rich medicinal materials for cyber resilience engineering and has classified the pharmacological properties of each medicinal material, and even provided the processing method as well as matters needing attention. However, it is still necessary to ask a richly experienced traditional Chinese medicine doctor to give diagnoses and prescribe medicines. In other words, it is far from enough to solely rely on the formula, material, and mixing technologies of concrete to make digital products from the manufacturing side have the cybersecurity quality of “reinforced concrete.” How to design the skeleton configuration and choose suitable materials according to the application scenarios is the first engineering problem to be solved in manufacturing “safer and more secure digital products.” This book focuses on the three major problems existing in cyber resilience engineering and describes in detail the cyber resilience engineering

scheme and application examples based on the ESS theory and methods founded by the author.

Since 2008, the author's research team has been committed to applying the axiom that "structure determines function, structure determines performance, structure determines efficiency, and structure determines security" in innovative theoretical researches and technological invention in the fields of the computer architecture and cybersecurity. In 2008, the team was the first in the world to propose "the high-performance architecture of domain-specific dynamic variable structure software and hardware collaborative computing based on active cognition." Different from the static or rigid expression of the classic von Neumann structure, this computing architecture creates a new path for variable structure computing where "structure adapts to application." It follows the idea of dynamic reconstruction of the multi-dimensional environment of active cognition, perceives the time resource utilization, load distribution, energy consumption constraints, and other environmental parameters of computation tasks in real time, and schedules software and hardware functional modules (or computational particles) with appropriate architecture to collaboratively complete domain-oriented high-performance computing tasks. This enables the system to proactively select domain-specific modules or computational particles with appropriate energy efficiency ratios for the same task under different time periods, different loads, different resources, and different operating scenarios to obtain ideal full-task cycle processing efficiency. However, for quite some time, the author was not satisfied with the Socratic naming of this dynamic variable structure computation until 1 day in 2010 when I inadvertently watched a program on the National Geographic Channel (NGC). The program introduced the nature's most outstanding camouflage creature—the striped octopus. The author deeply felt that in terms of working mechanisms there is a striking similarity between the mimic phenomenon and the calculation model of "structure adapts to application," both of which are caused by endogenous functions, so the domain-specific software-hardware collaborative variable structure computing architecture based on active perception is nicknamed the mimic structure calculation (MSC). Ten years later, at the ISCA 2018 Annual Meeting, a joint paper by Turing Award winners David A. Petterson and John L. Hennessy pointed out that domain-specific software and hardware collaborative computing architecture (Domain-Specific Architecture; DSA/DSL) would be one of the four important directions in the golden development period of the computer architecture in the next decade.

As we know, the phenomenon of biological mimicry is endogenous in nature and does not require any built-in (internally installed) elements, embedded (in-line) elements, internalization or endosymbiosis in its mechanism to provide assistance or pseudo-decorations. Broadly speaking, the ESS presented in the "man-machine-environment-management" system is strongly correlated to the completeness and real-time nature of prior knowledge, so it does not fall into the category of endogeneity.

In 2013, the world's first domain-specific software and hardware collaborative computing principle verification prototype based on MCA passed the national acceptance test in Shanghai. This energy-efficient software and hardware variable

structure collaborative computing can be easily converted into high-performance variable structure collaborative processing. Therefore, MCA has the functions of free conversion, collaborative optimization, and dynamic management between efficiency and performance targets. The traditional performance-oriented computing and processing architecture is often rigid, and its processing environment is usually static, deterministic, and similar. In addition, there is a lack of security analysis and related design indicator systems, resulting in some wrong security assumptions when building the system. It cannot prove its own security or avoid various ESS problems, nor can it be used as “Secure Enclave,” a trusted execution environment (TEE). The mimic computing environment has the characteristics of active cognitive variety, dynamics, randomness, and emergence of software and hardware co-processing. Under the condition of functional equivalence, the non-deterministic or diverse relationship between its task function and algorithm implementation structure can just make up for the endogenous shortcomings of staticity, certainty, and similarity of traditional information processing systems in dealing with attacks based on ESS problems. A natural inference is that, in view of the deliberate internal-external collaborative actions of attackers who take advantage of the vulnerabilities and backdoors unknown to defenders in digital facilities, using the functionally equivalent dynamic variable structure based on threat perception to deal with the uncertainty mechanism of the environment should be able to effectively disrupt or disintegrate the stability or reliability of the attack chain based on ESS problems. In order to transform the underlying driving paradigm of the digital ecosystem, we must first revolutionize the theory and methodology of cybersecurity. The basic understanding is that “only ESS-empowered cyber resilience engineering practice” can break the vicious circle of “patch after patch.”

To this end, in 2013, the author formally proposed and created the CMD theory based on ESS mechanisms. At that time, however, it was only inspired by the phenomenon of endogenous mimicry and camouflage in the biological world. In practice, the author explored and invented a dynamic heterogeneous redundancy (DHR) architecture that does not rely on (but can be integrated with) prior knowledge and can effectively deal with “known unknown” or “unknown unknown” potential threats or attacks. The architecture has “mimicry disguises fog (MDF),” “uncertainty/entropy non-reducing effects,” “dimension-reduction solution mechanisms of complex security problems,” “double-blind effects generated by structural coding or environmental encryption,” “trusted execution environment,” “structure-based exponential security gain,” and other characteristics. Tests/experiments have confirmed the generalized robust control properties of the architecture’s endogeneity (fully covering the vision and goal of cyber resilience engineering currently proposed by the United States and Europe). Unfortunately, at that time, it was not yet possible to theoretically reveal the contradictory nature of CESS problems, clearly provide proof of the existence of ESS in cyberspace and related prerequisites, and thoroughly analyze the structural coding or the mathematical nature of structural encryption of DHR, allowing “vest-wearing” stakeholders in the market to play tricks such as stealing concepts and selling dog meat while hanging sheep’s heads and blur the subversive or revolutionary significance of the ESS theory.

It is not difficult for readers to find that mimic computing and ESS mimic defense mechanisms are essentially based on collaborative computing or processing of software and hardware with variable structures under functional equivalence conditions, and both achieve the perception and cognition of the operating environment through construction technologies. Therefore, considering the mimic defense as a transformation of mimic computing in the application dimension is an inheritance and development in both theoretical and practical senses.

In 2016, the CMD principle verification system passed the authoritative test and evaluation organized by the Ministry of Science and Technology of China in Shanghai. Its unique generalized robust control architecture based on ESS mechanisms is outstanding in five aspects. Firstly, it can transform stealth attacks targeting unknown vulnerabilities or backdoors of individual executors within the mimic structure into unknown differential-mode or perceivable common-mode security events within the mimicry interface. Secondly, it can uniformly express differential-mode attacks or uncertain disturbances as generalized uncertain threats with probability attributes. Thirdly, the “uncertain effect” produced by the strategy scheduling based on mimic ruling and the multi-dimensional dynamic reconstruction feedback control mechanism can break down the prerequisites for trial-and-error or blind APT attacks. Fourthly, with the help of the logical expression of the “relatively-true” axiom, it can acquire threat perception ability based on the relative consensus mechanism without relying on the attacker’s prior knowledge or behavioral characteristic information. Fifthly, it can normalize traditional or non-traditional security threats into generalized robust control problems and achieve integrated processing.

Since 2013, the author’s research team has published a series of leading research papers and technical reports on mimic defense and ESS at home and abroad, and since 2017 has published four monographs, i.e., *Introduction to Cyberspace Mimic Defense*, *Principles of Cyberspace Mimic Defense: Cyberspace Endogenous Security and Generalized Robustness Control*, *Cyberspace Mimic Defense: Generalized Robust Control and Endogenous Security and Safety*, and *Cyberspace Endogenous Security and Safety: Mimic Defense and Generalized Robust Control*. The author and his team have always been committed to philosophically revealing the causes for endogenous contradictions in cyberspace security and safety problems. They use general analysis models such as the “Impossible Triangle” or Venn diagrams and set computing, and taking dynamics/randomness (D), variety/heterogeneity (V), and redundancy (R) as the three key elements or core function set of unknown threat defenses, and creatively build a universal deconstruction model based on the “Impossible Triangle of Unknown Threat Defense.” Through qualitative analysis, the author systematically sorted out the existing cybersecurity defense technologies in terms of transforming or resolving unknown ESS contradictions and concluded that the incomplete intersection of DVR based on the triangle structure relationship definitely does not have the ability to prevent unknown threats. The author further proposed the conjecture that ESS may exist in the completely intersecting function set of DVR, and theoretically derived the “endogenous security and safety existence theorem” in cyberspace and the “perfect security” analysis conclusion based on

cryptography. For the first time, it was qualitatively and quantitatively proved that there is indeed an ESS mechanism in cyberspace that can effectively combat “known unknown” or “unknown unknown” threats or destruction without relying on prior knowledge and additional security technologies. Through modeling and analysis of the DHR architecture invented in the early years, it is proved that the feedback control and dynamic reconfigurable operating scenario architecture based on strategic ruling have the completely intersecting set properties of DVR and reveal the interaction and complementary relationship among the three, as well as the six unique characteristics of the DHR structure with the mathematical and physical properties of structural coding or environmental encryption. It can comprehensively solve the multiple intertwined problems of functional safety and cybersecurity (even information or data security) on the manufacturing side of cyber-physical systems or digital products, effectively provide a generalized robust control function integrating high reliability, high credibility, and high availability and solve the worldwide problem of “concrete structures lacking steel bar reinforcement” in cyber resilience engineering so far. Its extraordinary technical significance is like the invention of heterogeneous redundancy and isomorphic redundancy, a milestone in breaking through the “ceiling” of reliability technologies. The author team’s groundbreaking and fruitful theoretical and technical researches show that not only does the biological world have structure-based endogenous defense functions such as mimic disguise, but cyberspace also has ESS defense functions based on structural coding or environmental encryption that do not rely on (but can be integrated with) prior knowledge and additional security technologies.

After more than 10 years of theoretical iteration and practice, the author’s team has completed a gorgeous transformation from obtaining inspiration for the mimic defense from natural phenomena to analyzing and deriving the cyberspace ESS theory at a rational cognitive level. Potential security threats and damages such as random failures or uncertain errors, man-made or non-man-made, “known unknowns” or “unknown unknowns” existing in architecture-enabled cyber-physical systems or digital products can be transformed into generalized reliability problems of differential-mode or perceivable common-mode properties that can be expressed probabilistically in the DVR domain through creative, universally empowering DHR architecture or algorithms, making it possible to use the mature resilient function design and automatic control theory and methods to deal with the problems. At this point, we can give a scientific answer to the question of “why the ESS design of digital products can effectively deal with the unknown security threats or destruction within the DHR architecture without violating the common sense of epistemology” from both the engineering level and the theoretical level.

This leads to an academically interesting conjecture: “Since the effectiveness of ESS does not rely on any prior knowledge and add-on security measures, and the security of most encryption algorithms also does not depend on any prior knowledge and “bodyguard-like” or “watchdog-like” defense technologies, do they have some similar or identical mathematical properties? Obviously, if this conjecture can be proved to be true, then DHR structural coding is a newly created encryption mode in addition to “information encryption,” which the author names environment

encryption (EE). Therefore, for the defenders, the ESS mechanism is to “encrypt the environment (or code the structure)” for the “dark functions” such as potential design defects of software and hardware resources or virus Trojans that may be hidden, so that the attacker’s asymmetric advantage of penetration or destruction by means of “one-way transparent information” such as unknown vulnerabilities and backdoors in the structural environment is offset or disabled, forcing attackers to fall into the trap of “having to crack the environment password or structure codes before being able to exploit the interior of the target object.” In other words, if DHR structural coding or environmental encryption mechanism is established, the attacker’s efforts to implement any “internal-external collaborative” or “one-way transparent” network attacks will be futile in mechanism, while the defender can not only transform “unknown unknown” disturbances into “known unknown” differential-mode or perceivable common-mode perturbations and achieve the cyber resilience goal of “mission assurance,” but also create “trusted execution environment (TEE)” without relying on the trustworthiness of trusted roots. Under the condition that it is difficult to be bypassed or short-circuited, structural coding or environmental encryption makes the probability of ESS problems being successfully turned into ESS events small enough and provides a TEE for “information encryption algorithm” based on “structural coding or environmental encryption.”

Based on this conjecture, the author’s research team analyzed the encryption properties of the endogenously secure DHR structural coding based on cryptographic principles for the first time, and gave preliminary yet general conclusions, pointing out that all cyber defense or trusted services that do not rely on prior knowledge or add-on security measures have similar or identical encryption properties (interested readers may conduct a more in-depth theoretical analysis on this proposition). What is exciting is that the creation of the concept of structural coding or environmental encryption allows the cyberspace ESS theory to give a more popular and easy-to-understand scientific expression. What is more gratifying is that at the beginning of 2024, another research team of the author made a groundbreaking analysis of the properties of the DHR architecture from the perspective of non-cooperative game equilibrium, and initially proved that the ESS strategy based on this architecture is easy to achieve the Nash equilibrium, i.e., it is impossible for an attacker to improve his own gains no matter how his attack strategy is unilaterally changed (the paper is to be published).

As an original theory and disruptive technology, ESS-empowered structures have universal applicability for digital products. Any cyber-physical system or digital product based on the DHR architecture will naturally have integrated high reliability, high credibility, and high availability. Compared with “the ability to sustain basic functions in the case of damaged network resources and resume cyber resilience by recovering from and adapting to adverse conditions,” the ESS empowerment structure can achieve the goal of “mission assurance” before, during, and after the event and can provide TEE-based trusted service functions, while the former can only meet the “minimum requirement” for mission assurance on the premise of “selectively excluding or ignoring” unknown threats or damage.

There is no doubt that the endogenously secure DHR architecture is just the “magic doctor” in the traditional Chinese medicine shop or the skilled craftsman to “string the beads into a necklace” whom the current cyber resilience engineering is urgently looking for. The author believes that ESS-empowered cyber resilience engineering and trusted (computing) services will open up a new path for the development of functional safety technologies and the digital industry, greatly promote the upgrading of information technologies and related industries and digital products, and an era in which the design and manufacturing side of digital products incorporates “co-governance of cyberspace” into the primary goal and responsibility. Although the ESS contradictions such as “genetic defects existing in the architecture of stored program control” and code vulnerabilities and backdoors “cannot be completely eliminated,” we can still achieve the unity of opposites of “seeking solutions that coexist with problems in the problem space.” As long as digital products have endogenously safe “non-specific immune function,” even if there are still “antigens” similar to “Covid-19” in the products or systems, the “severe disease rate and mortality rate” can still be controlled within an acceptable range. This idea of autoimmunity and herd immunity coexisting with viruses is highly consistent with the goal that cyber resilience engineering tries to pay attention to antagonism against any potential damage. At the same time, we can also give a technical explanation to the philosophical problem of “self-proof of innocence”: “If given constraints, the generalized uncertain (including man-made or non-man-made) disturbances can be transformed into differential-mode or perceivable common-mode probability events that the existing reliability and automatic control methods and technologies can handle.” This cybersecurity mechanism expressed by “controllable probability” rather than “best effort” does not need “self-proof of innocence.”

Nowadays, the problem of network security has risen to a strategic height of national and political security, and the development mode of economic globalization is facing unprecedented challenges. If we want to continue to promote the free trade of IT-related products but do not vigorously promote the transformation of the underlying driving paradigm of the digital ecosystem, and do not break through the cyber resilience engineering problem of quantifiable design and verifiable measurement of the ESS quality of digital products, it will be difficult for us to break through the various trade barriers and national security obstacles caused by the generalized cybersecurity threats. The ESS theory and mimic structure that empowered digital product design security and default security are indispensable theoretical and technical tools to break this dilemma.

The author is convinced that mankind will usher in a new era of transformation of the underlying driving paradigm of the digital ecosystem with the empowerment of ESS. The serious imbalance between cybersecurity threats and responsibilities on the user side and the manufacturing side is expected to be fundamentally reversed; the contradictions between “security and openness” and between “advancement and trustworthiness” can be unified unprecedentedly on the basis of the ESS theory and methods; the current mainstream attack theories and methods based on the design defects of digital products’ software and hardware codes will be subverted;

information technologies and related industries will be bursting with fissile innovation vitality and usher in a strong market demand for upgrading. A new generation of digital products, such as cyber-physical systems, industrial control devices, digital infrastructure, Internet of Vehicles, IT-OT devices, and confidential computing platforms, with generalized robust control structure and ESS functions will surely reshape the order of cyberspace security.

What is exciting is that since its inception, ESS theories and methods are gaining increasing consensus in the academic and industrial circles, and are accelerating their dissemination at home and abroad. Various groundbreaking or instructive theories and technical practices are achieving or have achieved unprecedented successes in more and more fields and industries. In particular, they have provided a basic solution to the three hardcore problems that cyber resilience engineering cannot cope with “unknown unknown” man-made destruction, that “there are only concrete materials but no steel skeleton,” thus resulting in the lack of structural gains, and that network security cannot be quantifiably designed and verifiably measured.

At present, many research institutions around the world are focusing on a very valuable proposition: how to create “resilience or toughness to resist systematic network security threats.” They expect to construct a new underlying driving paradigm of digital ecosystem, which can effectively resolve the “cyber resilience deficit” accumulated while digitalization, intelligence, and the Internet of Everything bring technological progress to human society, as well as the systematic network risks and even social disasters induced by it. It is certain that the ESS theory and technology will provide a brand-new thinking perspective, methodology, and practical norms for the underlying driver paradigm of the digital ecosystem. Just as Global Seismographic Network (GSN) and Global Positioning System (GPS) provide undeniable evidence of spatial-temporal uniqueness for effectively verifying global underground nuclear tests, cyber resilience engineering empowered by the ESS theory and method will surely make a milestone contribution to the healthy and sustainable development of the global digital transformation and the construction of an inclusive, resilient, safe, and trustworthy cyberspace.

Zhengzhou, China
February 2024

Wu Jiangxing

Introduction

The 2022 Nobel Prize in Physiology or Medicine was awarded to the Swedish scientist Svante Pbo for his discoveries about the extinction of the ancient human genome and human evolution. In 1997, Pbo and his colleagues pioneered paleogenomics after they successfully sequenced the mitochondrial DNA of Neanderthals that died out 30,000 years ago.

Existing research evidence suggests that although Neanderthals disappeared from the earth tens of thousands of years ago, their bloodline has reproduced through different human races to the present day. In 2021, researchers from Montreal discovered that Neanderthal genes affect the immune system of modern humans. Among them, a gene called HLAclassl has a strong antiviral ability that enhances the immunity of modern humans.

From this, the author thought of Turing-Church-Gödel's computability theory and the engineering model of Turing machine—the Von Neumann architecture of the modern digital computer. As a gene with extremely strong survival ability, it has profoundly promoted the progress of the human society for decades in digitalization, intelligence, networking, and informationization. Unfortunately, while it creates unlimited development prospects to the human society, it also brings unprecedented security challenges to cyberspace. The reason is that there is no anti-virus gene in Turing-von Neumann's "mitochondrial DNA." In other words, there is no innate anti-virus immunity in cyberspace, a network world empowered by Von Neumann's computer architecture. Borrowing the terminology of vertebrate immunology researches in biology, cyber-physical systems, or digital products empowered by modern computers' framework of "stored program control" have no innate non-specific immunity in any sense except acquired specific immunity. Even if cyber resilience engineering is in full swing at present, it is still trapped in the cybersecurity dilemma caused by the structural genetic defects of computers when confronting potential cyber threats or destruction based on unknown vulnerabilities, backdoors, and virus Trojans.

According to the public literature, the concept of the computer virus was first proposed by an American computer expert named Fred Cohen at an international computer security academic lecture in 1983. He defined the computer virus as "a

computer program that can affect other computer programs by modifying those programs, including possibly copying itself.” Earlier in 1949, John von Neumann mentioned the concept of “self-replicating computer programs” in a paper titled *Theory of Self-reproducing Automata*. As for the vulnerabilities (backdoors) of software and hardware codes that are regarded as a scourge by computer scientists and cybersecurity experts today, they were mentioned as early as 1947 when von Neumann established the theory of the computer system architecture. He believes that the development of computers is similar to that of biological life, and a computer system also has inherent genetic defects. “Even simple (stored program controlled) computing systems can show complex behaviors, and the possibility of errors is inherent in all computing systems, and unexpected problems may arise in the process of use and development,” he said. The author believes that Turing-Church-Gödel computability theory, which is respected as the bible of the computer science and technology, only provides a model or sufficient conditions for what is a computable problem, but it is impossible to distinguish between “good-faith” computing and “malicious” computing. As a result, the human weakness of “being both good and evil” has been magnified “without a bottom line” in cyberspace, causing viruses, Trojans, backdoors, trapdoors, and other malicious codes to proliferate as a lingering “evil ghost” in the Internet age.

With such a lengthy preface and introduction, the author wants to point out that today’s cyberspace security problems have become an opened Pandora’s box because, first, these problems are related to the inherent genetic defects such as the inevitable code design vulnerabilities or the dark functions that are difficult to thoroughly investigate in the digital system empowered by the stored program control architecture; second, yet more importantly, the network security market has a strong negative externality (the users of digital products bear the cost of security defense, and the third party bears the consequences of network attack or destruction), and manufacturers lack the necessary incentives to invest in the safety development of products in the whole life cycle, always put commercial interests above their due social responsibilities, and recklessly and perversely pass on the risks and responsibilities of network security to the users. What needs to be emphasized is that the transformation of the underlying driving paradigm of the digital ecosystem, which has been set off around the world recently, and the new trend of cyber resilience engineering can no longer continue to “selectively ignore” the harsh fact that almost all information carriers or digital products today have CESS problems without exception, and can no longer blindly pursue extreme functional performance and profitability while neglecting in a utilitarian way the prevention of systematic cybersecurity risks. The practice of engineering technology over the past decades shows that it is impossible to achieve the desired vision of cyber resilience engineering by continuing to follow the security guarantee mode based on “mending the fold after a sheep is lost, guarding the home and protecting the yard, and learning from a mistake.” The digital industry has finally realized that network security should be regarded as the core function of all products containing digital elements, which runs through the whole life cycle of products from design,

manufacturing to use, not just as a technical option of products. Therefore, based on the feedback control mechanism of cybernetics, the necessary diversity theorem (Nash equilibrium theorem), and the relatively true axiom and following the system engineering method of “structure determining security,” we need to develop subversive technologies that empower ESS and “integrate network security into the design and manufacture of technologies and products. Instead of letting two engineering teams design two systems, of which one protects the other, a single system should be designed to integrate security capability into it, so that the “manageability” goal of cyber resilience at the engineering specification level can at least be expressed at the software and hardware carrier (or base) level. We must adopt an innovative cybersecurity paradigm to support the transformation of the underlying driving paradigm of digital ecosystem, realize the “left shift” of network security responsibility from the user side to the manufacturing side, and establish a tough and resilient digital ecosystem that is easier to form inherent defense capabilities. We must adopt transformative theories and subversive technologies to form an inter-generational advantage in defense mechanism against network attackers, obtain a “Nash equilibrium solution to non-cooperative game,” establish a situation conducive to the transformation of defense advantage, and make it difficult for attackers to improve their own gains even if they pay higher and more lasting costs than defenders, so that we can always take the initiative in defeating various threats faced by the digital ecosystem.

However, it is a pity that whether it is the security-by-design that “moves from the user side to the manufacturing side” or the out-of-the-box security by default, the beautiful vision still lacks the support of revolutionary cybersecurity theories and methods. The current cyber resilience engineering continues to rely on or follow the perspective of cybersecurity thinking, methodology, and practice norms based on prior knowledge, so it is doomed to be unable to effectively deal with unknown or uncertain attacks or sabotage, and it is impossible to fundamentally jump out of the vicious circle of “patch after patch,” and it is even more impossible to give the cybersecurity indicators of digital products that can be quantitatively designed and verifiably measured.

The theoretical research and technical practice of the author’s team in the field of cybersecurity for 10 years show that although the only definite nature of cyberspace is uncertainty, it does not rule out the possibility of transforming uncertain problems into controllable probability events under certain preconditions. The exploration and practice of cyber resilience engineering empowered by the ESS architecture has proved that even in cyberspace with genetic defects, a reliable solution can be found to coexist with ESS problems. Uncertain security disturbances caused by man-made or non-man-made factors can be transformed into differential-mode or perceivable common-mode events which can be handled by the theory and technology of generalized functional safety. Undoubtedly, ESS can provide the much-needed “skeleton of reinforced concrete” for cyber resilience engineering.

In order to systematically elucidate the function and significance of ESS-empowered cyber resilience engineering and some typical application

examples from the aspects of basic theories, methods, and practice norms, the related chapters provide the following contents.

Chapter 1 of this book divides cyberspace security and safety problems into two types, ESS and non-ESS problems, which are completely different in nature. It emphasizes that ESS problems are caused by the inherent contradictions of things and the structural mechanism, while the generality of contradictions determines that ESS problems cannot be completely eliminated but can only evolve, transform, reconcile, and achieve a unity of opposites. At the same time, it is pointed out that ESS problems can be further divided into common security and individual security problems. The ubiquitous and antagonistic characteristics of the former require seeking universal solutions at the mechanism or structural level, while the local and long-term nature of the latter requires exploring special solutions that are technically and economically feasible. It is also emphasized that the intertwining and overlapping of cybersecurity, functional safety, and even information/data security has caused the development of cyber-physical systems or digital product technologies to face unprecedented broad functional safety challenges. It introduces the general concepts and basic characteristics of ESS problems, the basic concepts, properties, threats, damage, and examples of vulnerabilities and backdoors, and then gives the definition of common ESS problems and GFS problems in cyberspace as well as their causes, basic properties, and measures against them. Then it is pointed out that common ESS problems are objectively existent and exploitable resources which advanced persistent attacks (APTs) mainly rely on and take advantage of. The traditional cybersecurity and safety defense measure of “repairing the system and making patches” cannot totally eliminate the “unknown unknown” attacks and sabotage. Therefore, for a long period of time in the future, special attention should be paid to the evolution and transformation of common ESS problems or contradictions and the study of the laws and methods of unity of opposites, especially the innovations in blocking or controlling the interaction between internal and external factors. It is hoped that a set of methods can be developed that will effectively solve or avoid the common CESS problems without relying on (but integrating with) prior knowledge or expecting to totally eliminate vulnerabilities and backdoors.

In Chap. 2, the author reconstructs the theoretical framework and practical norms of *Cyberspace Mimic Defense —Generalized Robust Control and Endogenous Security and Safety*, which was founded 10 years ago. Firstly, the concept and method of development paradigm are introduced to divide the development history of the existing cybersecurity technologies into three milestone stages, and it is pointed out that the existing cybersecurity defense paradigms cannot bring guidance at the level of thinking perspective, methodology, and practice norms to the “unknown unknown” cyber threats or sabotage based on ESS problems of cyber-physical systems or digital products. Taking dynamics/randomness (D), variety/heterogeneity (V), and redundancy (R) as the three vertices of the universal deconstruction analysis model’s impossible triangle, the author creatively puts forward the “Incomplete Intersection Principle (IIP)” for the defense of unknown security threats and draws the conclusion that as long as D, V, and R are in a triangular relationship in construction or combination, it will be impossible to deal with unknown security

threats or “internal-external collaborative” attacks. Secondly, the author analyzes the existing cybersecurity defense technology by using the IIP theorem and gives the fundamental reason why additional or plug-in, built-in or embedded security technologies cannot defend against the “unknown unknown” cyberattacks or sabotage. Then, the author puts forward the vision of ESS defense and expounds upon the expected ESS defense mechanisms, structure, and characteristics. On this basis, a theoretical framework of ESS architecture design is established from the aspects of thinking perspective transformation on the design/manufacturing side, methodology innovation, updating practice norms, and refusing to stop trial-and-error attacks; the differences between it and the existing network security defense paradigm are pointed out; the application scope and constraints of the theory are given.

Chapter 3 firstly puts forward the conjecture that there may be ESS in the “DVR fully intersecting” domain and proves that if a structure or algorithm has the completely intersecting expression of three elements: dynamics/randomness (D), variety/heterogeneity (V), and redundancy (R), even in the absence of prior knowledge and additional security measures, any differential-mode or perceivable common-mode attacks based on unknown vulnerabilities, backdoors, and virus Trojans can be curbed by the endogenous effect of the structure itself, and random or uncertain differential-mode or perceivable common-mode disturbances can also be effectively dealt with. Secondly, this chapter discusses the engineering realization of the DVR complete intersection structure from the classical DRS architecture. The analysis shows that DRS construction can transform ESS problems such as random failure and uncertain errors, known/unknown security perturbation, and artificial/non-artificial disturbance into probabilistic events expressed in the form of differential modes or common modes in the VR domain (although the cause for events is still unknown). However, due to the inherent static and certain dissimilar redundancy construction, it cannot cope with unknown cyberattacks, APTs, or sabotage based on internal-external collaboration, nor does it have the expected stability robustness and quality robustness. Nevertheless, it is found that if a feedback control mechanism is introduced on the basis of DRS construction, it is possible to meet the property of ESS of DVR complete intersection. The author’s further research shows that according to Shannon’s channel coding and error correction theory, the ESS mechanism expected by DVR complete intersection can also be described as a scientific problem of “how to correctly process and transmit information on a reconfigurable memory channel with non-random noise.” Then, the author constructs an ESS transmission and processing construction coding model which is equivalent to DVR complete intersection, and then deduces the ESS-Existence Theorem (ESS-ET), which is centered on the first and second theorems of ESS, the randomness lemma of the memory channel, and the randomness lemma of the non-random noise memory channel and proves that the DVR completely intersection coding structure has expected ESS properties, which is theoretically invulnerable.

Ten years ago, the author’s team invented an enabling technology structure—the DHR architecture, which meets the principle of endogenous security and safety existence (ESS-ET), that is, introducing innovative mechanisms such as dynamic feedback control based on strategic ruling and encryption of the running

environment structure into the DRS structure, making the DHR structure coding have mimicry camouflage fog (MDF), entropy non-reduction and uncertainty, dimensionality reduction of GFS problems, emergent security gain (ESG), structural coding (SC) or environmental encryption (EE), and TEE.

Section 3.5 introduces the prerequisites and constraints for the establishment of the DHR architecture and expounds upon the DHR structural encoding, working principles, attack surface (AS), band-pass filter, and double-blind effect.

Section 3.6 gives, for the first time, the research results of ESS analysis based on cryptography. Inspired by Shannon's concept of "perfect secrecy-keeping," the author puts forward the conjecture that there are some similarities between "perfect safety and security" without prior knowledge and "perfect secrecy-keeping" without prior knowledge, creates a cryptographic model of intrinsic function safety and security, gives the definition of perfect intrinsic function safety and security (PIFS), points out that there is a strong correspondence between the PIFS system and the perfect information security system, as well as the key elements to realize it. It is proved that DHR is the relevant analysis and inference of the PIFS system under given conditions, and that the conjecture that "perfect safety and security independent of prior knowledge is similar to the PIFS system" is valid. Different from the information encryption mechanism, ESS pioneered a "structural or environmental" encryption mechanism that is transparent to the intrinsic function, and the theoretical correctness of the inference that "structure determines safety and security" derived from "structure determines function" is proved for the first time.

After Section 3.7, the main engineering and technical effects of the DHR architecture, restricted application problems, backdoor (trapdoor) problems of application software, and their subversive influence on the current rules of cyberspace games are given.

Section 3.11 gives the principle of "the white box plug" test of the ESS function, which is essentially a way of manually inserting "hooks" or "loading test modules" to "implant" unknown and non-destructive test case codes such as "vulnerabilities, backdoors/virus Trojans" into the corresponding executors of the current running environment, so that testers can inject or modify the functional codes needed for "white-box test cases" through the attack surface channel of the target system. Then, through the input channel of the tested system, the activation sequence of "vulnerabilities, backdoors/viruses and Trojans" that meets the compliance check are sent to observe whether the expected attack response of the test case appears at the output end of the system.

At the end of Chap. 3, the author points out the supporting function of the diversified ecology of cyberspace for the ESS empowerment technology, and the role of the ESS in promoting the prosperity of cyber ecology and the progress of cybersecurity technologies.

Chapter 4 introduces the development history, basic concepts, basic connotations, the functional safety integrity technology, and the random failure and system failure analysis technology and analyzes the technical challenges faced by functional safety in terms of quality crisis brought by software expansion, generalized functional safety problems, and uncertainty brought by AI technologies. This chapter briefly

introduces the basic concept, engineering framework, analysis and evaluation of cyber resilience engineering which evolved from functional safety resilience and summarizes in a nutshell the challenges and shortcomings of cyber resilience from the perspectives of theoretical thinking, system architecture, and evaluation measurement analysis. The author emphatically points out that there are major hard defects in the thinking perspective of avoiding unknown attacks, major hard problems in the engineering system that lacks architectural guidance of architecture, and major hard challenges in the evaluation method that lacks the measurement of core competence, as well as other prioritized problems that need to be solved urgently. The author also emphasizes that it is necessary to make a fundamental breakthrough and innovation from the perspective of cybersecurity paradigm, so that cyber-physical systems or digital products can effectively deal with unknown network attacks or sabotage, empower the indispensable GFS quality, and open up a new way of cyber resilience engineering.

Chapter 5 firstly analyzes the indispensable role of ESS in empowering cyber resilience engineering from four aspects: thinking perspectives, theoretical basis, architectural guidance, and evaluation innovation. Secondly, based on the technology tree model, the root technology, branch technology, and leaf technology of ESS-empowered cyber resilience engineering are given layer by layer using a hierarchical method. Finally, the design principles for ESS-empowered cyber resilience engineering are proposed, and it is pointed out that the DHR architecture, as the overarching technical architecture of existing cyber resilience engineering methods and implementation principles, can play the role of a “concrete reinforcement skeleton.” It is proved that ESS-empowered cyber resilience engineering can not only effectively deal with “known or unknown” random perturbations or deliberate disturbances, but also effectively block the impact of “unknown unknown” faults or network attacks, thus ensuring the realization of the vision of cyber resilience engineering “to cope with any potential threat or sabotage” at the level of trusted service or TEE.

Chapter 6 provides the evaluation indicators and methods of system architecture cyber resilience from a new perspective of evaluating the cyber resilience of system architecture. The system architecture cyber resilience evaluation framework focuses on the common problems of cyber resilience and the assessment of related core capabilities. It provides core indicators for quantifiable design, verification, and measurement of system cyber resilience capabilities, which can play a “critical-vote” role in engineering practice. The system architecture cyber resilience assessment includes static assessment, adversarial assessment, and destructive assessment. Among them, ESS white box testing, as a non-subjective assessment method, not only can lessen the current problem of excessive reliance on qualitative methods in cyber resilience assessment, but can also quantitatively evaluate cyber resiliency gains brought by ESS construction techniques.

Chapter 7 briefly introduces the concept of ESS architecture empowerment, selects typical IT, ICT, ICS, or CPS systems such as the network communication control system, trusted cloud computing service system, Internet of Vehicles system and industrial control system, and the application examples of ESS-enabled cyber

resilience engineering in many fields are introduced from the aspects of threat analysis, ESS overall architecture, isomer construction, strategic ruling design and solutions, showing the superiority and vitality of ESS-enabled cyber resilience engineering from the practical level. At the same time, it is also pointed out that although the ESS empowerment architecture has universal application significance, the empowerment design of different digital products still requires customized optimization or even creation based on the application characteristics of each field or industry.

Chapter 8 explores the application prospects of the ESS theory and methods as “stones from other mountains” in emerging fields such as wireless communications, artificial intelligence, and chip design. First, on the basis of reviewing the development paradigm of wireless communication and sorting out the ESS problems of wireless communication, the development vision of ESS of wireless communication is proposed, including the ESS architecture, mechanisms, key technologies, and performance analysis methods of wireless communication. Second, based on the analysis of the common characteristics, individuality, and GFS problems of artificial intelligence ESS, a preliminary design plan for the ESS defense framework of artificial intelligence is given. Finally, based on a brief introduction to the design ideas of ESS chips, the structural advantages of MCU chips with ESS architecture in resisting heavy effect upset are quantitatively analyzed. Relevant researches show that the ESS architecture not only has an ESS empowerment effect, but also has significant scenario or domain diffusion effects. It has strong permeability and extensive reference significance.

What needs to be emphasized is that, in a universal philosophical sense, the only definite nature of the physical world is uncertainty. In the sense that human beings transform nature, the core function of technological invention is to transform uncertain problems into controllable probability problems with differential-mode properties under constraints. From the perspective of network security defense, although there is no “absolute security” in cyberspace, it is still necessary to find a way to transform uncertain disturbances caused by human or non-human factors into events with controllable probability or to prove “self-innocence.” In the sense of engineering technology, ESS theories and methods solve the world-class problem of trusted services lacking a TEE, and confidential computing/trusted computing cannot answer the fundamental question of “whether the root of trust itself is trustworthy.” With the help of the ESS architecture, even if there exist multi-intertwined known or unknown differential-mode or perceivable common-mode threats or damages to functional safety, cybersecurity, and even information/data security in the TEE platform, they will not mechanically affect the “trusted software” on it (such as encryption authentication and other security software). Structural coding or environmental encryption can effectively block or quantitatively control the internal and external effects of inherent security contradictions: even if there are ESS problems, it is difficult for attackers to exploit them or it is difficult for them to change into ESS events; even if the TEE platform lacks a trusted “Enclave” or “absolutely safe trusted root,” it can still ensure that the platform has security quality indicators that can be quantifiably designed and verifiably measured. In short,

“structural coding or environmental encryption” makes the attempts of system attacks that bypass “information encryption” futile in most cases.

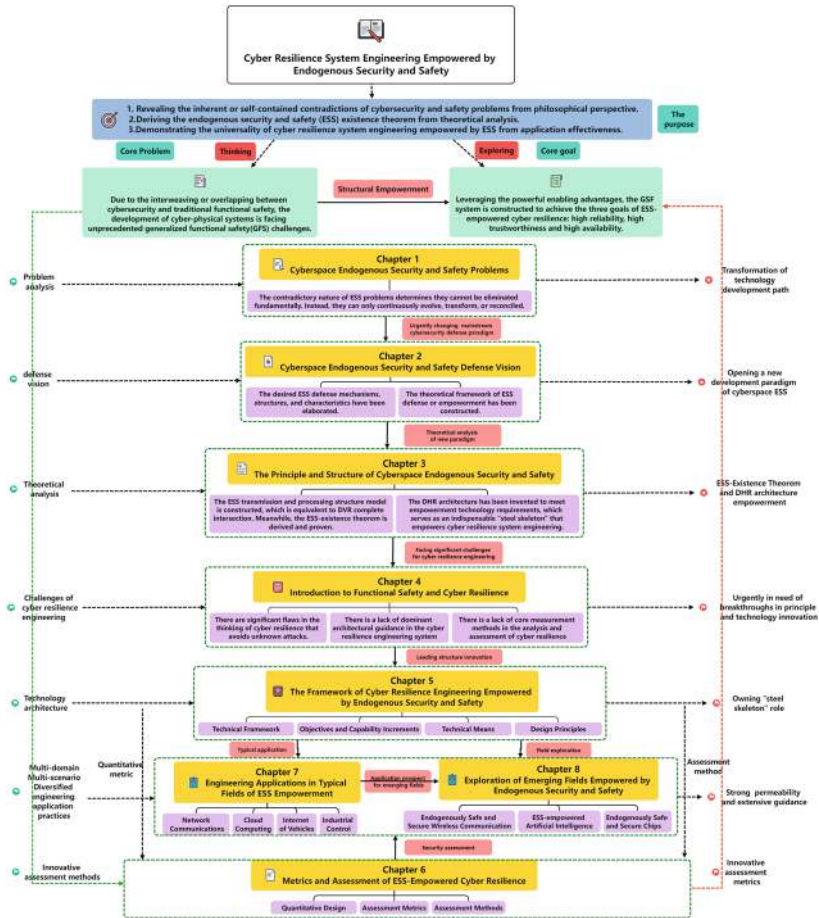
The author’s research team has been dedicated to the original theory and subversive technology research in the cybersecurity field over the past 10 years. From the initial theory and method of cyberspace mimic defense (CMD) based on the natural mimicry camouflage phenomenon and the enlightenment of the Relatively-true Axiom, to the creation of the “impossible triangle” universal deconstruction analysis model IIP based on the three elements of DVR, and then to the conjecture that there may be ESS in the completely intersecting set of DVR, the author deduces the ESS Existence (ESS-ET) Principle based on structural coding, and gives the ESS analysis conclusion based on cryptography principle, which not only proves that the inference of “structure determines security” is established in theory, but also created a new knowledge system of “structural coding or environmental encryption” until the DHR empowerment architecture is invented, which can solve multiple overlapping problems such as functional safety, network security, and even information/data security. Finally, through continuous iterative researches and technological innovation, a more scientific and general theory and methods of cyberspace endogenous security and safety (CESS) have been established.

This book is suitable as a reference for graduate students in cybersecurity, cyber resilience engineering, confidential computing/trusted computing, integrated safety, cyber-physical systems, and industrial control, or for professionals in related disciplines. It is also suitable for those who are interested in practicing the ESS architecture. It is of introductory significance for researchers and engineers who are interested in practicing ESS architecture-empowered cyber resilience and trusted service technology innovation, or who are interested in iteratively improving ESS theory and methods, or are committed to exploring derivative application fields.

To make it easier for readers to selectively read relevant content, a relationship view of different chapters is attached.

For the successful publishing of this book, the author would like to sincerely thank Professor Wei Qiang, Professor Li Yufeng, and Associate Professor Liu Wei who participated in the writing of the relevant sections of Chap. 1; Professor Jin Liang, Associate Research Fellow He Lei, Dr. Ren Quan, Dr. Hu Xiaoyan, etc., who participated in the writing of Chaps. 2 and 3; Professor Ji Xinsheng who participated in the writing of relevant sections of Chap. 4; Dr. Zhang Jin, Professor Hu Hongchao, and Associate Research Fellow Jiang Yiming who participated in the writing of parts of Chap. 5; Professor Yi Peng and Professor Ma Hailong who participated in the writing of relevant sections in Chap. 6; Associate Professor Cheng Guozhen, Associate Professor Song Ke, and Associate Professor Zhang Zhen who participated in the writing of parts of Chap. 7; Professor Huang Kaizhi, Dr. Li Yu, Associate Research Fellow Zhang Fan, Associate Research Fellow Huang Ruiyang, Associate Research Fellow Guo Wei, Associate Professor Zhong Zhou, Associate Research Fellow Yi Ming, etc., who participated in the writing of the relevant sections of Chap. 8; Research Fellow Liu Caixia and Professor Qi Jianping were

responsible for the organization, verification, and revision of the manuscript. Last but not least, special thanks are extended to the Purple Mountain Laboratory, Songshan Laboratory, and Zhejiang Laboratory for their long-term support and help.



Chapter relationship view

Zhengzhou, China
February 2024

Wu Jiangxing

Brief Introduction

This book systematically expounds the knowledge system of cyberspace endogenous security safety (CESS or more simply ESS), emphasizing the theoretical role and technical significance of ESS-empowered cyber resilience engineering, as well as its basic role in supporting the transformation of the underlying driving paradigm of the current digital ecosystem. It reveals the strong coupling relationship between the “unknown unknown” network threat or destruction and the ESS contradiction of digital facilities; puts forward a general deconstruction model of “impossible triangle against unknown threats”; qualitatively analyzes the root cause why the current mainstream network security technology cannot effectively prevent unknown security problems; deduces the ESS existence theorem; invents the endogenously safe and secure DHR architecture with the property of complete DVR intersection; and proves the environmental encryption property of the DHR’s structural coding for the first time based on the principle of cryptography. This book theoretically answers the question why cyber resilience engineering needs the empowerment of ESS to overcome three major engineering and technical problems that need to be solved urgently, such as “inability to deal with ‘unknown unknown’ threats or destruction, lack of integrated security gains based on the structure, and inability to provide evaluation methods that can be quantifiably designed/verifiably measured.” The book concisely introduces the basic concept, framework, evaluation, measurement, and impact analysis of cyber resilience engineering, emphasizing that ESS-empowered cyber resilience engineering can bring systematic and structured security gains in four dimensions: prevention, resistance, recovery, and adaptation. It establishes an evaluation indicator system and methodology of cyber resilience evaluation based on an integrated system architecture, gives several typical application examples of ESS-empowered engineering, and introduces the research and exploration of the ESS theory in derivative application fields such as wireless communication security, artificial intelligence security, and security chip design against heavy effect upset.

This book can be used as a reference for scientists and engineers in the fields of digital product cyber resilience engineering, cyber-physical systems, network security, industrial control, intelligent networked vehicles, unmanned control, trusted execution environment, as well as university teachers and graduate students.

Contents

- 1 Cyberspace Endogenous Security and Safety Problems 1**
 - 1.1 General Concepts of ESS Problems 3
 - 1.2 Basic Characteristics of ESS Problems 5
 - 1.2.1 Universality of Contradictions 5
 - 1.2.2 Potential Hazards 6
 - 1.2.3 Multiple Expressions 6
 - 1.2.4 External Action 7
 - 1.3 CESS Problems 7
 - 1.3.1 Vulnerability and Backdoor-Related Concepts 10
 - 1.3.2 Basic Properties of Vulnerabilities and Backdoors 12
 - 1.3.3 Basic Problems of Vulnerabilities 26
 - 1.3.4 The Threats of Vulnerabilities and Backdoors 30
 - 1.3.5 Examples of ESS Problems 44
 - 1.4 Common Problems of Cyberspace Endogenous Security and Safety 46
 - 1.4.1 Definition of Common Problems 46
 - 1.4.2 Causes for Common Problems 47
 - 1.4.3 Basic Nature of Common Problems 49
 - 1.5 Generalized Functional Safety Problems 51
 - 1.5.1 Definition of Generalized Functional Safety Problems 51
 - 1.5.2 Characteristics of GFS Problems 52
 - 1.5.3 The Challenge of GFS Problems 56
 - 1.5.4 Definition of GFS 58
 - 1.6 Difficulties in Defending Against CESS Problems 60
 - 1.6.1 Advanced Persistent Threats Are Difficult to Defend Against 60
 - 1.6.2 Uncertain Unknown Threats Are Unpredictable 60
 - 1.6.3 Limited Effects of Traditional “Containment, Patching, and Repair” 64

1.7	Summary of This Chapter	68
	References	70
2	Cyberspace Endogenous Security and Safety Defense Vision	75
2.1	Current Security Defense Paradigm Problems	77
2.1.1	Concept of the Paradigm	77
2.1.2	Classification of Cybersecurity Paradigms	78
2.1.3	Defense Paradigms Need to Be Urgently Changed	81
2.2	Impossible Triangle of Defense Against Unknown Threats	83
2.2.1	Krugman's Impossible Triangle	84
2.2.2	Brewer's Impossible Triangle	85
2.2.3	Three Elements of Cyber Security Defense	87
2.2.4	Incomplete Intersection Principle (IIP)	91
2.2.5	Examples of IIP Analysis of Existing Security Technologies	94
2.3	The Vision of Cyberspace Endogenous Security and Safety Defense	96
2.3.1	Desired ESS Structure	97
2.3.2	Desired ESS Mechanism	98
2.3.3	Desired ESS Features	99
2.4	CESS Defense Paradigm	100
2.4.1	Prerequisites and Constraints	101
2.4.2	Change of Thinking Perspectives	104
2.4.3	Methodological Innovation	105
2.4.4	Update of Practice Norms	106
2.4.5	Rejecting Trial-and-Error Attacks	107
2.5	Summary of This Chapter	107
	References	108
3	The Principle and Structure of Cyberspace Endogenous Security and Safety	111
3.1	A Conjecture About DVR Complete Intersection	112
3.2	ESS Existence and DVR Transformation	113
3.3	Deriving DVR Transformation Structure from Heterogeneous Redundancy	115
3.3.1	True Relatively Axiom and Dissimilar Redundancy Structure	116
3.3.2	Qualitative Analysis of the Ability of DRS to Defend Against Unknown Attacks	117
3.3.3	Enlightenment and Discovery	119
3.3.4	DVR Complete Intersection to Be Realized by Feedback Control	121
3.3.5	Discovery of the DVR Transformation Structure	124
3.4	Principle of Existence of ESS Coding	125
3.4.1	The Concept of ESS Coding	128
3.4.2	Mathematical Model of ESS Defense	130

3.4.3	ESS Coding Existence Theorem	133
3.4.4	Strategic Ruling Feedback Control and Common-Mode Disturbance	136
3.5	DHR Architecture	139
3.5.1	Prerequisites of the DHR Architecture	139
3.5.2	DHR Architecture and Its Working Principle	140
3.5.3	Attack Surface of the DHR Architecture	145
3.5.4	Filter Effects and Double-Blind Expression	148
3.6	ESS Analysis Based on Cryptography	150
3.6.1	The Revelation of Perfect Secrecy	150
3.6.2	A Conjecture	151
3.6.3	Perfect Intrinsic Functional Safety (PIFS)	152
3.6.4	The DHR System Is the Condition and Proof of the PIFS System	159
3.6.5	Information Theory Basis of DHR's Relatively Correct Strategic Ruling Mechanism	164
3.6.6	Proof and Analysis of the Conjecture	165
3.7	Engineering and Technical Effects of the DHR Architecture . . .	169
3.7.1	Dimensionality Reduction and Transformation of GFS Problems	169
3.7.2	Local DHR	170
3.7.3	Strategic Ruling Mechanism	172
3.7.4	Entropy Nonreduction and Uncertainty	173
3.7.5	Mimic Disguise Fog (MDF)	174
3.7.6	Theoretical Prerequisites of Subverting Trial-and-Error Attacks	175
3.7.7	New Challenges of Code Injection	176
3.7.8	Natural Integration of Add-on Security Technologies	177
3.7.9	Safety Quality That Can Be Designed and Measured	178
3.7.10	Endogenously Safe and Secure TEE	179
3.8	Restricted Applications	183
3.9	Backdoors in Application Software	187
3.10	A Game-Rule Changer in Cybersecurity	189
3.11	White Box Testing of ESS Functions	191
3.12	Promotion of Diversified Ecological Development	197
3.13	Summary of This Chapter	199
	References	201
4	Introduction to Functional Safety and Cyber Resilience	203
4.1	Review and Development of Functional Safety	203
4.2	Basic Concepts of Functional Safety	206
4.2.1	Definition of Functional Safety	206
4.2.2	Attributes and Differences	207
4.2.3	Evolution of Functional Safety	209

- 4.3 Basic Connotation of Functional Safety 210
 - 4.3.1 Risk-Based Safety 210
 - 4.3.2 Levels of Safety Integrity 212
 - 4.3.3 Safety Throughout the Life Cycle 213
 - 4.3.4 Safety Under Systematic Management 215
- 4.4 Overview of Functional Safety Technologies 215
 - 4.4.1 Safety Integrity Technology 216
- 4.5 Hardware Safety Integrity Technology 216
- 4.6 System Safety Integrity Technology 217
 - 4.6.1 Random Hardware Failures and System Failures 222
- 4.7 Development Trend of Functional Safety 222
 - 4.7.1 Industry-Specific Functional Safety 222
 - 4.7.2 Complexity of Safety-Related Systems 223
 - 4.7.3 Deepening of the Unity of Opposites 224
 - 4.7.4 Resilience of Safety-Related Systems 226
- 4.8 The Basic Concept of Cyber Resilience 227
 - 4.8.1 The Origin of the Concept of Cyber Resilience 227
 - 4.8.2 The Connotation of Cyber Resilience 229
- 4.9 Framework of Cyber Resilience Engineering 231
 - 4.9.1 The Goal of Cyber Resilience 231
 - 4.9.2 The Objectives of Cyber Resilience 233
 - 4.9.3 Cyber Resilience Strategies and Design Principles 237
 - 4.9.4 Cyber Resilience Technologies 237
 - 4.9.5 Cyber Resilience Construction Plan 243
 - 4.9.6 Cyber Resilience Throughout a System’s Life Cycle 243
 - 4.9.7 Application Fields of Cyber Resilience 255
- 4.10 Cyber Resilience Analysis and Assessment 259
 - 4.10.1 Cyber Resilience Analysis 259
 - 4.10.2 Static Assessment of Cyber Resilience 259
 - 4.10.3 Cyber Resilience Attack and Defense Assessment 263
- 4.11 Challenges to and Shortcomings of Existing Cyber Resilience 264
 - 4.11.1 Major Hard Flaws Exist in the Mindset of Avoiding Unknown Attacks 264
 - 4.11.2 Major Hard Problems with Engineering Systems Lacking Architectural Guidance 267
 - 4.11.3 Significant Hard Challenges in Evaluating the Lack of Core Competence Measurement 271
- 4.12 Cyber Resilience Engineering and Manufacturing-Side Network Security 273
- 4.13 Summary of This Chapter 275
- References 276

5	The Framework of Cyber Resilience Engineering Empowered by Endogenous Security and Safety	279
5.1	Overview of Cyber Resilience Empowered by Endogenous Security and Safety	279
5.1.1	New Thinking Perspectives	279
5.1.2	New Theoretical Basis	282
5.1.3	New System Architecture	286
5.1.4	New Evaluation Mechanism	289
5.2	Incremental Capability for Achieving the Goals of ESS-Empowered Cyber Resilience	291
5.2.1	Seeing the Unseen	296
5.2.2	Refusal Trial-and-Error	301
5.2.3	Stop Loss and Recover	302
5.2.4	Iterative Upgrade	303
5.3	Technologies of Cyber Resilience Engineering Empowered by ESS	304
5.3.1	Technology Tree	304
5.3.2	Root Technologies	305
5.3.3	Branch Technologies	307
5.3.4	Leaf Technologies	309
5.4	Design Principles of ESS-Empowered Cyber Resilience	312
5.4.1	Principles for Setting Up Critical Defensive Positions	312
5.4.2	The Principle for Unknown Threat Analysis	313
5.4.3	The Principle for System Architecture Application	315
5.4.4	The Principle of Security Technology Collaboration	316
5.4.5	The Principle of Strategic Ruling Feedback	320
5.5	Summary of This Chapter	329
	References	330
6	Metrics and Assessment of ESS-Empowered Cyber Resilience	331
6.1	A New Perspective on the Assessment of Cyber Resilience Empowered by the ESS Architecture	331
6.1.1	Current Perspectives on Cyber Resilience Assessment	331
6.1.2	A Cyber Resilience Assessment Perspective Based on the System Architecture	333
6.2	System Architecture-Based Cyber Resilience Assessment System	335
6.2.1	Assessment Framework	335
6.2.2	Metrics	338
6.2.3	System Architecture Scoring	340
6.3	Assessment Method	345
6.3.1	Overview	345
6.3.2	Static Assessment	351

6.3.3	Adversarial Assessment	355
6.3.4	Disruptive Assessment	358
6.4	Relevant Regulations and Standards	364
6.5	Summary of this Chapter	366
	References	366
7	Engineering Applications in Typical Fields of ESS	
	Empowerment	369
7.1	ESS Engineering Construction Baseline	369
7.2	ESS-Empowered Network Communications	371
7.2.1	Endogenously Safe and Secure Routing and Switching Equipment	371
7.2.2	Endogenously Safe and Secure Network Control System	381
7.3	ESS-Empowered Cloud Computing	388
7.3.1	Cloud Computing and Cloud Native	388
7.3.2	Threat Analysis	391
7.3.3	Design Idea	393
7.3.4	Architecture Design	395
7.3.5	System Practice	396
7.4	The Internet of Vehicles System Empowered by ESS	401
7.4.1	T-BOX Threat Analysis	401
7.4.2	Design Ideas	406
7.4.3	System Architecture Design	409
7.4.4	Design of Functional Units	411
7.4.5	Feasibility and Safety Analysis	420
7.4.6	Attack and Defense Examples	423
7.5	ESS-Empowered Industrial Control Systems	427
7.5.1	Threat Analysis	427
7.5.2	Design Ideas	430
7.5.3	System Architecture Design	432
7.5.4	Design of Functional Units	436
7.5.5	Safety Practices	441
7.6	Summary of This Chapter	443
	References	444
8	Exploration of Emerging Fields Empowered by Endogenous Security and Safety	447
8.1	Endogenously Safe and Secure Wireless Communication	447
8.1.1	Development Paradigms of Wireless Communication	447
8.1.2	Wireless ESS Problems	453
8.1.3	Wireless ESS Attributes and Architecture	454
8.1.4	The Assumption About the Wireless ESS System and Mechanism	458

- 8.1.5 Functions and Technologies of Wireless ESS 459
 - 8.1.6 Wireless ESS Performance Analysis 463
 - 8.2 ESS-Empowered Artificial Intelligence 466
 - 8.2.1 An Introduction to Artificial Intelligence Application Systems 466
 - 8.2.2 Analysis of Security Threats Faced by AI Application Systems 467
 - 8.2.3 Design of an Endogenously Safe and Secure AI Defense Framework 471
 - 8.2.4 ESS Experiment of AI 473
 - 8.3 Performance Analysis of Endogenously Safe and Secure Chips 484
 - 8.3.1 A Brief Introduction to Endogenously Safe and Secure Chips 484
 - 8.3.2 Uncertainty of the Advantages of Endogenously Safe and Secure Chips in Soft Error Rate Assessment 486
 - 8.3.3 Quantitative Analysis of Chip Safety and Security Performance Based on the Markov Model 487
 - 8.3.4 Simulation and Analysis of Anti-SEU Capabilities of Chips with Different Architectures 494
 - 8.4 Summary of This Chapter 497
- Appendix 498
- References 499

About the Author



Wu Jiangxing born in 1953, is a famous scientist and engineer in the fields of network and communications, computer architecture, and cybersecurity in China. He is currently the director of the National Digital Switching System Engineering and Technological Research Center (NDSC), the director of Songshan Laboratory, the chief scientist of the Purple Mountain Laboratories, the chief scientist of the Zhejiang Laboratory, and the dean of Fudan University Big Data Research Institute. He was elected in 2003 as an academician of China Academy of Engineering. In the 1980s, he invented revolutionary technologies in the field of digital program-controlled switching such as “software-defined functions, replicated T-shaped digital switching networks, and hierarchical distributed control structures.” In 1991, he presided over the successful development of China’s first large-capacity digital program-controlled switch—HJD04, which promoted the global rise of China’s high-tech communication industry. Since the early days of the twenty-first century, Wu has successively invented major original technologies in the field of network communication, such as “all-IP mobile communication, the asynchronous switching network with variable length packets, reconfigurable flexible network architecture and IPTV based on router selection and transmission mechanisms.” He presided over the successful R&D of the world’s first all-IP-based composite mobile communication system—CMT, China’s first high-speed core router, the world’s first large-scale convergence access router ACR supporting IPTV services and other information

communication network infrastructure or equipment. In 2006, he led the successful development of “China High-speed Information Demonstration Network—3Tnet” and completed the first large-scale stress test of HDTV over IP network in the Yangtze River Delta region of Jiangsu, Zhejiang, and Shanghai, which proved that it is feasible in theory and technology for the Internet to carry HDTV services on a large scale. In 2008, he proposed a “multi-dimensional reconfigurable software and hardware collaborative computing architecture based on active perception—Mimic Computing Architecture (MCA)” for domain computing. In 2013, he developed the world’s first high-performance computer prototype system based on the mimic architecture, which was rated by academicians of the Chinese Academy of Sciences and the Chinese Academy of Engineering as one of the “Top Ten Scientific and Technological Achievements in China in 2013.” Also in 2013, he founded the theory of Cyberspace Mimic Defense (CMD) based on the mechanism of endogenous security and safety of structural coding/environmental encryption. Since 2017, he has published the Chinese monographs *Introduction to Cyberspace Mimic Defense*, *Principles of Cyberspace Mimic Defense*, *Cyberspace Endogenous Security and Safety*, and the English monograph *Cyberspace Mimic Defense: Generalized Robust Control and Endogenous Security and Safety*. In 2018, he proposed the theory and methodology of Polymorphic Intelligence Networking Environment (PINE) to change the development paradigm of the network technology system. In 2021, the world’s first PINE principle verification system developed under his leadership passed national acceptance at Zhejiang Laboratory. In 2023, he published the Chinese version of his monograph *Cyber Resilience Engineering Empowered by Endogenous Security and Safety*.

Since 1995, Wu Jiangxing has successively won four first prizes and four second prizes for national scientific and technological progress. He received the Ho Leung Ho Lee Science and Technology Progress Award and the Science and Technology Achievement Award in 1995 and 2015, respectively. The NDSC network and switching research team led by him won the Innovation Team Award of the 2015 National Science and Technology Progress Award.

Chapter 1

Cyberspace Endogenous Security and Safety Problems



While people are enjoying the new living and working experiences that digital, networking, and intelligent technologies have brought to the modern society, cyberspace security and safety problems have become a lingering specter, threatening the basic order and rules of the digital world at all times, and trampling on the common values and codes of conduct of the human society without any bottom line. In uneasy anxiety and unintelligible fear, people have been struggling to think about the causes for all this and why the “ancestral tactics” of “fighting back the enemy with soldiers headed by generals” or the modern defensive tactics based on all-dimensional perception and precision strike are no longer effective. Even if defensive positions are deployed in great depths and even “zero trust architecture” technical arrangements are adopted, they are still used as a stage or arena by those “ghostly” network hackers from time to time to show their muscles or skills. In particular, today’s mainstream security defense technologies based on prior knowledge and various knowledge bases, the patching methodology of “closing doors and blocking holes,” and the practical norms of “best bodyguard-style services,” all of which constitute a cybersecurity defense paradigm, is increasingly unable to deal with advanced persistent threats (APTs) based on the design defects of the underlying driver technology of the digital ecosystem. In other words, how to prevent “unknown” man-made attacks that have crossed the current boundary conditions of network security defense has become a severe challenge that the development of cyber-physical systems (CPS) and digital industry have to face in the digital society today.

In the Internet age, there is a very strange phenomenon that people are blind to, that is, the design of almost all products or cyber-physical systems containing digital elements “selectively ignores” cybersecurity quality problems, at least in designs that are strongly related to intrinsic functional robustness or functional resilience, without considering the potentially devastating impact of unknown cyberattacks (i.e., the so-called normalization of deviation). As a result, most of the software or hardware products sold in the current market, including various plug-in or add-on protective products that claim to be the “patron saint of cybersecurity,” never provide cybersecurity-related technical indicators, and even have no minimum

warning tips. It seems that cybersecurity problems, just like information security problems, are only due to insufficient protection measures for user-side private information or service functions and that they have nothing to do with the design and manufacturing quality of the digital products themselves. Therefore, manufacturers of digital products do not have to bear any legal or moral responsibility for the tangible or intangible (even property and life) losses that may occur during the whole product life cycle. At most, they will release some free software patches afterward or promise to improve them in subsequent upgraded versions. What is even more ridiculous is that while people are intoxicated with the magical experience brought by various digital products, they seem to have calmly accepted the reality that there is no security quality, and they are no longer surprised to solve the “deviation normalization” status quo of “bodyguard service” of digital facilities out of their own pockets. The problem of attacks and sabotage in cyberspace has been a “lingering ghost” in a world where “humans, machines, things, and networks” are deeply integrated. We cannot help but ask why the current network security defense paradigm has not been able to reverse this frustrating security situation so far. As for the related causes, the author believes that for a long time, people have not realized that network security problems are largely caused by the inherent or endogenous contradictions in the digital ecosystem. Hegel, a master of German philosophy, pointed out: “Everything itself is self-contradictory, and contradictions are the root of all movement and vitality” [1]. According to dialectics, contradictions can be transformed or reconciled; they can achieve a unity of opposites, but they can never be completely eliminated. Therefore, if we try to deal with the inherent security conflicts in cyberspace merely by patching or layer-by-layer protection, we are doomed to fail to achieve the goal of exhausting network security problems such as the design defects of the underlying drivers of the digital ecosystem or software and hardware bugs. Accordingly, the author divides cyberspace security problems into the ESS problem domain and the non-ESS problem domain. The former belongs to endogenous or self-contradictory problem domain, which include two subdomains: the subdomain of common ESS problems and that of individual ESS problems, as shown in Fig. 1.1.

Common problems generally involve a large area, have a wide range of influence, and usually require universal solutions, while individual problems often require special solutions. For example, software and hardware code design flaws (vulnerabilities) or the implantation of backdoors and trapdoors through the supply chain of the digital ecosystem are common problems for which universal solutions must be sought. However, the inexplicability, unpredictability, and noninferability of the results of artificial intelligence based on the neural network algorithm belong to individual problems that need special solutions. This book will focus on the cybersecurity and safety problems caused by inherent or self-contained contradictions, mainly discussing how to evolve, transform, or resolve the common ESS contradictions in cyberspace at the theoretical and technical levels and obtain a “solution to seek coexistence with problems in the defective space” under given conditions.

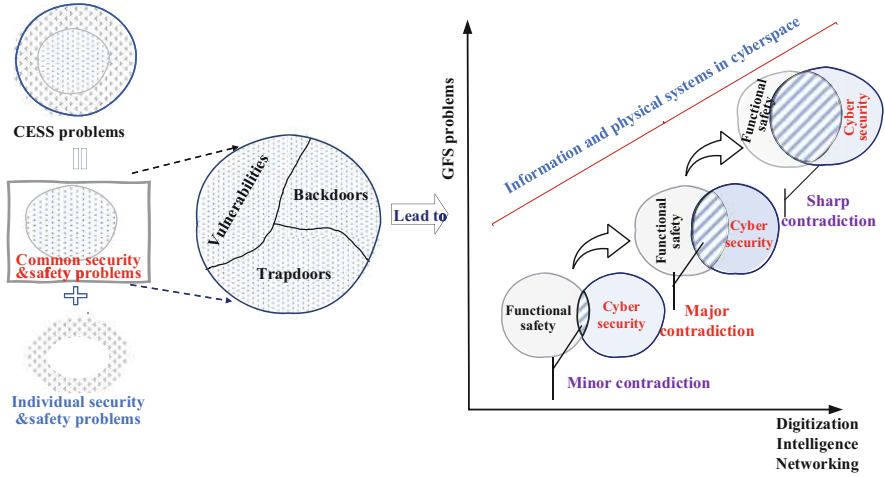


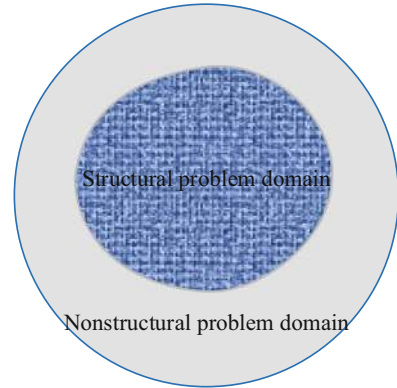
Fig. 1.1 Cyberspace security and safety problem domain

1.1 General Concepts of ESS Problems

In a general philosophical sense, there is no “if and only if function” in a logical or mathematical sense in nature or artificial systems, that is, there is nothing without contradictions or defects. From the perspective of the classical reliability and traditional functional resilience theory, no artificially designed and manufactured physical or logical entity is “perfect,” and there are always unexpected “dark functions.” Cognition-limited design rules and their chain combinations are not enough to deal with various exceptions that cannot be anticipated in advance. Under the influence of various disturbances, there are always different degrees of functional failure under different preconditions in the whole life cycle of a system [2]. If anything has undesirable (or unexpected) side effects or unknown unknown functions in addition to its intrinsic functions, or if there are interdependent and contradictory “endogenous problems” determined by the architecture of a system or model, they are called ESS problems [3]. ESS problems can be classified into two domains: the structural problem domain and the nonstructural problem domain, as shown in Fig. 1.2.

According to the viewpoint of the unity of opposites of materialist dialectics, ESS problems are essentially the external manifestation of the mutually dependent and mutually exclusive relationship between two or more opposites within a thing and are an inseparable part of the thing itself [4]. It can be seen that ESS problems are determined by structural contradictions. They cannot be dealt with in isolation, let alone fundamental elimination. They can only evolve, transform, or reconcile (the so-called resolution of contradictions does not mean that contradictions and their opposites no longer exist but that they exist in reconciliation or exist in a relationship of unity of opposites). For instance, through continuous improvement of

Fig. 1.2 Classification of structural and nonstructural problems



technologies and processes, we can gradually reduce but cannot completely avoid the problem that carbon emissions from coal-fired and oil-fired power generation still exceed the circulation and purification capabilities of nature. If fossil fuel power generation is converted into nuclear power generation, the problem of atmospheric pollution caused by carbon emissions can be fundamentally solved, but the leakage and pollution of nuclear materials caused by deep well or ocean trench landfill treatment of nuclear waste will become a new environmental problem. Similarly, to ensure the confidentiality, availability, and integrity of network information, digital encryption and authentication have become an indispensable technical measure. However, this will also bring a lot of inconvenience to the use of digital resources and related services, especially for people with memory impairment or the elderly who will feel that the operation interface is very unfriendly. For another example, smartphones can bring great convenience to people in terms of communication, Internet browsing, online video viewing, electronic games, travel navigation, online shopping, electronic payment, etc. However, cybersecurity problems can also lead to the leakage of sensitive information or the loss of private property. In particular, Professor von Neumann, the founder of the modern computer architecture theory, pointed out in his related works in the 1940s: “Even simple (stored program controlled) computing systems can exhibit complex behaviors, and the possibility of errors is inherent in all computing systems, and unexpected problems may arise in the process of use and development.” At the same vein, the control mechanism of stored programs has the inherent contradiction of “being good or being evil.” In other words, there is a “genetic defect” in today’s cyberspace, which is strongly related to the weakness (vulnerability) of software and hardware code design. Man-made attack or destruction will bring unprecedented challenges to preventing systematic network security risks, which is also the essential reason why cybersecurity threats are getting worse.

It needs to be emphasized that any process of evolution, transformation, or reconciliation of contradictions must come with additional costs. For example, if renewable energy sources such as wind energy and solar energy are used to replace fossil fuel or nuclear power, the contradiction between atmospheric, surface, or

ocean pollution will be transformed into the technical, economic, and technical cost of deploying costly complex power storage and intelligent power transmission and distribution systems to utilize these energy sources. Although network physical isolation technologies enable us to cut off (in fact, it is difficult to completely cut off) the physical or logical connection between private networks and external networks, it will also bring serious “island effect” to the use of information or data. However, as long as the benefits of the transformation or reconciliation of major contradictions far outweigh the cost, the relevant inputs and costs will be acceptable. For instance, to prevent cyberattacks, people are willing to invest in additional security protection measures such as firewalls, intrusion detection, “antivirus and anti-Trojan” software, or encryption authentication in their terminals or servers and can afford the maintenance and upgrade service fees of related security products throughout their life cycle. For another example, the market price of smart connected cars, which can ensure both functional safety and cybersecurity [5], is definitely higher than that of products without such performance. “Although design-based and default-based security may increase product development costs (mere software development costs will increase by at least about 19%), it can greatly improve customer safety, and in the long run, it can reduce the maintenance and repair costs of manufacturers and the life-cycle use costs of end users” [6].

In short, the philosophical essence of ESS problems is caused by the structural contradiction of the target object, and the multifaceted nature of things determines the inevitability and universality of the existence of ESS problems. As can be seen from Fig. 1.1, events such as “unforeseen disasters” that do not constitute an endogenous contradictory relationship do not fall into the category of ESS problems. Air pollution caused by fossil fuels is an ESS problem, but there is no structural contradiction between coal-fired power plants and nuclear radiation or nuclear pollution.

Scientific knowledge tells us that there is no irrelevance between ESS problems and non-ESS problems or between structural contradictions and nonstructural contradictions. In fact, there is a relationship between local and global problems or a relationship between primary and secondary contradictions. Especially when the premises of the problem or the target object change, there is also a transformation between the two. For example, as far as the global environment is concerned, both fossil fuel and nuclear fuel power generation have different degrees of pollution. Even hydropower generation, solar and wind power generation will bring short-term or even long-term negative impacts to the local environment.

1.2 Basic Characteristics of ESS Problems

1.2.1 *Universality of Contradictions*

In addition to their intrinsic functions, things also have contradictory functions such as derivation and explicit expression. That is to say, the two sides of the contradiction must be interdependent, and the prerequisite for the existence and development

of one side of the contradiction must be that the other side of the contradiction also undergoes corresponding changes. They are mutually dependent and prerequisites for each other. For example, artificial intelligence has the dual nature of “being good” and “being evil.” In addition, in a universal rather than individual sense, ESS problems are structural contradictions and inseparable, and unity of opposites is its inevitable expression. For instance, the software and hardware design defects of smartphones can only be gradually exposed and solved in the process of large-scale or long-term use, and there is no way to eliminate them once and for all on the design or manufacturing side. Therefore, when we deal with cybersecurity problems, we must analyze the contradictory nature of things, study the existence conditions and possible transformation relationships of primary and secondary contradictions, and understand and handle them scientifically to avoid falling into the dilemma of getting half the result with twice the effort.

1.2.2 Potential Hazards

As an old saying goes, “it’s easy to dodge an open attack but hard to guard against a hidden arrow.” It is possible to take preventive or evasive measures for explicitly expressed problems, but implicitly expressed ESS contradictions are often more threatening or destructive because of their concealment and unknown attributes. For instance, the product instructions can usually tell you whether the explicit side effects of a drug are toxic or not but cannot give any information about potential toxic side effects. For another example, whether there is a backdoor in the open-source software and hardware itself, whether the trusted root of trusted computing is credible, and whether the confidential computing platform belongs to the “soul torture” in the cybersecurity field such as trusted executive environments (TEE) all belong to the category of potential threats or destruction.

1.2.3 Multiple Expressions

Like all contradictory problems, ESS problems have both individualized and common expressions and even mixed forms of expressions (as shown in Fig. 1.3). There are always differences between primary and secondary contradictions, structural or nonstructural contradictions, and global and local contradictions in different scenarios or under different premises. Whether individual or local problems have special solutions depends on specific analysis. But for common problems, long-term absence of a universal solution will lead to serious overall impacts.

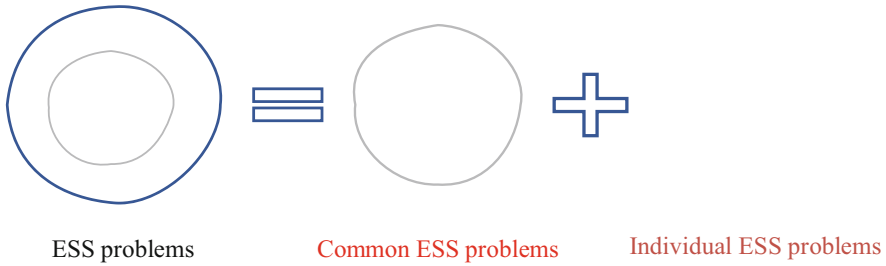


Fig. 1.3 Multiplicity of ESS problems

1.2.4 External Action

ESS problems are inherent structural contradictions, which usually require the establishment of external triggers before they can lead to ESS events or incidents, that is, only “two hands can clap loudly.” The philosophical interpretation is as follows: “Internal causes are the source of changes in things, external causes are the conditions for changes in things, and external factors work through internal factors” [4]. Carbon dioxide will certainly be produced during coal combustion, but if there are no external conditions such as ignition factors and oxygen, coal itself will not automatically release carbon dioxide. Similarly, in a structure empowered by stored program control, if an unknown software/hardware code design defect is not disturbed by external factors or driven by events outside the design and test set, the defect itself will not automatically become a fault beyond the expected function. In other words, there is not necessarily a connection between ESS problems and ESS incidents.

1.3 CESS Problems

As we all know, Alan Mathison Turing, a mathematician, proposed an abstract computing model—Turing machine—which answered the question of what is computable, and John von Neumann, another mathematician, proposed a stored program control architecture to solve “the computing problem of how to control the direction of the program.” Subsequent generations of computer scientists and engineers have devoted themselves to the research and practice of how to improve computing reliability and processing performance, lower the threshold of use, and improve human-machine efficiency. The progress of the microelectronics science and technology has created a new era of digitalization, computerization, networking and intelligence of the human society, as well as a ubiquitous computer-controlled cyberspace. However, the ESS problems of stored program control architectures or computers have long been “selectively ignored” by scientists and the industry. In recent years, hardware vulnerabilities such as Meltdown, Spectre, and VoltJockey are even more embarrassing. It seems that such design defects involving the classic

Fig. 1.4 Associated side effects and dark functions

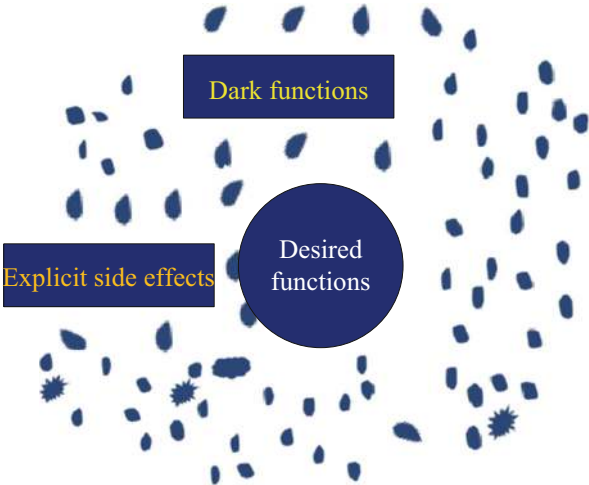
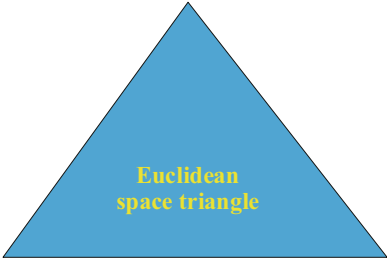
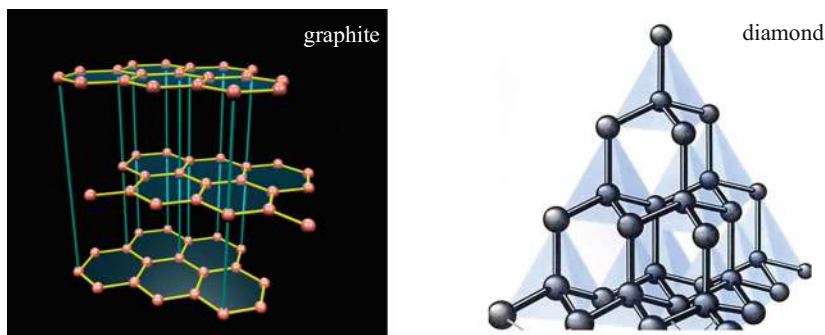


Fig. 1.5 Triangles are geometrically stable. (The following are the English terms corresponding to the Chinese terms in the figure)



computer architecture are unlikely to be perfectly remedied by pure software. Although people have long recognized the fragility (vulnerability) of software design and made unremitting efforts for it, no ideal solution has been found so far. According to the concept of ESS problems, any given or designed functions always have explicit side effects or implicit dark functions (as shown in Fig. 1.4).

The current mainstream methods of perceptual defense based on knowledge base accumulation or active defense based on active detection are all aimed at achieving the defense goal of “knowing what is happening and knowing why it is happening” through the technical route of “doing your best and making up for it before it happens.” Even the dynamic defense based on the principle of moving target defense (MTD) and the proactive protection measures based on encryption authentication have never considered the system engineering technical route based on the theory that “structure determines security,” that is, the theory and method of controlling or avoiding threats and attacks from the institutional mechanism using the structural effect of the target object. In other words, it is to try to make the system structure of the target object have quality robustness against random or uncertain disturbances to a certain extent [3]. For example, the triangular structure of Euclidean space has natural geometric stability (as shown in Fig. 1.5); the arrangement of carbon atoms



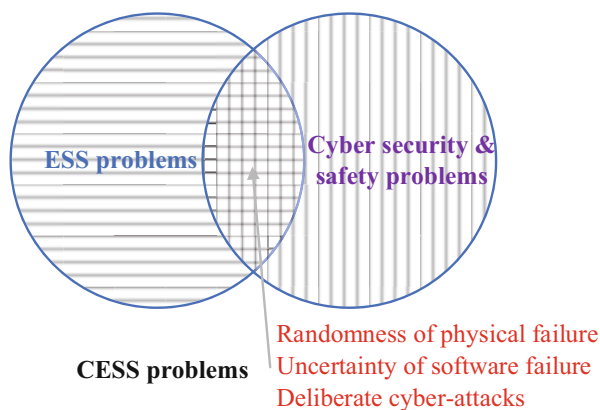
(a) Arrangement of carbon atoms in graphite (b) Arrangement of carbon atoms in diamond

Fig. 1.6 Different arrangements of carbon atoms. (a) Arrangements of carbon atoms in graphite. (b) Arrangements of carbon atoms in diamond

makes graphite and diamond have very different hardness (Fig. 1.6a, b); the unique immune mechanism of vertebrates is almost completely immune to diseases of nonvertebrates.

In fact, virtual cyberspace and physical space of the real world have the same philosophical essence but different problems and basic characteristics [7]. Observed from the phenomena in cyberspace, a stored program control system with a certain function always has explicit side effects or implicit dark functions (including deliberately set backdoors or inadvertently introduced trapdoors); seen from the code of practice of cyberspace engineering, no matter what design methods or technical measures are adopted, it is impossible to obtain a “pure function” without accompanying or derivative functions [3]. In other words, the function of “if and only if” in the mathematical sense cannot exist in the physical domain. From the perspective of cyberspace functional safety, there are both the unreliability of system functions caused by random or uncertain disturbance of software and hardware ESS problems and systematic security incidents caused by the deliberate exploitation of ESS problems. Therefore, we give the following correlated definitions: “In a digital system with a stored program control structure, if there are inevitable inherent contradictions such as code design fragility or hidden functions that are difficult to thoroughly investigate, they are called cyberspace endogenous security & safety problems (CESSPs or CESS problems)” [7]. “If the CESSPs can be exploited by man-made or non-man-made external factors, and they can produce uncertain disturbances beyond the expected function, then they are called cyberspace endogenous security & safety incidents (CESSIs or CESS incidents).” The ESSP domain is shown in Fig. 1.7.

Obviously, like the general ESS problems, CESS problems also have the basic characteristics of universality, potential harm, multiple manifestations, and external effects. However, influenced by special factors such as the ubiquity of cyberspace, the correlation between virtual and real worlds, the supply chain pattern of technologies and industries, and the integration of “man, machine, network and things,” the

Fig. 1.7 CESS problems

formation mechanism and interwoven manifestations of CESS problems are more complicated and have some special characteristics. In particular, vulnerabilities and backdoors, as the most important common ESS problems in digital systems empowered by stored program control structures, have threatened almost every corner of cyberspace and affected all aspects of digital society.

1.3.1 Vulnerability and Backdoor-Related Concepts

Regarding vulnerabilities, academic and industrial circles, international organizations, and institutions have given different definitions at different historical stages and from different perspectives, but no broad consensus has yet been formed. There have been definitions based on access control [8], definitions based on state migration [9], definitions based on security policy violations [10], definitions based on exploitable weaknesses [11–13], etc. Among them, an easy-to-understand and widely accepted definition [14] is that vulnerabilities are defects, flaws, or errors intentionally or unintentionally produced during the design, implementation, configuration, and operation of software systems or information products. Vulnerabilities exist in different forms at all levels and links of the information system, and they change as the information system changes. Once a vulnerability is exploited by a malicious subject, it will cause security damage to the information system, thereby affecting the operation of normal services of the information system and endangering the security attributes of the information system. It should be said that this definition better explains the concept of vulnerabilities from the perspective of software and systems. However, to fully understand the concept of vulnerabilities under the conditions of ubiquitous computing, Internet of Things, and complex systems we are facing today may require more in-depth discussion about the connotation and extension of vulnerabilities.

Before the advent of computers, people rarely talked about the concept of “vulnerabilities.” We may be able to understand the meaning of vulnerabilities more essentially and generally from a computational perspective. Computation is considered to be an evolutionary process based on given basic rules [15]. In fact, computation is to explore the equivalence relationship or identity between things, and the modern computer architecture created by von Neumann is a tool to process data according to a series of instructions based on the control mechanism of stored programs. In this sense, vulnerabilities can be understood as some kind of “flaws” in the implementation of computation and can be manifested as a “dark function” in a logical sense or in the form of a specific “error” in technical implementation. The existence and form of vulnerabilities are directly related to and change with the development paradigms of the computing technologies, information technologies, system application, attackers’ ability, and resources. At the same time, the persistence and timeliness, generality and specificity, and exploitability and reparability of vulnerabilities have the characteristics of unity of opposites. Backdoors generally refer to program methods that bypass security controls and gain access to a program or system. A backdoor is a method of entering a system, and it not only bypasses the existing security settings of the system but also can defeat various enhanced security settings of the system. Based on the above understanding of vulnerabilities, backdoors can be described as intentionally creating “dark functions” or “undisclosed” intrinsic functions that have logical meaning in computing implementations. As a result, proactive vulnerability discovery becomes a process of actively looking for the “dark functions” of computing or finding some kind of “error” that may change the original purpose of computing.

The vulnerabilities and backdoors referred to in this book are mainly based on the above understanding. Backdoors are often an unavoidable problem for countries that are backward in digital technologies and dependent on international markets, which is also one of the fundamental reasons why credibility cannot be guaranteed under the conditions of open industrial chains, open-source innovation chains, and globalized or multipolarized supply chains.

With the continuous improvement of the hardware technology, hundreds of millions, billions, or even hundreds of billions of transistors can be integrated on a single chip. With the increasing complexity of software systems, the code amount of operating systems has exceeded 100 million lines and that of some application systems has exceeded hundreds of thousands of lines, millions, or even tens of millions of lines. With the promotion and application of new R&D and production technologies, the automatic generation technology of application-oriented executable files, for example, makes it unnecessary to care about specific programming or coding problems, and the wide application of reusable technologies makes software and hardware engineers not need to care about specific logic implementation problems such as standard modules, middleware, IP cores, or process libraries. This has made it easier to hide backdoors or vulnerabilities in the globalized or multipolarized innovation chains, production chains, supply chains, and service chains. With the development of cyberattack technologies, the core technology of backdoor attacks has developed from the exploitation technology that relies on key vulnerabilities to

the attack technology that is strongly related to the underlying driving links of the digital ecosystem such as the digital product innovation chain and the supply chain and service chain.

1.3.2 Basic Properties of Vulnerabilities and Backdoors

As digital products are more and more abundant in the context of globalization or multipolarization, their functions will be more complex, supply chains will be more open, and there will be more design defects, hidden vulnerabilities, and preset backdoors. Practice shows that there is a rule that vulnerabilities of a system will appear and be discovered one after another over time, rather than all at once. This is due to people's limited security cognition and technical capabilities in a particular period. Therefore, some vulnerabilities cannot be found or exposed in a timely manner under the current cognitive level and technical ability and need to be gradually discovered with the improvement of scientific and technological ability. Similarly, because different groups or individuals have vastly different cognitive levels, technical capabilities, and resources, the scope of vulnerabilities exposed is also different.

Different from the unintentional nature and exploitability of vulnerabilities, backdoors are deliberately designed or reserved hidden functions (especially hardware backdoors). They usually have strong camouflage, clear functional targets, and various triggering or injection methods. In addition to using the normal input and output channels of the host system to activate and upgrade functions, they can also use sound waves, light waves, electromagnetic waves, infrared radiation, and other nontraditional side channels to penetrate physical isolation barriers to achieve covert interaction of internal and external information. Their internal-external collaborative attacks often make the traditional security protection system seem like a dummy or makes the target system "transparent in one direction" and even can easily cause permanent damage to the host system. No matter in terms of defense difficulty or in terms of direct destruction, the influence of backdoors is far bigger than that of vulnerabilities that rely on injection technologies.

In a word, unknown vulnerabilities and backdoors are always the "sword of Damocles" hanging over cyberspace and digital ecosystem, just like landmines whose burial location and detonation mode are unknown to us.

1.3.2.1 The Inevitability of Vulnerabilities and Backdoors

Since entering the Internet age, vulnerabilities and backdoors have been discovered from time to time without stop. According to statistics, there is a certain proportional relationship between the number of vulnerabilities and the number of codes. With the increase of system complexity and the number of codes, the number of vulnerabilities also increases. Because each software and hardware engineer has a different

thinking in the process of coding, their understanding of design rules cannot be completely consistent, and there will be some cognitive deviation in the process of cooperation. In addition, every engineer will have his own coding habits and thinking, so with the increase of code quantity and the complexity of system functions, various vulnerabilities such as logic vulnerabilities, configuration vulnerabilities, and coding vulnerabilities may be brought into the final application system. At the same time, due to the development of global or multipolar economy and the specialization and refinement of industrial division of labor, integrated innovation or manufacturing has become a common production organization mode. The design chains, tool chains, production chains, supporting chains, service chains, and supply chains of various products are getting longer and longer, and the scope and links involved are getting wider and wider. Untrustworthy supply chains have brought great challenges to comanagement and cogovernance of cyberspace and also provided a lot of opportunities to hide vulnerabilities and preset backdoors. These nonsubjective factors introduce vulnerabilities into the system or artificially embed backdoors into the system. So, whether explained from the perspective of technological development or the perspective of interest games, their emergence is inevitable and unavoidable.

The Contradiction Between Complexity and Verifiability

In the development of the modern software and hardware industry, to ensure product quality, the role of testing in the life cycle of products has caught universal attention. In some well-known large software and system companies, the amount of resources consumed in the testing phase has even exceeded that in the development phase. Even so, no one in theory or in engineering dares to claim to have eliminated all logical defects in software and hardware. On the one hand, the continuous growth of code volume and complexity leads to a sharp increase in the demand for verification ability. However, the verification ability at this stage is not enough to keep up with the increasing demand for computing power caused by code complexity. On the contrary, code verification does not have the ability to verify unknown and unpredictable security risks, and the more complex the codes are, the greater the possibility of unpredictable vulnerabilities will be, which leads to the contradiction between code complexity and verifiability. In addition, the completeness design of the verification rules itself has not achieved a fundamental breakthrough in theory, and with the rapid expansion of the volume of verifying object codes, the global implementation of verification rules often becomes difficult to achieve in engineering.

1. Code complexity increases the possibility of vulnerabilities. The number of software and hardware vulnerabilities is related to the number of code lines. It is generally believed that the larger the code size and the more complex the functions are, the greater the number of vulnerabilities will be [16]. First, intuitively, a significant increase in the number of lines of code usually results in a

more complex code structure and an increase in the number of potential vulnerabilities. Second, from the perspective of program logic, the logical relationships of complex codes are more complicated than those of simple codes, and the probability of logical vulnerabilities and configuration vulnerabilities will be greater. For instance, in the codes of the browser, operating system kernel, CPU and complex ASIC, and race conditions often occur in complex synchronous and asynchronous event processing. Finally, complex or large-scale program development poses a more severe challenge to designers' ability to control complex codes. Developers' technical and experience levels are uneven, and vulnerabilities are inevitable in architecture design, algorithm analysis, and coding implementation.

2. Limited verification ability is difficult to meet the challenges. There are at least three problems in the current verification ability. First, the types of vulnerabilities are complex and diverse, and it is difficult to accurately model and uniformly express them. The ability to summarize, extract, model, and match vulnerability features needs to be improved. Second, from the perspective of the computability of the problem analyzed and the scalability of the implementation algorithm, existing code program analysis methods perform path traversal and state search on the code program while meeting the requirements of path sensitivity, interprocess analysis, and context-related conditions. Problems such as "path explosion" or "state space explosion" are commonly encountered in the process, making it difficult to achieve high-level code coverage and analysis goals of traversal testing. Third, with the increasing complexity of information systems, various new problems brought about by complex hierarchical association, dynamic configuration management, and evolution derivative between systems pose bigger challenges for the discovery, analysis, and verification of system vulnerabilities. Under the comprehensive effect of these three factors, the contradiction between the increasing complexity and the limited verifiability is more prominent, and it is more difficult to discover vulnerabilities and backdoors.

Difficulties in Supply Chain Management

In the context of economic globalization or multipolarization and refined division of labor, links such as "design, manufacturing, production, maintenance, and upgrades" are arranged and deployed as a complete supply chain. Each link in the supply chain can be regarded as a "plank" of the final product, but the safety of the final product will be problematic because of one of the "short boards." With the rapid development of globalization or multipolarization and the increasingly refined division of labor, the product supply chain is getting longer and longer, its scope is getting wider and wider, and its management is becoming increasingly important and difficult. With too many links in it, the supply chain has become the target of attacks such as engineering attacks, vulnerability attacks, embedded backdoor attacks, and other attacks. For example, in Apple's XcodeGhost incident, a certain version of Xcode compiler was embedded with a backdoor, causing all programs compiled by that

version of the compiler to have preset vulnerabilities, allowing vulnerability developers to control, at will, the system running these programs. The SolarWinds incident at the end of 2020 was also a security incident caused by supply chain problems. The incident occurred when the Orion infrastructure management platform under SolarWinds was invaded by hackers, who tampered with the source codes of one of the components and added backdoor codes. Its rapid outbreak, wide spread, deep social impact and serious potential threats shocked the world, making it the most serious cybersecurity incident in the past decade.

Therefore, how to ensure the credibility of components from globalized or multipolar markets and commercial-grade untrusted sources has become a very thorny problem.

1. The importance of supply chain security: As more and more digital products and services related to the national economy and people's livelihood are released and provided through the Internet, national security and cybersecurity are becoming increasingly inseparable. Therefore, the security of various network products and their supply chains, especially the security of cyber-physical systems or digital products and their supply chains, plays a vital role in cybersecurity and even national security. For example, in the "Quantum" project, the United States took advantage of its leading position in the IT supply chains, markets, and monopoly to deploy and control its surveillance system around the world. The constantly exposed materials about the "Prism Gate" incident show that the United States uses its core advantages in chips, network equipment, and technologies to embed vulnerabilities and preset backdoors in exported electronic products, so as to realize global network intrusion and intelligence acquisition and enhance its global control and monitoring capabilities. In February 2020, Western media reported that since the 1970s, the Central Intelligence Agency (CIA) and the German Federal Intelligence Agency secretly acquired Crypto AG, a Swiss encryption company that provided 120 countries with communication encryption equipment, through which the United States obtained a lot of information. It can be seen that if a country has mastered more links in the product supply chains, especially the irreplaceable core links, it will have strategic superiority in cyberspace security and safety, including the strength to control networks, an argument that at least facts can support.
2. Difficulties in supply chain management: The main reason why supply chains are difficult to manage is that they are open and involve many links, making it impossible to adopt an ideal closed model to implement precise control. Today, with the irreversible trends of globalization or multipolarization and refined division of labor, supply chains will be longer, their scope will be wider, and their management will be more difficult. The more the links and the wider the scope may be, the greater exposed attack surface, the more uncontrollable factors, and more risks there will be. Therefore, supply chain security requires shortening the links and narrowing the scope of the chains. In theory, a closed supply chain can minimize the attack surface exposed. However, with the development of human production, the trend of globalization or multipolarization is unstoppable,

and total self-sufficiency means isolation and backwardness. Moreover, with the continuous development of sciences and technologies and the rapid accumulation of knowledge, it is increasingly difficult for people to fully master all the knowledge and technologies, so specialization and refined division of labor have become inevitable. Globalization, specialization, and refined division of labor will inevitably lead to a basic situation in which the openness, variety, collaboration, and credibility of the supply chains cannot be 100% guaranteed.

The United States mentioned in its *National Strategy for Global Supply Chain Security* that the development of information technologies and networks is an important cause for supply chain risks. The National Institute of Standards and Technology (NIST) mentioned in the “Best Practices for Cyberspace Supply Chain Risk Management” that infrastructure security is an important object of supply chain risk management. The development of electronic information technologies and Internet technologies has overcome the limitations of geographical restrictions in people’s social interactions, enabling people in all corners of the world to interact conveniently in real time. With the development of the Internet of Things, the whole process from product R&D, production, transportation, storage, sales, use to maintenance is increasingly dependent on the Internet. With the rise of industrial Internet and the introduction of network and digital elements, the whole process from product market demand analysis, product design, finalization, production process design, process planning, and organization to quality control will undergo revolutionary changes. Therefore, network infrastructure, as the basis for almost all activities in the modern society, requires a high degree of controllability and credibility. However, network infrastructure itself is a cyber-physical system that belongs to the category of software and hardware, and it is the same as other products in terms of design, production, sales, and service. Its security is subject to the credibility of the supply chains. It is difficult to envision a digital facility built with hardware and software components of questionable trustworthiness being able to secure its upper-layer applications. Under current technical conditions, the openness of supply chains and the security and trustworthiness of digital products are inherently contradictory.

Limitations of the Current Level of Theories and Engineering

From the epistemological point of view, vulnerabilities seem to be inherent in information technologies and exist in the whole life cycle of cyber-physical systems. Limited by the present scientific, theoretical, and technological level, the discovery of vulnerabilities lacks systematic theory and complete engineering foundation, and there is often hindsight, so it is impossible to detect and deal with all vulnerabilities in advance. The probability of software vulnerabilities cannot yet be assessed. Humphrey [17] of Carnegie Mellon University has conducted researches on nearly 13,000 programs for many years and believed that usually for every thousand lines of codes written by professional coders, there will be 100–150 errors. According to

this data, the Windows NT 4 operating system with 1.6 million lines of codes may have no less than 160,000 errors. Many bugs may be so minor that they have no impact, but perhaps thousands of them pose serious security threats.

1. The dilemma of program proof: Program verification and analysis based on formal methods are important means to ensure that software is correct and credible. Compared with software testing, program verification based on theorem proving has syntactic and semantic rigor as well as attribute-related completeness. Proving the security attributes of a piece of software through program analysis is also a problem that has been studied by the academic community for a long time, aiming to completely base the reliability and correctness of the program verification system on strict mathematical logic. However, many of the problems we are facing now have been proved to be shutdown problems, such as classic pointing analysis, alias analysis, etc. With the rapid expansion of the scale of software and the increasing complexity of functions, the theory and method of program correctness proof can neither cope with the completeness analysis of complex software nor make a safe and credible proof of given software, and the theorem proving path is in trouble. Similarly, the program proof of hardware code also faces the same or similar problems. In addition to proving its own security, there are two thorny problems, namely, how to prove the correctness of the patched program to fix the vulnerability and how to prove the consistency of source codes and compiled executable codes in security attributes. For example, the patch MS11-010 released by Microsoft in 2011 is a repatching of the patch MS10-011 in 2010. The reason is that the patching of the vulnerability is not complete and logical conditions can be constructed to trigger the vulnerability again. Unfortunately, as of today, such problems have not been satisfactorily resolved or effectively controlled.
2. Limitations of software and hardware testing: As an analysis method to detect program security, software and hardware testing can truly and effectively discover errors in codes. However, software and hardware testing also has two obvious shortcomings, namely, the problem of “incomplete testing.” First of all, generally speaking, it is almost impossible to consider all possible input values and their combinations and conduct exhaustive testing according to all different test preconditions. In fact, exhaustive testing of software and hardware means an astronomical number of test cases. Typically, each test can only be a sampling test. Therefore, the workload of testing must be controlled according to the risk and priority. Second, it is difficult for program testing to automatically cover all codes, and with the increase of code volume and the complexity of functional structure, the branch paths of codes will increase explosively. Under the constraints of limited time and computing resources, the soundness of program testing becomes more and more difficult to guarantee. In traditional white-box tests, even if a certain path has been followed, it does not mean that the relevant problems on this path have been well tested. The uncovered part of the codes becomes a “blind spot,” and it is impossible to draw a correct conclusion about the security of this part of the codes. Even for the codes that have been

covered, regardless of the commonly used fuzzy testing or symbolic execution technologies, the testing process is only a verification of the covered path under certain conditions, not a complete verification. Finally, and most importantly, testing can only prove the existence of defects, but it cannot prove that no defect exists.

3. The confusion about security programming: To improve code security, reduce the probability of vulnerabilities, and guide engineers to carry out security practices, secure programming specifications came into being. As one of the best security practices, security programming specifications undoubtedly promote the overall improvement of the coding security level, but there are also many shortcomings in their implementation and application. First, the summarization, enrichment, and improvement of security programming specifications are processes of experience accumulation. The templating of new problems will have a certain time delay, which means that there will be a time lag before the specifications are widely promoted. This lag will lead to the introduction of new security problems into programming practices. Second, the level of programmers varies, and there is insufficient special training. Some protocols have strict logic and timing requirements for the codes, which puts forward higher requirements for programmers to understand, master, and skillfully apply. Finally, industry competition in the Internet era has led to the pursuit of rapid version iterations of software and hardware. Generally speaking, development teams generally pay more attention to functional development than to security programming, or even selectively ignore the latter, resulting in rapid increase in the probability of code security problems.
4. Restrictions of automation level: The existing static analysis tools include products such as Coverity and Fortify that automatically check errors, as well as Fuzzing test suite tools such as Peach. The main advantage of automated tools is that they can automate program analysis and eliminate the huge workload of manual code auditing. At present, automatic analysis is mainly based on pattern matching, which has many false positives and false negatives. Vulnerability discovery is a highly experience-dependent work, and most automation tools can only be used as auxiliary tools for vulnerability discovery. From the current point of view, automated analysis is mainly based on pattern matching, which has many false positives and false negatives. Vulnerability discovery is a task that highly relies on the accumulation of experience, and most automated tools can only be used as auxiliary tools for vulnerability discovery. Balancing human-computer interaction with the accuracy of automated analysis is also a challenge faced by automated tools. In addition, vulnerability discovery often relies on the processing power provided by exhaustive algorithms, and even if technical and economic conditions permit, the processing time will be too long to be accepted.

To sum up, most of the existing vulnerability discovery theories and technologies are aimed at the analysis, discovery, and solution of known vulnerability characteristics but cannot comprehensively or thoroughly solve the problem of software vulnerabilities. For unknown vulnerabilities, in particular, the current theories and

engineering technologies are either completely ineffective or too costly. Therefore, the solution to the vulnerability problem cannot rely on the existing passive defense thinking based on feature extraction. It is necessary to develop active defense theories and technologies that do not depend on features, so that information systems have endogenous security defense attributes. The fact that information systems are easy to attack but difficult to defend can only be reversed by increasing the difficulty of vulnerability and backdoor exploitation instead of eliminating backdoors and vulnerabilities.

1.3.2.2 The Contingency of Presentation

As mentioned before, vulnerabilities are either introduced into the system because of the limitations of the coder’s thinking or problems in coding habits or are preset into the system by certain stakeholders through various means. Therefore, the presentation of vulnerabilities must be accidental. From the epistemological point of view that things can always be known, the existence and discovery of vulnerabilities are inevitable events, but the specific time, system, and way in which they appear are accidental. This involves not only the temporal limitation in understanding vulnerabilities but also the theoretical and technical ability to check the completeness of complex codes.

The Temporal Contingency of Presentation

As shown in Fig. 1.8, each vulnerability or backdoor has a life cycle. From the day it is brought into the system, the time points of its discovery, disclosure, repair and extinction, and the length of the time window in each period are all influenced by many factors, such as the development level of theoretical methods, the maturity of technical tools, the skill level of research and disposal personnel, and so on, all of which have certain contingency. In the process of continuous evolution of

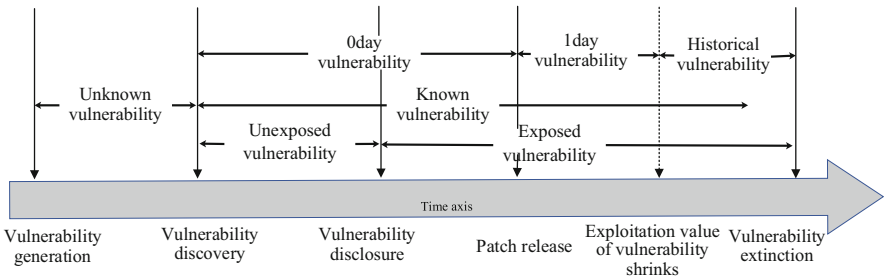


Fig. 1.8 Names of vulnerabilities in various time periods [18]

vulnerability types from code vulnerabilities, logic vulnerabilities to combination vulnerabilities, many interesting historical turning points occur. New vulnerabilities and new vulnerability types are often discovered due to a genius' flash of inspiration, and the emergence of new defense methods and mechanisms has given rise to new countermeasures and vulnerability types.

"Latent or dormant" vulnerabilities (also referred to as old-age vulnerabilities) in the program sometimes go without being exposed for more than 10 years, but they may be exposed accidentally or may be exposed as analysis tools enhance their understanding of "deeper" paths. They may also be discovered because a piece of code that people did not pay attention to was accidentally seen. As can be seen from Table 1.1, some vulnerabilities have even existed for nearly 20 years before they were discovered. These vulnerabilities are characterized by a significant "seniority" or "cinerary-casket level" reflecting their great randomness in the time of discovery.

The temporal features of vulnerability presentation reflect the development process of people's cognition of vulnerabilities and the process of knowledge accumulation. Vulnerabilities may be discovered with the occurrence of a bug, or they may be discovered through the proof of a certain theory, that is, the final presentation of vulnerabilities is often accidental. Based on the time when vulnerability types are known to the public (published documents or early vulnerability numbers), as a reference, Fig. 1.9 shows the discovery time of the nine most typical vulnerability types in both web and binary codes, illustrating the temporal features of vulnerability presentation. Among them, the buffer overflow vulnerability [19] was discovered earliest in the 1980s. Successively discovered after the 1990s were race condition vulnerabilities [20], SQL injection vulnerabilities [21], and format string vulnerabilities [22]. Since the early days of the twenty-first century, new types of vulnerabilities have been emerging, such as integer overflow vulnerabilities [23], XSS vulnerabilities [24], server-side request forgery vulnerabilities [25], PHP deserialization vulnerabilities [26], and use-after-free vulnerabilities [27].

The Contingency of the Way of Presentation

As far as the statistical law of vulnerability distribution and the chance and occasion of exposure are concerned, the emergence of vulnerabilities is often quite accidental. Vulnerability distribution can be affected by many factors such as research hotspots, new product release, technological breakthrough, product popularity, and even economic benefits, and their statistical characteristics such as quantity change, type distribution and proportion are irregular. Figure 1.10 shows a total of 163,476 vulnerabilities collected by China National Vulnerability Database (CNVD) from 2012 to 2022. It can be seen that the number of vulnerabilities discovered every year fluctuates statistically.

In addition, there is great uncertainty about how and when vulnerabilities are exposed. The ways of exposure include the following: analysis after capture of attack samples; APT attacks are exposed; researchers themselves publish vulnerabilities (often called irresponsible disclosure); manufacturers announce the repair; and

Table 1.1 Duration of existence of some “old-age” vulnerabilities

Vulnerability name	Vulnerability number	Vulnerability type	Discovery time	Duration of Existence	Hazards
LZO vulnerability	CVE-2014-4608	Buffer overflow	July 2014	20 years	A remote attacker could exploit this vulnerability to cause a denial of service (memory corruption)
Old-age vulnerability	MS11-011	Privilege escalation	February 2011	19 years	Gain the highest authority on the system
Local privilege escalation vulnerability	MS10-048	Privilege escalation	January 2010	17 years	Gain the highest authority on the system
MY power system “credential-stuffing” vulnerability		SQL injection	March 2012	10 years	Can cause the database address to leak out, causing the privacy of website users to be leaked
Shell-breaking vulnerability	CVE-2014-6271	Operating system command injection	September 2014	10 years	Remote attackers can execute arbitrary code using specially crafted environment variables
Dirty cow vulnerability	CVE-2016-5195	Race condition	October 2016	9 years	A malicious user could exploit this vulnerability to gain high privileges
Phoenix Talon	CVE-2017-8890, CVE-2017-9075, CVE-2017-9076, CVE-2017-9077	Remote code execution	May 2017	11 years	It can be used by attackers to launch DOS attacks and can lead to remote code execution if certain exploitation conditions are met, including TCP, DCCP, and SCTP at the transport layer, and impacts on IPv4 and IPv6 protocols at the network layer
WinRAR Directory	CVE-2018-20250	Remote code execution	February 2019	19 years	It allows files to be written into boot-up items during the

(continued)

Table 1.1 (continued)

Vulnerability name	Vulnerability number	Vulnerability type	Discovery time	Duration of Existence	Hazards
Traversal Vulnerability					decompression process, resulting in code execution, and the attacker can completely control the target computer

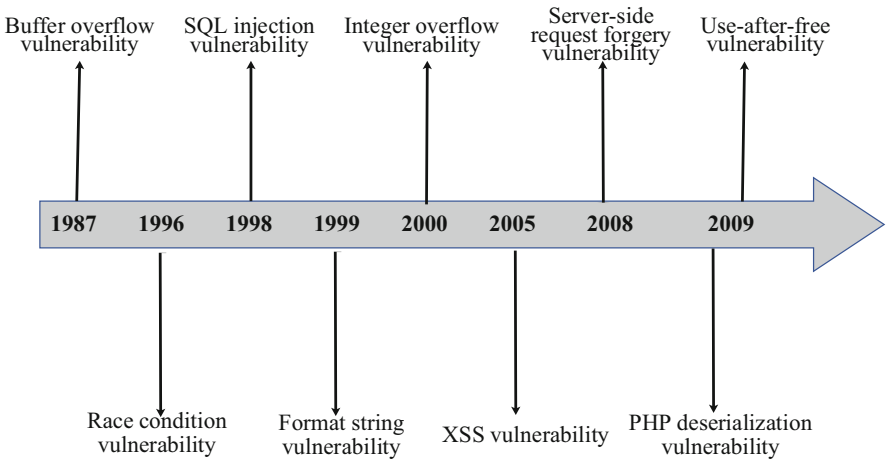


Fig. 1.9 Time when some important types of vulnerabilities appeared

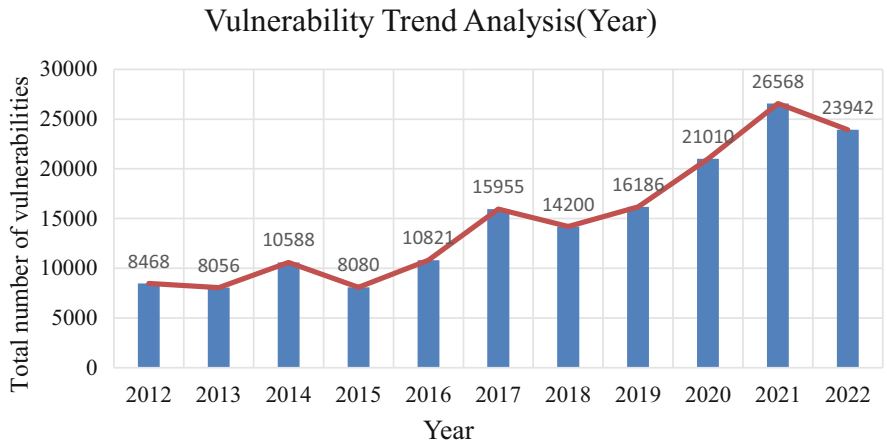


Fig. 1.10 Distribution of vulnerabilities collected by CNVD from 2012 to 2021. (Data from CNVD official website)

exposure of use by contestants in contests (such as the 0-day vulnerability in PWN contests). So, the way of exposure of a specific vulnerability is unpredictable.

1.3.2.3 Spatial-Temporal Characteristics of Cognition

People's cognition is an accumulation process, whether it is the accumulation of individual cognition under constraints or the accumulation of overall human cognition over time. As Newton said, we succeed because we stand on the shoulders of giants. Everyone's cognition has certain limitations. Especially in the era of data explosion, information explosion, and knowledge explosion, no one can master all fields of knowledge. Therefore, human society needs to form a hierarchical and domain-specific cognitive pattern, so as to conduct continuous and in-depth research in different fields and different professional directions and to continuously improve human beings' overall cognition of nature. It is this process from the individual to the whole and "from the beginning to the end" that makes human cognition have spatial-temporal characteristics, which is also true of human cognition of the spatial-temporal nature of vulnerabilities. A system considered safe today may not be safe tomorrow; a system considered safe by one person may not be safe in the eyes of another person; a system that is safe in one environment may not be safe in another. This is the spatial-temporal difference that vulnerabilities present due to cognition.

The Process from Quantitative Change to Qualitative Change

There are always vulnerabilities, but the discovery of vulnerabilities has a temporal attribute, and it requires continuous accumulation of knowledge to a certain level for vulnerabilities to be discovered. Take the Rowhammer vulnerability [28] that shocked the world in 2014 as an example. Initially, the vulnerability was just a phenomenon discovered by researchers at Carnegie Mellon University who were engaged in integrated circuit research. Repeated use of machine code instructions CLFLUSH or Cache Line Flush can clear the cache and force reads and updates. The researchers found that if this technique is used to force the memory to read repeatedly and charge a row of capacitors, it will cause a large number of bits to flip, which is the so-called Rowhammer phenomenon. Google's Project Zero team, as a security research group, paid attention to this achievement and conducted in-depth research. They found that malicious programs could use bit-flipped Rowhammering to run unauthorized codes. The team designed a method to redirect the CPU, run the code from the wrong memory address, and use Rowhammering to change the content of the operating system's memory map. Four researchers from Vrije Universiteit Amsterdam successfully launched an attack on Windows 10 by combining Rowhammering with a newly discovered method of storing deduplication vectors. Even if the system was fully patched and various security reinforcement measures were in place, this approach could still allow the attacker to take control of the operating system. Hardware vendors are trying to introduce measures or functions to

prevent or reduce Rowhammering vulnerabilities in the DDR4 architecture, but research by Google and Third I/O companies showed that DDR4 was not immune to Rowhammering either. The process from quantitative to qualitative change is reflected in the increasingly widespread application of certain types of information systems or the increase in the number of certain types of vulnerabilities that attract more attention and lead to the discovery of new problems in such information systems. It is also reflected in the fact that more in-depth research on the exploitation of certain types of vulnerabilities will trigger the revolution of vulnerability exploitation technology and then lead to more attacks involving such vulnerabilities. For example, cloud computing is widely used at present, so some people are looking for vulnerabilities in virtual host penetration and virtual host allocation algorithms based on the multitenant and virtualization characteristics [29]. Take array out-of-bounds as an example. In the 1970s, people never checked for the problem of array out-of-bounds because it was generally considered a data integrity error in the C language. Given the computing power, automatically checking for this problem would reduce program execution efficiency. Considering the cost-effectiveness ratio, the usual practice was to leave this problem to the programmers, without realizing that array out-of-bounds may cause serious security hazards. It was not until the Morris worm outbreak in the 1980s that people realized the seriousness of this problem. This just shows that stack overflow vulnerabilities have always existed in programs but are subject to the limitations of people's cognitive understanding of stack overflows. Only when human cognition has developed to a certain stage over time and people can understand how stack overflows occur and how they cause security hazards can the stack overflow vulnerability be revealed.

Interdependence and Mutual Transformation of Absoluteness and Relativity

Relativity of transformation from bugs to vulnerabilities: In software and hardware engineering, people often talk about the concept of code bugs. Generally speaking, only security-related bugs are regarded as vulnerabilities. However, whether a bug is related to security depends on the running environment of the codes, and the judgment is quite subjective. In other words, only under certain conditions is a particular bug considered a vulnerability, but not under other conditions.

There are no vulnerabilities that are absolutely exploitable, and there are no vulnerabilities that cannot be exploited. Even if a vulnerability does exist, the exploitation of it can be made very difficult by technical means, which has become an important research direction in active defense today. Similarly, some vulnerabilities may not be well exploited at present, but this does not mean that they still cannot be exploited in the future when exploitation technologies improve to a higher level. Although some vulnerabilities cannot be exploited in a given running environment, they may be exploited in another running environment. Some vulnerabilities are unexploitable individually, but they may be exploitable when put together. For example, most memory corruption vulnerabilities are ineffective when encountering Address Space Layout Randomization (ASLR) protection mechanisms. However, if

combined with certain problems in chips, these vulnerabilities can be exploitable. For example, in early 2017, the VUSec team in Amsterdam [30] created a JavaScript program that can easily bypass the ASLR protection of 22 kinds of processors such as Intel, AMD, and NVIDIA. The attack demonstrated by VUSec exploits a vulnerability in the way the chip interacts with the memory. The chip has a component called the memory management unit (MMU), which is responsible for mapping the addresses of programs stored in the memory of the computer. To keep track of those addresses, the MMU constantly checks a directory called page table. Typically, the device will store the page table in the processor cache, so that the most frequently accessed information can be called into a small piece of memory by the computing kernel program at any time. However, a piece of malicious JavaScript codes running on the web page can also be written into that cache. Worse still, it can also check the working speed of MMU at the same time. By closely monitoring MMU, the JavaScript codes can find their own address. So they created a bypass attack program called ASLRCache (AnC), which can detect the position of the page table when MMU searches the page table. The most critical thing is that they can also check the working speed of the MMU at the same time. By closely monitoring the MMU, the JavaScript codes can find out their own address. So the VUSec team created a bypass attack program called “ASLR $\oplus$ Cache(AnC),” which can detect the position of the page table when MMU searches the page table.

The Unity of Opposites between Specificity and Generality

Vulnerabilities exist in specific environments, so it is unscientific to talk about vulnerabilities without specific environments. The presentation of each vulnerability requires a specific environment and specific conditions. If a remote overflow vulnerability, for example, is in an isolated environment, it will not have an impact on the system because there is no environment for its presentation. This is why a system that is safe in one environment may not be safe in another. A vulnerability type or pattern that originates from a specific system or software codes is a common problem associated with that type or pattern. Vulnerabilities are also general. For example, vulnerabilities in the same version of the Windows operating system must exist in all application systems where this version is installed, and similar or identical exploitable conditions exist in similar or identical operating environments. However, the existence and exploitability of a vulnerability are not always consistent. Differences in operating environments may cause the vulnerability to be unexploitable. In other words, the unexploitability of a vulnerability does not mean that the vulnerability does not exist. If a bug loses certain conditions, it may not become a vulnerability. If a vulnerability loses the environment on which it is reliant, it may lose its exploitability and may not be regarded as a vulnerability. This unity of opposites is an important basis for ensuring the effectiveness of endogenous defense mechanisms that do not rely on feature extraction.

1.3.3 Basic Problems of Vulnerabilities

Regarding vulnerabilities, worth studying are four basic problems: accurate definition, reasonable classification, unpredictability, and effective elimination.

1.3.3.1 The Problem of Accurate Definition of Vulnerabilities

With the evolution of computing paradigms, the development of information technologies, and the deployment of new applications, the connotation and extension of vulnerabilities are changing, making it difficult to define vulnerabilities accurately. Tesler [31] gave four computing paradigms ranging from the batch processing, time sharing, desktop system to networks during the period from the appearance of computers to 1991. Since the 1990s, new paradigms such as cloud computing, molecular computing, and quantum computing have emerged. Vulnerabilities vary under different circumstances such as single-user, multiuser, and multitenant conditions and in different computing environments such as distributed and centralized computing environments. Different systems have different security requirements, and the identification of vulnerabilities differs from system to system. What is a vulnerability in one system may not be a vulnerability in another. The same vulnerability is more harmful in some systems than in other systems. It is difficult to uniformly define the degree of harm caused by vulnerabilities. What is more, the harm level of the same vulnerability is not the same in different systems or different environments. In particular, the intervention of human factors may make a vulnerability become an uncertain disturbance. In the face of attackers with different resources and abilities, some software/hardware design defects may become vulnerabilities, while others may not. As time goes by, people's understanding of vulnerabilities is changing. From the initial definition based on access control to the present definition involving the whole process of system security modeling, system design, implementation, and internal control, with the development of information technologies, people will have a deeper understanding of and give a more accurate definition to vulnerabilities.

1.3.3.2 Reasonable Classification of Vulnerabilities

Linnaeus, a Swedish scientist, once said: "Through methodical classification and exact naming, we can distinguish objects when recognizing them. . . Classification and naming are the basis of science." Vulnerabilities widely exist in various information systems, and the number and types are increasing. To better understand and manage vulnerabilities, it is necessary to study the classification method of vulnerabilities. The classification of vulnerabilities refers to the division and storage of a large number of vulnerabilities according to their causes, manifestations, and consequences, so as to facilitate indexing, search, and use. Because the current

understanding of the nature of vulnerabilities is not comprehensive enough, it is difficult to reasonably, scientifically, exhaustively, and exclusively classify them. In early researches, vulnerability classification was mainly based on the need to eliminate coding errors in the operating system. Therefore, the classification basis focused more on the reasons for the formation of vulnerabilities. Today, it does have certain limitations and cannot fully reflect the nature of vulnerabilities. These classifications include Research into Secure Operating System (RISOS) classification [32], Protection Analysis (PA) classification [33], etc. Aslam [34] of the COAST Laboratory of Purdue University in the United States proposed an error classification method based on the cause for occurrence in UNIX systems. With the deepening of research, researchers noticed the concept of a vulnerability life cycle. At the same time, cyberattacks also brought serious threats to system security. Therefore, at this stage, researchers began to focus on the harm and impact of vulnerabilities and introduced this into the classification basis. Scholars involved in this classification research include Neumann [35], who proposed a vulnerability classification method based on risk sources; Cohen [36], who proposed an attack-oriented vulnerability classification method; and Krsul et al. [37], who proposed an impact-oriented vulnerability classification method. Gradually, researchers integrated their understanding of the relationship between vulnerabilities and information systems, the attributes of vulnerabilities, and the ways of vulnerability exploitation and repair into the basis of vulnerability classification, thus more accurately depicting the attributes of and correlation between vulnerabilities. Scholars involved in these studies include Landwehr [38], who proposed three-dimensional attribute classification and established three classification models according to the source, formation time, and location of vulnerabilities; Bishop [39] of the University of California, Davis, who proposed a six-dimensional classification method, divided vulnerabilities into different categories from six aspects, namely, cause, time, mode of exploitation, scope, number of exploited components, and code defects; Du et al. [40], who pointed out that the life cycle of vulnerabilities is defined as the process “from introduction, destruction to repair” and that vulnerabilities are classified according to the introduction reasons, direct effects, and repair methods; and Jiwnani et al. [41], who proposed a classification method based on cause, location, and influence. As vulnerabilities were becoming a widespread security issue, professional organizations emerged to manage the increasing industrial and social impacts of vulnerabilities, and vulnerability databases came into being. Vulnerability databases, as organizations that manage and disclose vulnerabilities, also set strict standards for the naming and classification of vulnerabilities. The US National Vulnerability Database (NVD) provides a list of common public vulnerabilities and exposures (CVEs). China National Vulnerability Database of Information Security (CNNVD), China National Vulnerability Database (CNVD), Open Source Vulnerability Database (OSVDB), BugTraq Vulnerability Database, and Secunia Vulnerability Database, as well as a large number of commercial vulnerability databases, have their own classification methods. China has also issued information security, technical security, and vulnerability-related specifications,

such as the national standard “Information Security Technology—Specification for Cybersecurity Vulnerability Management” (GB/T 30276-2020).

Unfortunately, none of these classification methods has been widely accepted so far. It should be said that theoretical research and practical analysis are still insufficient in revealing the characteristics, development laws, differences, internal relations of vulnerabilities, and the role of human factors in the formation and exploitation of vulnerabilities.

1.3.3.3 The Problem of Unpredictability of Vulnerabilities

The unpredictability of vulnerabilities can be summarized as a “4W” problem, that is, people do not know when, where, what, and by whom vulnerabilities will be discovered. Here, unknown vulnerabilities include unknown types of vulnerabilities and unknown vulnerabilities of known types. At present, human beings cannot predict new types of vulnerabilities, nor can they exhaust specific types of vulnerabilities.

Vulnerability types have evolved from the initial simple password type to buffer overflow, structured query language (SQL) injection, cross-site scripting (XSS), race condition to complex combined vulnerability types. Vulnerabilities have appeared in all kinds of software, components, and firmware, and even related vulnerabilities have emerged that exploit the cooperation between the system and firmware. The causes for and mechanisms of vulnerabilities are also becoming more and more complex. Some “geeks” pursue a high degree of self-identity and are committed to seeking new types of undiscovered vulnerabilities. They are no longer satisfied with pursuing known types of vulnerabilities. According to a report [42] released by Project Zero, a security team composed of top security personnel of Google, a total of 25 0-day vulnerabilities were exploited in wild attacks publicly disclosed in 2020, and the figure rose to 58 in 2021. Judging from the trend, there will be more and more attacks targeting browsers and mobile devices in the future. The rapid growth in the number of unknown software vulnerabilities exploited by hackers once again shows that the techniques of cybercrime and cyber espionage are becoming more and more advanced. Secret vulnerabilities in computer programs are of particular interest to criminal gangs, law enforcement agencies, and spies because software vendors will not release repair patches before receiving a warning. In 2021, Apple mentioned for the first time the situation of 0-day in the wild in its release notes. In 2021, there were five iOS 0-days in the wild in 2021, including the first publicly known macOS 0-day in the wild (CVE-2021-30869). Software vendors that develop and release patches either directly announce vulnerabilities or disclose related vulnerabilities when releasing patches.

1.3.3.4 The Problem of Effectively Eliminating Vulnerabilities

As software systems become increasingly complex, it is a consensus that software vulnerabilities will exist for a long time and are difficult to avoid. There are certainly many reasons for this situation, but generally speaking, it is related to the characteristics and traditional concepts of the software industry. The development of software is to match the hardware to achieve specific functions, so the realization of its functions is the first priority. Software security problems are considered only after the functions are realized. Moreover, due to the rapid update of software and fierce competition, it is far more important to launch available software to quickly occupy the market than to consider security problems.

People gradually realize that there will be bugs (errors) in the software implementation process, and these bugs will affect the stability of the program and the normal use of functions. As more and more systems and devices are connected to the network, people find that bugs may be strongly related to security and will affect the security of the software itself or even the system. In the early days, people were optimistic and tried to ensure the security of software through theorem proving, but in fact such a once-and-for-all thing could not be done. Then, people began to classify bugs or vulnerabilities, trying to study how to eliminate specific vulnerabilities. Disappointingly, analysis methods based on program analysis, such as abstract interpretation and static symbolic execution, have encountered bottlenecks, and static analysis techniques have encountered a series of problems such as low accuracy of interprocedural analysis and difficulty in point-to-point analysis. These problems are proved to be downtime problems in terms of computability. Since the 1980s, the model checking technology has been satisfactorily used to detect timing problems, yet it has given rise to the problem of state explosion, which is still an important obstacle to the development of the model checking technology. In the late 1990s, to meet the security needs of software development, the industry began to adopt a method called the fuzzing test to help find product vulnerabilities without waiting for the application of the above academic achievements. The fuzzing test is a method to trigger endogenous errors in a program by randomly constructing samples. As Moore's law continues to be effective and the design methods of software and hardware converge, the complexity of hardware systems increases abruptly. Like software systems, hardware systems also have design flaws and long-standing security vulnerabilities difficult to repair. For example, Meltdown and Spectre in Intel's CPU products in the past few years fall into this category of vulnerabilities. In fact, the technology and industry circles later discovered that most CPU products using the branch prediction pipeline technology had vulnerabilities of the same nature.

1.3.4 The Threats of Vulnerabilities and Backdoors

Since the 1990s, the Internet has not only achieved exponential growth but also rapidly penetrated and expanded into economic, social, and other fields. Especially in the past decade, the Internet of Things based on the concept of “Internet of Everything” or “deep integration of men, machines, things and networks” has greatly increased the number of networked devices in network information and communication infrastructure, and new applications such as industrial control system, artificial intelligence, cloud computing/cloud service, and mobile payment have surged. It is gratifying that the Internet is bringing huge wealth to the human society. According to Metcalfe’s law ($V = K \times N^2$), “the value of a network is equal to the square of the number of network nodes, or the value of a network is directly proportional to the square of the number of users connected to the network.”

However, what is disturbing is that the security risks in cyberspace are getting bigger and bigger, and various information security incidents are emerging one after another and getting worse, seriously affecting all aspects of human social activities and development. It can be said that cybersecurity threats have never been so close to everyone’s life as they are today. As an important part of the current network information technology, security vulnerabilities in the software and hardware of cyber-physical systems have become a decisive factor that directly affects system security. Practice has proven that most information security or cybersecurity incidents are initiated by attackers through software/hardware vulnerabilities. It is a mature mode of attack to exploit software and hardware vulnerabilities, develop backdoors based on vulnerabilities, or deliberately leave backdoors in design, with the aim of attacking and controlling targets. With the rapid improvement of attackers’ technical level, the harm of vulnerabilities and backdoors is increasing drastically.

1. The ransomware (WannaCry) swept the world. On May 12, 2017, a large-scale cyberattack hit 150 countries and infected 200,000 terminals. This attack involved a ransomware called WannaCry, which could spread quickly after infecting a computer. The files in the infected computer would be encrypted, and the user could only get the files decrypted after paying a certain amount of bitcoin (an untraceable online currency) as ransom. More than a dozen hospitals in the United Kingdom, as well as major companies such as FedEx and Telefonica, were targeted. Many universities in China were affected, and 20,000 PetroChina gas stations were disconnected from the Internet for nearly 2 days.
2. Sleepless nights for security personnel. On December 9, 2021, Apache officially released an emergency security update to fix the Apache Log4j2 remote code execution vulnerability. However, the updated Apache Log4j 2.15.0-rc1 version was found to still have vulnerabilities to bypass. Several security emergency response teams released a second vulnerability warning. At 2:00 am on December 10, Apache urgently released the Log4j-2.15.0-rc2 version to fix the vulnerability bypass caused by the incomplete remote code execution vulnerability repair of

the Apache Log4j-2.15.0-rc1 version. This vulnerability had a huge impact and was very simple to exploit. The attacker only needed to enter a piece of code to the target, and the vulnerability could be triggered without the user performing any operations, so that the attacker was able to remotely control the target server. More than 90% application platforms developed based on Java were affected!

3. The German government lost tens of millions of euros due to phishing attacks related with COVID-19. In April 2020, the government website of the state of North Rhine-Westphalia in western Germany suffered a phishing attack, which was roughly estimated to have caused a loss of at least 31.5 million euros. The hacker created a copy of the official website and used phishing emails to lure users to sign up so that he could collect detailed information of users. Then, the attacker made requests for government assistance on behalf of real users and replaced remittance bank accounts. According to foreign media reports, it was discovered that the hacker had forged a total of 3500–4000 fund requests, and the hacking operation lasted from mid-March to April 9, 2020. After the incident was discovered, the North Rhine-Westphalia state government immediately suspended payments to users and shut down its website.
4. Blockchain smart contract security. On April 19, 2020, Lendf.Me, a decentralized lending protocol, suffered a hacker attack, and \$25 million worth of assets in the contract were stolen. The direct cause was the new security risk introduced by the combination of the reentrancy problem of the product itself and the special ERC-777 token imBTC. New security risks were introduced after the imBTC combination. Also attacked because of this problem was Uniswap decentralized exchange.
5. Germany's main fuel suppliers suffered cyberattacks. On January 28, 2022, Oiltanking GmbH Group, a major German oil storage company, suffered a cyberattack. The attack affected the IT systems of Oiltanking and mineral oil trading company Mabanaft. Both companies are part of Hamburg-based Marquard & Bahls Group, one of the world's largest energy supply companies. On February 1, 2022, affected by the attack, the price of distillate diesel increased slightly in northwest Europe.

Compared with the past, cybersecurity incidents have shown the following four new trends:

1. Financial cybersecurity has caused widespread concern. In March 2021, AXA Group, an American insurance giant, spent US\$40 million to have its data decrypted. In August 2021, Tokyo Marine Holdings, Japan's largest property insurance company, was attacked by ransomware. In September 2021, the cryptocurrency exchange Liquid was hacked, resulting in the theft of at least US\$94 million worth of crypto assets. In April 2022, the financial system of the state Rio de Janeiro, Brazil, was attacked by LockBit ransomware, and 420 GB of data was stolen. This batch of data was stolen from the Sefaz RJ system, accounting for approximately 0.05% of all data stored in the state finance department. Since 2022, hackers have exploited LockBit, currently the most

popular ransomware-as-a-service platform, to attack at least 650 target organizations.

2. Security problems in emerging fields such as artificial intelligence and cloud computing have entered the public eye. In 2019, hackers stole more than 100 smart cars including brands such as Mercedes-Benz by invading smart car apps and rewriting programs and data. In March 2022, Denso Corporation of Japan, the world's second largest smart car manufacturer, suffered a hacker attack, resulting in the leakage of a total of 1.4 TB of data including the company's 157,000 purchase orders, emails, and design drawings. In June 2022, Pegasus Airlines, a Turkish low-cost airline, leaked about 6.5 TB of data due to a misconfiguration of the AWS S3 storage bucket. In October 2022, also due to a misconfiguration of the Azure Blob Storage bucket, 2.4 TB of Microsoft customer data belonging to more than 65,000 companies in more than 100 countries were leaked, causing serious social impacts.
3. Critical infrastructure has become a new target for hacker attacks. In April 2021, Israel launched destructive cyberattacks against Iranian nuclear facilities, causing a power outage to the facilities. In March 2022, German wind turbine manufacturer Enercon suffered a cyberattack, which caused the interruption of satellite communication in Europe and directly affected the monitoring and control of nearly 6000 wind turbines with a total installed capacity of 11 GW in Central and Eastern Europe. Looking back on the major cybersecurity incidents that have occurred in recent years, hackers are not only focusing on stealing various core data but also targeting some critical infrastructure. Governments, financial institutions, and the energy industry have all become new targets for hacker attacks.
4. Hacking with a political purpose is increasing. In April 2019, the Russian hacker group "Fancy Bear" launched cyberattacks by distributing malicious documents to interfere with the Ukrainian general election. In February 2020 when the COVID-19 pandemic was rampant in China, Indian hacker organizations used pandemic-related themes as bait documents to launch APT attacks on Chinese medical institutions. It can be seen from the many cybersecurity incidents that have occurred in recent years that there are more and more political hacker actions supported by states. Future cybersecurity will definitely affect the stability of a country. It is a foregone conclusion that cybersecurity will rise to the height of national security and depend on how each country responds.

All these information and cybersecurity incidents have one thing in common, that is, information systems and software/hardware themselves have ESS problems based on vulnerabilities and backdoors, which makes cyber-physical systems or digital facilities inevitably bring harm to national economy, national security, and social stability.

1.3.4.1 Security Threats Caused by the Widespread Existence of Vulnerabilities

Theoretically, all information systems or devices have vulnerabilities in design, implementation or configuration, and vulnerabilities are ubiquitous. The Internet of Everything has become a major trend today and even in the future. More and more individuals and things will be connected to the same network system, including not only smart wearables, smart homes, and other items used in life but also commercial circulation, energy, transportation, social undertakings, urban management, and other fields in the future. Strategy Analytics, the world's leading data analysis and business consulting company, predicts [43] that there will be 38.6 billion and 50 billion networked devices by 2025 and 2030, respectively. The huge number of networked devices has become and will continue to be the physical basis for widespread vulnerabilities.

The trend of “software definition” is getting stronger and stronger in the IT industry. From software-defined functions, software-defined computing (SDC), software-defined hardware (SDH), software-defined network (SDN), software-defined architecture (SDA), software-defined storage (SDS), software-defined interconnection (SDI), software-defined data center (SDDC) to software-defined infrastructure (SDI), all kinds of products and technologies are labeled as “software defined.” Some people even claim that “software defines everything (SDX),” “software defines the world,” and “everything is software defined.” It seems that “software defined” has become synonymous with the most advanced technology. In the future, more and more computing, storage, transmission, exchange, and even the entire IT infrastructure will become software defined. With the popularization of the term of “software defined,” vulnerabilities and backdoors in software will inevitably spread to related application fields, forming various potential vulnerabilities. “Software definition” has become the “amplifier” of the software's own ESS problem, which has brought a wider security threat or destruction to some extent. The term “software defined” has become an “amplifier” of the software's own ESS problems and has brought a wider security threat or destruction to some extent.

Compared with ESS problems in software, security threats caused by ESS problems in hardware have a wider impact and cause more serious damage. In recent years, the frequent release of hardware vulnerabilities has proved that there are a large number of vulnerabilities at the hardware level such as chips, devices, and systems. Among them, chip microarchitecture and hardware security protection mechanisms have become the “hardest hit area” of ESS problems in hardware. In addition, as the original closed systems such as industrial control systems become digital, networked, and intelligent, their ESS problems are gradually exposed, and a large number of software and hardware vulnerabilities pose a serious threat to the security of industrial control systems.

Threats Caused by Vulnerabilities in Hardware

A hardware vulnerability is a specific combination of status elements and control elements of a hardware unit in a hardware system. This combination may cause the hardware unit to violate the established working mechanism at a certain point in time, in a certain working state, and under specific environmental conditions, which may lead to some behavior in the system that violates the established information security objectives and security protection mechanisms, and even lead to partial or overall functional failures, or endanger the security of key data [44].

There are many reasons for the existence of hardware vulnerabilities. First, the hardware units or mechanisms added for the purpose of improving performance compromise the original stable state, resulting in uncertainties in hardware, such as chip microarchitecture vulnerabilities. Second, there are vulnerabilities in the hardware security mechanism introduced for security purposes, which leads to more serious security threats, such as RSA encryption vulnerabilities in trusted platform module (TPM) chips. Third, due to the complexity of production processes, third-party IP or protocol specifications will inevitably be used in hardware design, and these modules or protocol specifications may have vulnerabilities that cannot be covered by conventional tests, such as the vulnerabilities in Bluetooth communication modules in microcontrollers. Fourth, observed from a cross-domain perspective, the emergence of hardware vulnerabilities is inevitable. Changing the working conditions of hardware may lead to exploitable vulnerabilities in the originally safe hardware, such as inducing authentication bypass vulnerabilities in Starlink terminals through voltage fault injection attacks. The following are examples of these three types of vulnerabilities.

1. Vulnerabilities in the Hardware Security Mechanisms

The TPM (trusted platform module) chips provide a root of trust for the computer, aiming at protecting sensitive information from infringement when system security is threatened. TPM can be divided into two types: dTPM (independent hardware TPM) and fTPM (firmware-emulated TPM). Both types have been reported to have security vulnerabilities.

In 2017, security researchers at Masaryk University in the Czech Republic discovered a high-risk vulnerability ROCA (CVE-2017-15361) in the TPM made by Infineon, a German company. This RSA encryption vulnerability allows attackers to launch factorization attacks while reversely calculating the private RSA encryption key through the public key of the target system. This attack affected all encrypted smart cards, security tokens, and other security hardware chips produced by the company before 2012, putting billions of devices in danger.

In 2019, researchers found that the elliptic curve signature operation on TPM is susceptible to timing leakage, which means that the private key can be obtained by measuring the execution time of the internal operation of the TPM device.

Both Intel processor's fTPM (CVE-2019-11090) and STMicroelectronics' TPM chip (CVE-2019-16863) have similar vulnerabilities, and the correspondingly designed TPM-Fail attack [45] uses timing-side channel attacks to recover the key.

If an attacker directly operates the device, it will take about 4–20 min to obtain the ECDSA key, depending on the access level. If the attacker launches a remote attack, he will be able to obtain the key by performing only 45,000 authentication handshakes at regular intervals through the network. Once the key is leaked, the attacker can forge a digital signature, steal or change encrypted information, bypass the security function of the operating system, or destroy the applications that depend on the integrity of the key.

The discovery of this vulnerability exposed the computer products of many PC manufacturers such as Lenovo, Dell, and HP that use Intel TPM to the threat of attacks. It is worth noting that these vulnerabilities also exist in devices that have passed the certification of FIPS 140-2 Level 2 and Common Criteria (CC) EAL 4+, the latter being the highest internationally recognized security level in CC certification. This means that there are weaknesses in the above certifications, no product is absolutely safe, and no test can detect all problems.

In June 2022, researchers at the MIT Computer Science and Artificial Intelligence Laboratory discovered a vulnerability in the hardware security mechanism used by Apple's M1 series chips (pointer authentication code, PAC) and used this vulnerability to design a PACMAN attack [46]. The PACMAN attack guessed the PAC value and used side-channel information to judge whether the guess is correct. The PACMAN attack could further steal data through pointer authentication and gain full access to the operating system's kernel. Since PAC has been added to the ARM v8.3-A specification, it may affect the security of a large number of ARM SoCs.

2. Third-Party IPs or Protocol Specifications Introduce Vulnerabilities

Third-party IPs or protocol specifications have also led to the widespread existence of chip vulnerabilities, such as the BrakTooth Bluetooth vulnerability [47]. In 2021, the Automated Systems Security Engineering Team (ASSET) of the Singapore University of Technologies and Design disclosed a new set of security vulnerabilities in the commercial Bluetooth protocol and released a proof-of-concept attack program. The BrakTooth vulnerability contains a series of 16 security vulnerabilities, involving 13 Bluetooth chipsets from 11 chip manufacturers including Espressif Systems, Intel, Qualcomm, Texas Instruments, Infineon, Samsung, and MediaTek, covering 1400 commercial products including laptops, smartphones, programmable logic controllers, and IoT devices. Among them, there is a high-risk vulnerability CVE-2021-28139 in Espressif's ESP32 SoC chip. This vulnerability allows attackers to execute arbitrary code, which is extremely risky.

3. Cross-Domain Active Attacks Induce Vulnerabilities

Some forms of physical domain attacks, such as hardware fault injection, can make systems or devices that seem to be safe in the information domain also present exploitable vulnerabilities or backdoors. At the Black Hat annual security conference in November 2022, Wouters of the University of Leuven in Belgium released for the first time a voltage fault injection attack method targeting "Star Link" terminals [48]. The Star Link Internet system consists of Star Link satellites, gateways, and terminals. Star Link terminals are phased array dish-shaped satellite

antennas (Dishy McFlatface antennas) installed on roof of user homes and buildings. Wouters connected an embedded development board named Modchip to the PCB board of the Star Link terminal and used the Modchip circuit board to launch a voltage fault injection attack during the startup and boot phase of the Star Link terminal. Active attacks cause Star Link terminals to be temporarily short-circuited to bypass Star Link's firmware signature verification mechanism. After that, the attacker can successfully update the firmware and run custom codes on the terminal. After the vulnerability was released, Space X updated its software, and the updated version made the attack much more difficult. However, because the vulnerability exists in hardware, the security problem cannot be fundamentally solved. At present, Star Link has more than 500,000 users in 36 countries and regions. Due to the global popularity of Star Link, hacker attacks caused by this vulnerability may have a severe impact on international security.

In addition to the above three types of vulnerabilities, hardware vulnerabilities also include chip microarchitecture vulnerabilities, which are currently the most influential type of hardware vulnerabilities, and most CPU vulnerabilities belong to this type.

The Recent Frequent Occurrence of CPU Vulnerabilities Has Become a Hot Topic of Discussion

After Intel was reported to have CPU vulnerabilities that affected almost all computers around the world, related research attracted widespread attention in the cybersecurity field. Since the first CPU vulnerability appeared in 1994, the harm of CPU vulnerabilities has evolved from the initial denial of services to being used to steal information. The degree of harm has continued to worsen, and the skills of exploiting the vulnerability have gradually increased. The first CPU vulnerability "Pentium FDIV bug" [49] appeared in the Intel Pentium processor and was discovered by Professor Thomas of Lynchburg University in 1994. The reason for this vulnerability is that Intel burned the entire multiplication table inside the processor to improve the operation speed. However, among the 2048 multiplication numbers, there were 5 burning errors, so there would be errors in the operation of special numbers. The subsequent "CPU F00F vulnerability" [50] also appeared in Intel processors, affecting all CPUs based on the P5 microarchitecture. Later, AMD processors were reported to have a TLB vulnerability. In the affected Phenom processors, the TLB vulnerability would cause the CPU to read the page table incorrectly, resulting in a denial of services such as a crash. In 2017, the ME (management engine) vulnerability [51] appeared in Intel processors, which represented a turning point in the evolution of CPU vulnerabilities. ME is a low-power subsystem built into the CPU by Intel, which can assist professionals in remotely managing computers. It was originally designed for remote maintenance, but because of the vulnerability, attackers could manage to control computers through the backdoor of ME.

Typical CPU vulnerabilities with high risks are the Meltdown [52] and Spectre [53], which were jointly discovered by the Google Project Zero team, Cyberus Technology Company, and a number of foreign universities in January 2018. All Intel processors after 1995 were affected. The Meltdown vulnerability “melts” the hardware isolation boundary between the user mode and the operating system kernel mode. An attacker can use this vulnerability to break through the restriction of system permissions from the low-privileged user mode and “cross the boundary” to read the memory information of the system kernel, resulting in a data breach. Similar to the Meltdown vulnerability, the Spectre vulnerability breaks the isolation between different applications. Attackers use the CPU’s predictive execution mechanism to attack the system, control a certain variable or register of the target program through malicious programs, and steal private data that should be isolated. The Financial Times reported on August 15, 2018, that Intel disclosed the latest vulnerability in its chips, L1 Terminal Fault L1TF, nicknamed “Foreshadow,” which may allow hackers to obtain memory data. The vulnerability was discovered by teams from the University of Leuven in Belgium, the University of Michigan in the United States, and the University of Adelaide in Australia. The Computer Emergency Readiness Team (CERT) of the US government warned that attackers may use this vulnerability to obtain sensitive information including keys and passwords. According to experts, compared with ordinary vulnerabilities, the “Foreshadow” vulnerability is more difficult to exploit. On December 11, 2019, Intel officially confirmed and released the “VultJockey” vulnerability discovered by Wang Dongsheng, Lv Yongqiang, Qiu Pengfei of Tsinghua University, and Gang Qu of the University of Maryland. This vulnerability would affect Intel’s sixth, seventh, eighth, ninth, and tenth generations of Core™ processors, as well as Xeon E3v5&v6 and E-2100&E-2200 processors. This vulnerability is because the dynamic power management module DVFS used in the design of modern mainstream processor microarchitecture has security risks. This vulnerability can be exploited to obtain the key from the trusted zones SGX and TrustZone of Intel or ARM’s CPU. Without the need to use any special hardware technology, the key can be directly obtained through online attacks by pure software methods. In 2020, Intel released an announcement to confirm the CacheOut vulnerability whose number is CVE-2020-0549. Intel named this vulnerability L1D Eviction Sampling. On some processors of certain microarchitectures, recently cleared modified L1D cache lines may propagate to unused (invalid) L1D fill buffers. On processors affected by MDS (microarchitecture data sampling) or TAA (transactional asynchronous abort), one of these side-channel methods can be used to infer the data from the L1D fill buffer. Combining these two vulnerabilities makes it possible for an attacker to infer data values from modified cache lines. These vulnerabilities affect all CPUs released by Intel from 2015 to 2019. A 2022 study by researchers from ETH Zurich found that certain microprocessors from Intel and AMD are vulnerable to new speculative execution attacks related to Spectre Variant 2. The attacks could be used to leak data from the kernel memory, and the researchers named the vulnerability “Retbleed.” The attacks exploit vulnerabilities in Retpoline, a measure introduced in 2018 to mitigate certain speculative execution attacks. The mitigation measure

may incur an additional cost and affect the performance of the patched system. This vulnerability affects Intel Core sixth- to eight-generation processors, as well as AMD Zen 1, Zen 1+, and Zen 2 processors. In August 2022, researchers at the University of Rome I discovered a new vulnerability named $\mathcal{A}$ PIC in Intel CPU, which enabled attackers to obtain encryption keys and other confidential information from the processor. $\mathcal{A}$ PIC (CVE-2022-21233) is the first architectural CPU bug that could lead to the leakage of sensitive data and affect most of the 10th-, 11th-, and 12th-generation Intel CPUs. Table 1.2 lists the information about typical processor vulnerabilities in recent years. The causes for CPU vulnerabilities include design logic problems and specific implementation problems. Most of them are design logic problems. For example, Meltdown, Spectre, VoltJockey, and ME vulnerabilities are all caused by insufficient consideration of possible security risks when designing CPU functions, resulting in problems such as invalidation of privilege isolation, unauthorized access, etc. Specific implementation problems are caused by security risks that manufacturers have in the implementation details of the CPU, such as FDIV vulnerability, F00F vulnerability, etc.

A large number of vulnerabilities exist not only in general-purpose microprocessors but also in embedded microprocessors. In May 2022, a joint research team from the University of Illinois at Urbana-Champaign, Tel Aviv University, and the University of Washington discovered a vulnerability, Augury [54], in the data memory-dependent prefetcher (DMP) in Apple's M1 and A14 chips and proved that DMP microarchitecture vulnerabilities could leak static data. Unlike Spectre and Meltdown vulnerabilities that leak data in use, Apple's Augury vulnerability could leak the data of the entire memory, even if the data have not been actively accessed. Not only do Apple's M1, M1 Max, and A14 use DMP for prefetching, but M1 Pro and older A- series chips are also vulnerable to the Augury vulnerability.

Since the CPU is at the bottom core of the computer, its vulnerabilities are highly concealed and harmful and have a wide range of damage.

Highly Concealed The CPU is the lowest computing execution terminal, and its internal structure is transparent to the upper layers. Therefore, the vulnerabilities in it are highly concealed, and it is difficult for people who are not engaged in CPU security to catch such vulnerabilities.

Highly Harmful Because the operating system is built on the CPU, attackers can gain higher privileges by exploiting CPU vulnerabilities than by exploiting operating system vulnerabilities. Especially in the era Internet "cloud," CPU vulnerabilities can be used to achieve more harmful operations such as virtual machine escape and virtual machine isolation breakthrough.

Wide Range of Damage Since CPU is the basic computing component, computing devices such as IoT devices, PCs, servers, embedded devices, etc. all need CPU components. Usually, a CPU vulnerability will endanger a large number of devices, resulting in cross-industry and cross-domain damage.

Table 1.2 Typical vulnerabilities in processors

Manufacturer	Vulnerability name or number	Vulnerability type	Vulnerability details	Affected products
Intel	CVE-2012-0217	Local privilege escalation	There is a vulnerability in the sysret instruction	Intel processors produced before 2012
	Memory sinkhole	Local privilege escalation	Rootkits can be installed in the processor's system management mode	Intel x86 processors produced between 1997 and 2010
Broadcom	CVE-2017-6975	Code execution	An attacker in the same Wi-Fi network could exploit this vulnerability to remotely execute malicious code on the Broadcom Wi-Fi chip (SoC) in the device	iPhones 5–7, Google's Nexus 5 and 6/6P, and Samsung's Galaxy S7, S7 Edge, S6 Edge, and other devices
Intel	CVE-2017-5689	Privilege escalation	The attacker can remotely load and execute any program, read and write files	Intel-managed firmware versions include 6.x, 7.x, 8.x, 9.x, 10.x, 11.0, 11.5, and 11.6
Intel	CVE-2017-5754	Unauthorized access	Low-privilege users can access the contents of the kernel and obtain underlying information of the local operating system	Intel processors after 1995 except Itanium and Atom before 2013
Intel/AMD	CVE-2017-5753/CVE-2017-5715	Information leakage	In cloud service scenarios, Spectre can be used to break through the isolation between users and steal other users' data	Intel processors after 1995 and chip products from AMD, ARM, and Nvidia, except Itanium and Atom before 2013
Intel	TlBleed	Information leakage	An attacker can be allowed to eavesdrop on TLB information and then restore the key	CPUs of Skylake, Coffee Lake, Broadwell, etc.
ARM/Intel	Spectre-NG	Code execution information leakage	An attacker is allowed to access and use a virtual machine (VM) to execute malicious code, read the host's data, steal sensitive data such as passwords and digital keys, or even completely take over the host system	Affected models not accurately disclosed

Explosive Growth of Vulnerabilities in Industrial Control Systems

On September 1, 2021, the National Regulations on the Security Protection of Critical Information Infrastructure (hereinafter referred to as the Regulations) came into effect. The promulgation of the “Regulations” marks the further improvement of the legal system for the security of the critical information infrastructure, which has been regarded as an important strategic asset of the country [55]. However, the cybersecurity situation of the critical information infrastructure is becoming increasingly severe, and the threat of cyberattacks is rising.

Attacks on industrial control systems have been exposed in recent years. The most representative one is the outbreak of the Stuxnet virus in 2010. Stuxnet is the first worm virus targeting industrial control systems. It infects the supervisory control and data acquisition (SCADA) system by taking advantage of the vulnerabilities in Siemens control system SIMATIC WinCC/Step7. The virus targets the equipment of the Bushehr nuclear power plant in Iran. Stuxnet simultaneously exploited seven latest vulnerabilities in Microsoft and Siemens products to launch attacks, ultimately causing the delay in the startup of Iran’s Bushehr nuclear power plant. A similar incident was that in December 2016 hackers used the Industroyer virus to attack Ukraine’s power grid control system, causing a power outage in Kiev, the capital of Ukraine, for more than an hour. Millions of households lost power supply, and power facilities were severely damaged. The attacker used the vulnerability CVE-2015-5374 in Siemens SIPROTEC equipment to cause a denial of services on the target equipment and prevent it from responding to requests. In March 2019, a large-scale power outage occurred in Venezuela. About 20 of the country’s 23 states, including the capital Caracas, experienced power supply interruptions. National production and people’s life were paralyzed, leaving the country on the verge of collapse. The cause for the accident is unknown yet, but the President of Venezuela publicly accused the United States of using high-tech weapons to attack Venezuela’s energy supply system. In May 2021, the service systems of European energy technology suppliers were forced to shut down. A Norwegian manufacturer that provides technical solutions to European energy and infrastructure companies encountered a ransomware attack and was forced to shut down all infrastructure applications provided for 200 municipal water treatment facilities in Norway, affecting about 85% of the country’s residents. To prevent the ransomware from spreading to other computer systems, the company had to shut down various other applications it hosted and isolate all the devices used by about 200 employees.

On May 9, 2021, Colonial, the largest fuel transportation pipeline company in the United States, was forced to suspend its fuel transportation business due to a ransomware attack. This led to the US government declaring a state of national emergency for the first time due to a cyberattack. Colonial had to pay a huge ransom to the attacker, and it did not resume operations until May 13.

A growing number of industrial control system vulnerabilities are emerging, affecting more and more industries, and their danger and types are increasing as well. As of February 2020, CNVD had collected more than 2340 vulnerabilities in industrial control systems, most of which originated from communication protocols,

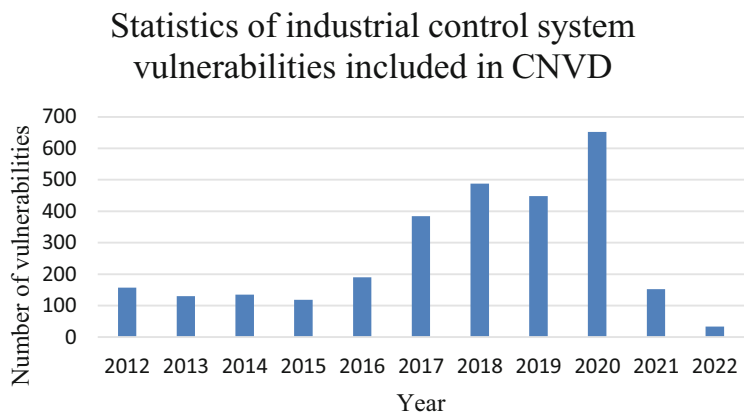


Fig. 1.11 Statistics of industrial control system vulnerabilities between 2010 and 2022

operating systems, application software, and field control layer equipment, including many common vulnerabilities such as buffer overflow, hard-coded credentials, authentication bypass, cross-site scripting, and so on. Figure 1.11 shows the statistics of industrial control system vulnerabilities collected by the platform between 2012 and 2022. Since the outbreak of Stuxnet in 2010, vulnerabilities in industrial control systems have shown a trend of explosive growth. The number of vulnerabilities included in CNVD has increased sharply from 32 in 2010 to 652 in 2020. Since 2021, the number of vulnerabilities recorded by CNVD has dropped significantly. There are two possible reasons for this. First, due to the repeated outbreaks of the COVID-19 pandemic around the world, a large number of employees work from home, and the vitality of the industrial control industry is low, resulting in a decrease in the number and types of industrial control targets, hence a decrease in the occurrence and discovery of industrial control system vulnerabilities. Second, with the continuous improvement of industrial control information security policies, systems, and regulations, the product systems and solutions for industrial control security have become more and more mature. In such a context, the decline in the number of vulnerabilities should be reasonable.

1.3.4.2 Excessive Homogeneity Brings Ecological Crisis

Isomorphism Leads to Environmental Similarity

Although many years of research have warned that software monoculture or homogeneity will cause great security risks, most information systems deployed today still adopt a relatively static, fixed architecture, and roughly similar operating mechanism. This makes an attack that works on one system easily and quickly applicable to all systems with the same environment or similar configuration.

The Turing-von Neumann computable architecture or equivalent architecture is the absolutely dominant processing architecture in cyberspace. The laws of market economy and monopolistic behavior have caused a lack of variety in cyberspace, and the homogenization of the cyberspace technology and system architecture is serious. The static, deterministic, transparent, and similar nature of the digital infrastructure architecture and operating mechanisms has become the biggest “security black hole” because once the existing “vulnerabilities or backdoors” are exploited by attackers, they will cause continuous security problems. In a globalized or multipolarized environment, the cyberspace ecological environment is very fragile, and cyber-physical systems are particularly sensitive to unknown technical flaws, vulnerabilities, and backdoors. The existing defense system based on accurate perception cannot cope with the threat of “unknown vulnerabilities” once it lacks prior knowledge. Even the blockchain technology based on the “consensus mechanism” of a large-scale user base may not have taken into account the fact that Wintel Alliance occupies more than 80% of the desktop terminal and operating system market and that Google’s Android system occupies more than 70% of the mobile terminal market. Once these software and hardware vulnerabilities and backdoors are deliberately exploited, coupled with the global coverage of timing and positioning services, it is not an insurmountable barrier to break through the “51% consensus threshold.”

Reuse Leads to Genetic Similarity

The reuse of codes includes at least three ways: integrating third-party libraries, borrowing from open-source code, and inheriting your own historical code.

Heartbleed, a program error that appeared in OpenSSL, an encryption library, was first disclosed in April 2014. Since the library is widely used to realize the Transport Layer Security (TLS) protocol of the Internet, China’s large websites including Alibaba, Tencent, and Baidu were affected by Heartbleed. Many domestic browsers, such as 360, Maxthon, Window of the World, UC, Sogou, etc., use open-source kernels, so once vulnerabilities are discovered in the open-source kernel, domestic browsers will also face security risks. See Table 1.3 for specific statistical information.

Since the operating system code focuses on backward compatibility, a piece of vulnerable code may involve or affect a wider range of versions for a long time. This is the case with the “old-age vulnerabilities” in the Windows operating system and the “shell-breaking vulnerabilities” in the Linux operating system.

Old-Age Vulnerabilities There has been a local privilege escalation vulnerability in the Windows operating system since 1992, which allows hackers to gain the highest control of the system, thus easily destroying and disabling any security software, including antivirus software, firewall, active defense software, sandbox, and system recovery. It can also be used to bypass the user account control (UAC) of Windows Vista/Windows 7 or to enhance the operation authority on the server

Table 1.3 Browsers using open-source kernels

Domestic browsers	Open-source kernels
Cheetah safe browser	Trident+Webkit for Versions 1.0~4.2; Trident+Blinkforversion
360 secure browser	Trident for Versions 1.0~5.0; Trident+Webkit for Version 6.0; Trident +Blink for Version 7.0
Maxthon browser	IE kernel for Maxthon1.x and 2.x; IE+Webkit for Maxthon3.x
Window of the World browser	IE kernel at first; Chrome+IE after 2013
Sogou high-speed browser	Trident for Version1.x; Trident+Webkit for Version 2.0 and later versions
UC browser	Blink kernel+ Trident kernel

website and control the whole network server, which directly threatens the information security of the government, enterprises, Internet cafes, and personal computer users. This vulnerability has been lurking for 18 years, affecting all versions of Windows operating systems, including Windows NT 4.0, Windows 2000, Windows XP, Windows 2003, Windows Vista, Windows 7, and Windows Server 2008.

Shell-Breaking Vulnerabilities On September 24, 2014, Bash was reported to have a remote code execution vulnerability. This vulnerability may lie dormant for more than 10 years, affecting the current mainstream Linux and Mac OS X operating systems, including but not limited to Redhat, CentOS, Ubuntu, Debian, Fedora, Amazon Linux, and OS X 10.10. This vulnerability can execute the attack code script that the hacker wants to execute by constructing the value of the environment variable. The vulnerability may affect a variety of applications that interact with Bash, including Hypertext Transfer Protocol (HTTP), OpenSSH, and Dynamic Host Configuration Protocol (DHCP). It may seriously affect the security of network infrastructure, including but not limited to network equipment, cybersecurity equipment, cloud and big data centers, etc. Since Bash exists in a wide range of devices, the elimination process of the vulnerability has a “long-tail effect,” and it is easy to write worm code for automatic spread, which will also promote the development of botnets.

Three security vulnerabilities were discovered in the Linux kernel SCSI (small computer system interface) component, namely, CVE-2021-27363, CVE-2021-27364, and CVE-2021-27365. The first vulnerability, CVE-2021-27365, is a heap buffer overflow vulnerability in the iSCSI subsystem. This vulnerability can be triggered by setting the iSCSI string attribute to a value greater than 1 page and then reading the value. The second vulnerability, CVE-2021-27363, is also a heap overflow vulnerability. Researchers found that kernel pointer leakage can be used to determine the address of the iscsi_transport structure. This vulnerability is slightly less impactful and can be exploited for potential information leakage. The third vulnerability, CVE-2021-27364, is an out-of-bounds kernel reading vulnerability in the libiscsi module (drivers/scsi/libiscsi.c). Similar to the case for the first vulnerability, an unprivileged user can trigger a controlled out-of-bounds reading operation

by constructing a net link message that specifies the cache size. Several user-controlled values are unverified, including the calculation of the size of the previous header. This vulnerability can cause data leakage and can trigger DoS conditions after being exploited. These vulnerabilities have existed in the system since 2006 and allow attackers to escalate from basic user privilege to root privilege. In addition, attackers can use these vulnerabilities to bypass security features such as kernel address space layout randomization (KASLR), supervisor mode execution protection (SMEP), supervisor mode access prevention (SMAP), and kernel page-table isolation (KPTI). These vulnerabilities affect all Linux distributions loaded by the `scsi_transport_iscsi` kernel module.

1.3.5 Examples of ESS Problems

To better understand the concept and connotation of CESS, this section lists the individual, common, and host-dependent ESS problems existing in current “hot” technologies such as big data, artificial intelligence, blockchain, zero-trust architecture, and digital encryption and authentication.

1. The big data technology can discover unknown patterns or characteristics according to algorithms and unsampled datasets, and deliberate contamination of data samples and malicious triggering of algorithm design flaws can also lead people astray [56]. The noninterpretability (or black-box effect) of results is an individual ESS problem of the big data technology, while the software/hardware vulnerabilities and backdoors in its algorithm implementation or host system are unavoidable common ESS problems.
2. The current artificial intelligence technology based on neural network algorithms relies on big data, large computing power, and deep learning algorithms to gain momentum, and the unexplainability, unpredictability, and uninference of the results are the characteristics of the current mainstream AI technology. There are also ESS problems in algorithm implementation or host systems [57].
3. The blockchain technology has opened up a new era of decentralized book-keeping, but its consensus mechanism cannot avoid the same vulnerability and backdoor in the same software/hardware products with a market share of more than 51%. For example, if more than 51% of accounting nodes use x86CPU, Windows, Linux operating system, or Oracle database, once the unknown vulnerabilities and backdoors in these software and hardware products are deliberately exploited, then more than 51% of the consensus mechanism will be unable to effectively protect data assets. The Blockchain 1.0 technology does not take into account the homogeneity of node software and hardware environments, so it has both individual and common ESS problems.
4. Based on the zero-trust security architecture of distributed step-by-step authentication system, any access subject (person/device/application, etc.) must undergo identity authentication and authorization before access is allowed to

avoid excessive trust. The right of the access subject to access resources is dynamic (not static) and is logically open and secure. However, in practice, if there are exploitable software/hardware vulnerabilities in the related authentication nodes or host systems [58], there will also be host-dependent common ESS problems.

5. Branch prediction in the computer architecture is a data processing optimization method to solve the problem of pipeline failure caused by CPU processing branch instructions (if-then-else). However, Spectre is a side effect or dark function of “predictive execution,” which reduces memory delay and speeds up execution and may cause an information leakage event in which sensitive data saved by the target process are identified [53]. This is a typical individual ESS problem.
6. The cloud computing/data center technology has changed the way of providing information or data services and improved the efficiency of resource utilization. However, people are extremely worried about the security of “putting eggs in one basket” because of cyberattacks that lead to sensitive data leakage, data integrity defects, service function interruption, and service performance deterioration [59]. This is a typical host-dependent ESS problem.
7. Even if the digital encryption or authentication algorithm is powerful enough in a mathematical sense, it still may lead to disastrous consequences if there are exploitable vulnerabilities or backdoors in the software/hardware execution environment where the encryption and authentication algorithm is executed [60]. Therefore, encryption authentication is also a typical host-dependent ESS problem. Therefore, encryption authentication is also a typical host-dependent ESS problem.
8. All the existing add-on or built-in cybersecurity software functions are realized through code design. While providing additional security defense functions for the target object, it is difficult for them to avoid introducing new vulnerabilities to the target system due to their own design flaws. This is caused by common ESS problems.
9. In the 5G service network based on the cloud platform/edge computing technology, cyberattacks are likely to cause information leakage, data resources being controlled, or system services being “down” because there are some common ESS problems such as unknown design defects, backdoors, and trapdoors in the software/hardware of cloud and edge computing system [61].
10. The Turing machine only answers the question of what is computable at the theoretical level, and the von Neumann computing structure only solves the problem of under what conditions computable problems can be realized in engineering, but computers cannot distinguish between benevolent and malicious computing and processing.

The above examples all belong to the category of the “endogenously insecure technology” in cyberspace. Although some of them have individual characteristics (such as big data and artificial intelligence), as mentioned above, all technical architectures or algorithms in cyberspace are based on software and hardware, so

there will inevitably be multiple or mixed presentations of individual, common, and host-dependent ESS problems.

1.4 Common Problems of Cyberspace Endogenous Security and Safety

1.4.1 *Definition of Common Problems*

Just like the definition of CESS problems, the cyberspace is essentially composed of cyber-physical systems that integrate computing, networking, sensing, and physical environment, while the “basic building materials” of cyber-physical systems are composed of entities expressed by various software and hardware codes. According to the theory of contradiction, there are contradictions in everything in cyberspace, and defects or vulnerabilities in human-designed and manufactured software and hardware are no exception. Philosophically speaking, common (generality, ubiquitous) characteristics are certain characteristics that are general or ubiquitous in a certain field or industry, which are internal rather than external, universal rather than special, and collective rather than individual. Therefore, we call the problems due to software/hardware design defects and artificial or nonartificial disturbances in the endogenous security and safety problem domain “common problems of cyberspace endogenous security and safety (common CESS problems),” and their problem domain and solution domain are shown in Fig. 1.12. It should be emphasized that there is an overlap between individual problems and common problems, and the overlapping area is expanding with the ubiquitous application of digital, network, and intelligent technologies. For example, artificial intelligence based on neural

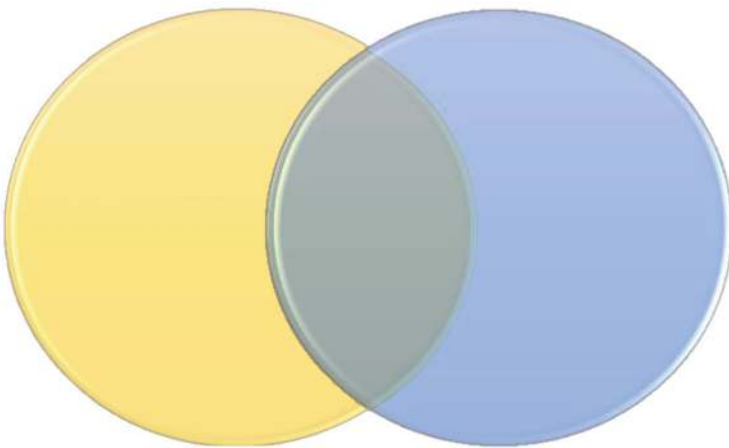


Fig. 1.12 Common ESS problem domain and solution domain

network algorithms has individual endogenous problems such as inexplicability, unpredictability, and noninferability of results. Because the host or cyber-physical system in the process of algorithm implementation does not have the attribute of “trusted execution environment (TEE),” it cannot prevent the influence of common problems such as software and hardware vulnerabilities and backdoors. In addition to individual problems such as nuclear leakage and nuclear pollution, the control devices or systems of nuclear power plants cannot avoid the network security threats of vulnerabilities, backdoors, and related human disturbances. In addition to mechanical failure, the complex control system composed of thousands of lines of software/hardware codes of intelligent networked vehicles cannot prevent cyberattacks based on vulnerabilities and backdoors.

1.4.2 Causes for Common Problems

1.4.2.1 Limitations of Cognitive Ability

Due to the limitation of the technological and human cognitive level, the vulnerability of software/hardware code design exists against people’s will, so there is no possibility of completely avoiding vulnerabilities. For example, the inventor of the computer branch prediction method certainly could not have imagined that this classic technical method would become the root cause for the “Spectre” and “Melt-down” attacks decades later. For another example, humans invented plastics many years ago, but at that time it was impossible to realize that the microparticles released from plastic waste would seriously pollute the whole ecological chain of the earth. Similarly, the inventor of the “666” pesticide had never expected that its widespread application would lead to huge ecological disasters; and the inventor of antibiotics had not expected that its abuse would put human beings in the danger of “super bacteria.”

1.4.2.2 Reliance on Industrial Ecology

Economic globalization is a milestone in the production mode of the human society. Although we are currently facing severe challenges in the international political situation and there is a certain degree of uncertainty, the basic pattern of the whole industrial chain based on the specialized division of labor, the technology chain, supply chain, and service chain where “all countries are dependent on each other” cannot be fundamentally changed. Therefore, the path dependence of the technology chain and supply chain in a certain field or industry is inevitable (there may be some differences in the degree of dependence of course), which makes it impossible to eliminate all the structural defects of software/hardware products in design, manufacturing, processing, marketing, and after-sales service.

1.4.2.3 Limitations of Verification Ability

As far as today's engineering ability is concerned, people can only check or verify the completeness of the expected design functions of a software system with millions to hundreds of millions of lines of code, or of a hardware chip composed of tens of millions to tens of billions of transistors. Such work not only is technically challenging but also costs a lot of manpower, material resources, and time. If we want to thoroughly check for "ghostly" vulnerabilities and hidden functions such as backdoors, overcoming the thorny problem of "state or path explosion" will be an insurmountable technical barrier in the foreseeable future, not to mention whether we can design large-scale, extremely complex, defect-free test specifications, tools, and software by relying on existing technological capabilities.

1.4.2.4 The Profit-Seeking Nature of Human Beings

As long as there are software/hardware code vulnerabilities, deliberately implanted backdoors, or unintentionally introduced trapdoors (such as backdoors in chip IP cores or open-source codes) in the development, design, manufacturing, marketing, and after-sales of cyberspace information facilities, CPS systems or related products, stakeholders will probably pursue various explicit or implicit benefits by any means. Provided profitable, cyberattacks may occur whether out of political, military, economic, or technical motives.

1.4.2.5 Unconstrained Actions

In the era of the Internet of Everything, as long as any CPS system or digital device has a high-risk vulnerability or an implanted or introduced backdoor (trapdoor), network attackers may launch attacks without the constraints of location, time, law, code of conduct, and moral norms, resulting in the failure of the service function of the target object and damaging the integrity, confidentiality, and availability of user information or data.

To sum up, vulnerabilities, backdoors, and potential malicious attacks in the whole industrial chain ranging from design, development, processing, manufacturing, sales, application, and after-sales service of software/hardware, algorithms, and protocols are common ESS problems in cyberspace. In particular, the threat caused by the common ESS problem in the homogeneous digital facilities or cyber-physical systems is more extensive, and its consequences are more destructive, which is the biggest "public hazard" (perhaps the biggest troublemaker) in the governance of digital economy or digital society! Unfortunately, so far there is almost nothing people can do about the unknown attacks based on common CESS problems. To make matters worse, for a long time, software/hardware products in the whole IT field and related industries cannot give quantifiable or verifiable technical indicators

for software/hardware design and measurement [62], including the existing cybersecurity defense products that claim to be “the patron saint of cyberspace security & safety.” Not only does this go against people’s understanding of the product quality assurance system in the modern market economy, but also the ESS problems in the software/hardware of digital facilities has a tendency to develop into an “eternal pain” in cyberspace.

1.4.3 Basic Nature of Common Problems

1.4.3.1 Ubiquitous Existence

No matter in philosophical or technical sense, there are inevitably “known unknown or unknown unknown” ESS problems in CPS systems or digital facilities composed of software and hardware, especially the common ESS problems related to vulnerabilities and backdoors spread throughout the network space.

1.4.3.2 Synergy of Internal and External Factors

A common ESS problem cannot be an available resource for cyberattacks under all conditions, nor can it lead to security incidents under all conditions. For example, when a cyberattack cannot reach the target system or cannot inject attack codes through vulnerabilities and backdoors, common ESS problems usually do not automatically cause network security incidents [7]. According to the attack surface (AS) theory, the attack surface of a target system refers to the subset of system resources that an attacker can use to launch an attack (as shown in Fig. 1.13). Attackers can use the methods, channels, data, software, and hardware in the

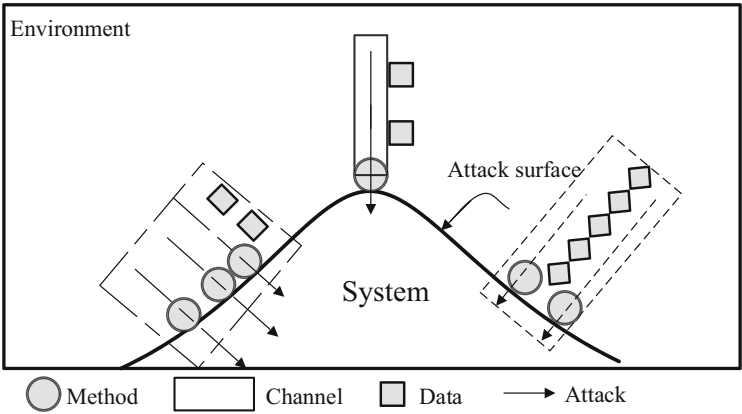


Fig. 1.13 Attack surface model

execution environment of the target object to achieve the purpose of an attack. So, vulnerabilities and backdoors in the attack surface are indispensable resources for cyberattacks.

1.4.3.3 Transferability of Contradictions

Any plug-in embedded or built-in software/hardware security protection measures on the target system, because the common ESS problems cannot be completely eliminated in the software/hardware design or manufacturing process, may introduce new unknown security problems with different forms while resolving a known security problem, that is, other or related common ESS problems will be introduced when transforming or resolving a contradiction. In other words, if any additional security measures cannot turn the main contradiction into a problem in other fields (spaces) or a relatively minor contradiction (e.g., turning the security contradiction into an economic one, turning an uncertain threat into a random one that can be expressed in probability, etc.) but simply make different expressions of the same contradiction within the same target object, it will be impossible for the common ESS contradiction to reach a reconciliation state of unity of opposites. For example, the traditional security technology usually adopts the method of patching once any software/hardware vulnerability in the target object is found, but if there are still unknown vulnerabilities or even backdoors after patching, isn't it a "security problem without solution"? In fact, all kinds of online antivirus software need to get privileges from the operating system before being able to play the expected role, but how can security protection software, Kaspersky or Symantec, prove that there is no ESS problem in itself? Therefore, the technical route of patching cannot avoid the problem of logical paradox in mechanism! By simple inference, it is not difficult to draw a general conclusion that as long as security protection components are built into the cyber-physical systems or a confidential computing execution environment, they will fall into the dilemma of endless credibility questioning no matter what their short-term effect is.

1.4.3.4 Intertwining of Problems

As information technologies, network technologies, intelligent technologies, and human attack factors infiltrate into the field of traditional functional safety or reliability, common CESS problems make the premise of elastic control theory based on random or "known unknown" nonhuman disturbances no longer valid, and the problem of functional safety inevitably evolves into an interweaving or overlapping problem between cybersecurity and functional safety, which is sometimes called the generalized robust control problem (GRCP). Unlike information security, which only focuses on the privacy, integrity, and availability of user data, functional safety pays more attention to the resilience or robustness of service functions. However, the intertwining or overlapping nature of cybersecurity and

functional safety problems often requires the CPS system or trusted execution environment (TEE) to meet the dual standards of information security and functional safety at the same time. In other words, to solve intertwined security problems, the target object must have integrated generalized robust control or cyber resilience/toughness [5].

In summary, common ESS problems based on vulnerabilities and backdoors are the most essential problems in the field of cyberspace security and safety, and how to effectively deal with “known unknown or unknown unknown” attacks is an urgent priority in the field of cybersecurity defense. However, the current cybersecurity defense paradigm based on the thinking of “better late than never,” the methodology of “patching,” and the practice norm of “best effort and bodyguard service” has become more and more inadequate for dealing with cyberattacks, and it is urgent to change the thinking, methodology, and practice norm.

1.5 Generalized Functional Safety Problems

1.5.1 *Definition of Generalized Functional Safety Problems*

As we know, people’s attention is often concentrated in areas closely related to personal safety. With the wide application of electrical/electronic/programmable electronic systems (E/E/PES) in industries closely related to personal safety, such as industrial control, power system, automobile industry, robots, medical equipment, etc., functional safety is becoming a major practical demand of related industries. Taking the field of automobile manufacturing as an example, ISO26262, a special standard for automobiles’ functional safety, was released in 2011, which mainly involves how to reduce safety risks caused by failures of automobiles’ electronic and electrical systems to an acceptable range. When the era of man-machine-thing integration and intelligent networking comes, cybersecurity problems are becoming increasingly prominent and forming a huge impact on the field of traditional functional safety. Taking the automotive industry as an example again, the cybersecurity problems of intelligent networked vehicles have rapidly penetrated into the field of traditional functional safety. According to statistics [63], from 2016 to 2020, the number of global cybersecurity incidents concerning automobiles increased by 605% (more than six times), and it more than doubled in 2019 alone. There are inevitably a lot of vulnerabilities and backdoors in the tens of millions or even hundreds of millions of lines of automobile software/hardware codes, allowing attackers to have enough available resources to cause new security risks including malicious control of vehicles. These new risks, different from the risk of leakage of private information or sensitive data, are directly related to the life and property safety of drivers and passengers, who have zero tolerance for them.

It is not difficult to come to the understanding that in the era of deep integration of the information world and the physical world, there is inevitably an extremely contradictory expression: on the one hand, a lot of digital infrastructure and

intelligent devices have transformed into cyber-physical systems (CPS) with intelligent networking properties; on the contrary, the software/hardware entities of CPS always have explicit side effects or implicit dark functions (including vulnerabilities, backdoors, and trapdoors). This makes traditional reliability or functional safety theories and practical norms face new challenges in the intelligent networking era where humans, machines, and things are deeply integrated. These challenges include not only the unreliability of system functions due to disturbances caused by random or natural factors but also abnormal expressions of system performance due to deliberate use of ESS problems.

If traditional technologies, processes, and materials have defects that are disturbed by nonhuman factors, leading to such problems as random hardware failure, system failure, etc. in the target system, and there are functional safety problems caused by human attacks based on target software/hardware vulnerabilities and backdoors, then all these problems can be collectively called “intertwined security and safety problems” or “generalized functional safety (GFS) problems.” It should be pointed out that the word “generalized” here emphasizes that the problem itself goes beyond the scope of traditional functional safety problems, crosses the boundaries of classic cybersecurity problems, and involves cross-cutting fields such as information storage and confidential computing.

It is not difficult for readers to find that GFS problems naturally involve ESS problems since the former include cybersecurity problems. At the same time, the GFS problem domain is also the solution domain of generalized robust control problems (GRCP).

1.5.2 Characteristics of GFS Problems

GFS problems have the following salient characteristics.

1.5.2.1 Multiple Relationships Intertwined

As the original boundaries between “functional safety,” “cybersecurity,” and “information security” are “collapsing” at a speed beyond our imagination, the “fusion” effect of intertwined security and safety problems has become increasingly prominent. For functional safety and information security-related systems that can be reached by cyberattacks, functional safety, cybersecurity, and information security often have multiple or high-dimensional complex relationships.

1. **Interdependence:** For example, if functional safety is not well guaranteed, the occurrence and consequence of security incidents will seriously weaken cybersecurity defense capabilities and endanger the normal order of the digital society, and unscrupulous cyberattacks will become extremely rampant. In the same way, if cybersecurity is not well guaranteed, network attacks will not only cause

- damage to the availability of the target system but also affect the confidentiality, availability, and integrity of data storage and processing, bringing double risks or destruction to functional safety and information security.
- 2. Mutual reinforcement: Sometimes the measures to deal with functional safety problems and those to deal with cybersecurity problems can be mutually enhanced. For example, a comprehensive test and detection of the system can not only find many potential failure points but also help to find potential threats such as vulnerabilities and backdoors, and thus both functional safety and network security can be reinforced.
 - 3. Contradiction: The existing measures are also contradictory. For example, to enhance the internal security of an intelligent networked system, it may be necessary to authenticate the communication process of the internal network and encrypt the data. However, the implementation of authentication and encryption technologies may damage the real-time performance of related system function processing, and many technical measures (such as various redundant structures) implemented or added to ensure the functional safety may also increase the attack surface of the network, thus increasing the pressure on network security defense.

Functional safety, network security, and information security intertwine, change, mutually feedback, and overlap, making the problem of the system security more complicated. Very often, one technical arrangement in a specific field will affect other measures in other fields, which is a thorny problem that cannot be solved by relying solely on one technical measure, as shown in Fig. 1.14. The CPS system represented by the overlapping area in Fig. 1.14 not only has functional safety risks caused by software/hardware failures but also is prone to the risk that cyberattacks can reach the system and affect functional safety (or information security). The two types of risks combined together constitute a complex intertwined security risk targeting at two different attributes of a target system and cannot be defused separately.

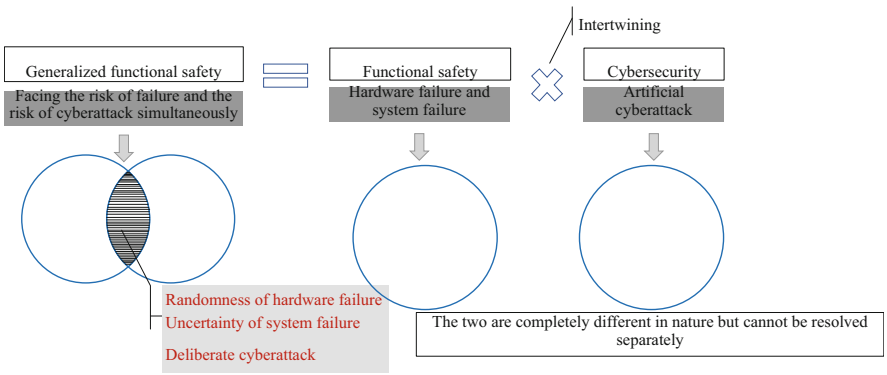


Fig. 1.14 Intertwined problems cannot be solved separately

1.5.2.2 Overlapping of Physical Space and Information Space

As shown in Fig. 1.15, if a CPS undertakes security-related functions, under the condition that cyberattacks can reach the system, the errors generated by the CPS in the information space may be caused by random nonhuman failures or system failures or by deliberate cyberattacks. For the CPS, errors in the information space may also lead to personal injury or environmental damage in the physical space, thus causing double (or even triple) security problems. Therefore, the generalized functional safety of a CPS is not only a complex intertwined problem of interdependence, conflict, and enhancement between functional safety and cybersecurity but also a superposed problem in physical space and information space on this basis. The classical functional resilience theory and method based on random failure cannot solve this problem independently, and the theory and method of “patching” network vulnerabilities based on knowledge accumulation and accurate perception cannot solve this problem independently either. In other words, the intertwining of multiple relationships and the overlapping of physical and information spaces determine that innovative theories, methods, and practical norms are needed to deal with GFS problems.

In addition, GFS problems and common CESS problems have the same nature in essence. Both are due to the uncertainty caused by network attacks targeting unknown vulnerabilities and backdoors in the CPS system, which make traditional functional safety problems more complicated.

1.5.2.3 Cascaded Amplification of Hazard

First of all, the intertwining of functional safety and cybersecurity has greatly increased the security risks of the system, making the potential hazards more destructive. For example, in the traditional communication, electric power, gas, oil transmission, and water supply networks that are crucial to people’s livelihood and social security, the damage caused by functional failures is generally controllable,

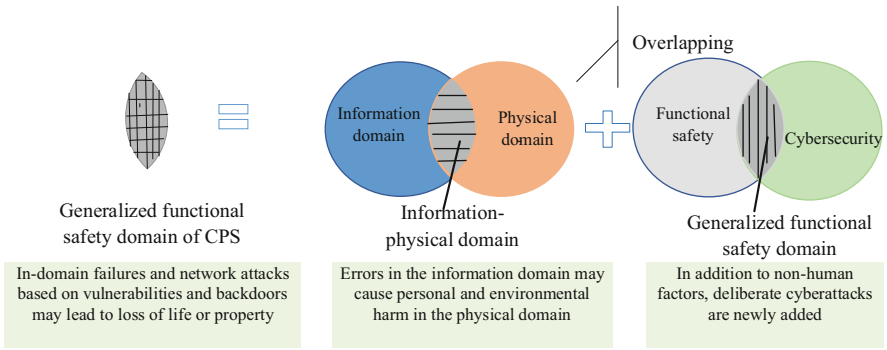


Fig. 1.15 The overlapping nature of GFS

and the recovery is relatively easy. However, the risks or threats of cybersecurity problems are extremely uncertain and destructive, and their consequence is often that “a defense breakthrough at one place damages or destroys the whole system” and even leads to a domino-like social disaster.

Second, the intertwining of functional safety and cybersecurity problems as well as the overlapping of physical and information spaces may cause cascaded amplification of accident hazards. Currently, cyber-physical systems are playing an irreplaceable role in security and safety-related industries such as industrial process control, smart transportation, intercity high-speed rail, smart medical care, smart home, nuclear industry, robots, and intelligent manufacturing. Cyberattacks on these target systems from the information level only cause harm to the physical space but also have an attack path that acts on thousands of similar targets at the same time and even “overlap” each other in the physical space and the information space, resulting in a cascaded amplification effect. Accidents and hazards caused by various failures in the field of functional safety may bring more opportunities to cyber attackers, and more cyberattacks may cause greater failures to the system. Such circular feedback may eventually lead to serious consequences such as power outages and urban traffic “paralysis and chaos” in the physical space or the real world, which may result in information security-related social disasters or secondary disasters and in some cases may produce “social shutdown” effects beyond the destructive power of nuclear weapons.

1.5.2.4 Continued Expansion

Since the early days of the twenty-first century, more and more security and safety-related systems have become cyber-physical systems (CPS) that integrate computation, communication, and control, providing a channel for the interaction between the physical world and the information world. In the industrial field, for example, security and safety-related systems monitor the parameters of the environment and production equipment in real time and feed them back to the information world for computation and analysis, so as to perceive the security situation and find security risks, based on which people will be able to take mitigation and protection measures against risks in the physical world. Self-driving cars take real-time radar, camera, GPS data, and high-precision map information as inputs and, after a series of computations and processing in the information world, accurately perceive the road condition and surrounding environment in the physical world. Then, the car’s automatic control system makes a decision and plans the driving route, so that the vehicle can reach its destination accurately, safely, and smoothly in the physical world through lateral and longitudinal control. Similar cases include cardiac pacemakers in the medical field, industrial robots, smart home equipment, and computer-controlled systems for natural gas pipeline valves. As a unity of computing process and physical process, these infrastructure facilities that provide services for human production and life make CPS have the functions of computing, communication, precise control, remote collaboration, and autonomy through system engineering

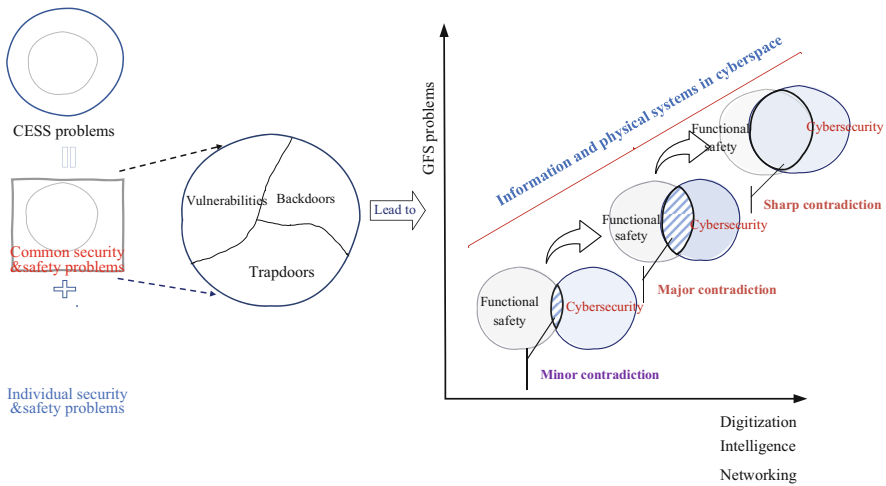


Fig. 1.16 Continuous expansion of GFS problem domain

such as environmental awareness, embedded computing, network communication, and network control.

With the ubiquitous application of digital, intelligent, and networked technologies, CPS will serve as a key support for new digital infrastructure, where the breadth and depth of GFS problems involved will continue to expand, as shown in Fig. 1.16. In terms of breadth, more types and larger numbers of cyber-physical systems will be affected not only by physical or logical faults and failures but also by cyberattacks based on software/hardware vulnerabilities and backdoors of digital facilities. In terms of depth, the continuous improvement of the networking level of CPS will, to a great extent, provide wider accessibility for cyberattacks, and the continuous enhancement of CPS intelligence often needs more software/hardware support, which means that more vulnerabilities and backdoor risks will be introduced at the same time.

In the face of ubiquitous GFS problems, whether CPS can ensure the expected function and performance under artificial, nonartificial, random, or uncertain disturbances is becoming a big question or doubt in system design and network security development. At the same time, these problems also pose a severe challenge to the existing information technology development paradigm and the digitalization process of the human society.

1.5.3 The Challenge of GFS Problems

First of all, many security-related cyber-physical systems that can be reached by cyberattacks have the attributes of both information and physical domains and the attributes of both functional safety and cybersecurity domains. The intertwining of

their endogenous functional safety and cybersecurity and the cyberattacks that may be caused by unknown design defects or unknown vulnerabilities and backdoors can cause errors or failures in the information domain, and once the errors or failures are “superimposed” in the physical domain, they may cause direct harm to life or property. According to the latest definition of functional safety, these are all within the scope of risk management of cyber resilience. Undoubtedly, the introduction of cyberattack factors has fundamentally changed the preconditions of traditional functional safety theories and methods or is subverting the corresponding engineering design and technical implementation criteria.

Second, the GFS problem of cyber-physical systems not only makes functional safety have randomness and uncertainty, two reliability problems with completely different mathematical attributes, but also puts it under the impact of more complex and multidimensional human and nonhuman disturbances or threats with two different attributes. The connotation and extension of functional safety have broken through the basic laws that general faults lead to errors, errors cause failures, and failures bring security risks, posing unprecedented challenges to the theories, methods, and practice norms of functional safety that have been followed so far.

Third, in the field of cybersecurity, it is a recognized world problem that security or security quality cannot be quantitatively designed, verified, and measured. When cyberattacks are introduced as a factor that can also disturb system functions, even the disturbances caused by various nonhuman factors in the field of functional safety can be described by probabilistic methods (in fact, the uncertainty of software faults cannot be expressed quantitatively so far), and the generalized uncertain disturbances faced by the entire system (perturbations caused by various faults or disturbances caused by cyberattacks) will also become difficult to characterize using probabilistic tools. Currently, none of the global functional safety design standards for CPS can quantitatively describe the impact of “unknown unknown” cyberattacks on the functions of the target system. Moreover, cyberattacks targeting unknown vulnerabilities and backdoors often make the reliability or functional safety design of complex systems such as CPS directly fall into the dilemma of serving as a mere “ornament.” That is to say, the target system may be 99.99% functionally safe or even safer before an attack succeeds, but once the attack becomes successful, the system’s functional reliability will immediately reduce to zero.

Finally, although functional safety and cybersecurity theories and technologies have each experienced a long period of development and evolved into different disciplines, technologies, concepts, and cultures, with the technical progress of security-related systems in digitalization, intelligence, and networking, the demand for functional safety and cybersecurity will be more and more concentrated in the same intelligent, networked, and digital system, so it is necessary to establish a comprehensive security system to solve the GFS problem in an integrated manner. However, the reality is that two types of risks, mechanisms, causes, technical measures, and even two cultural and language systems are still in the initial stage of seeking consensus and coordination. Many basic theories are missing, and major breakthroughs in cutting-edge technologies to support functional safety and cybersecurity are urgently needed. Now, people are faced with quite a lot of challenges,

including but not limited to a lack of relevant standards, technologies, tools, and verification methods.

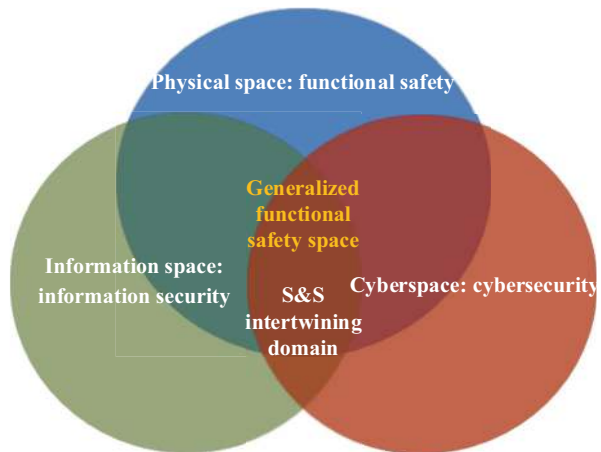
Although GFS problems will bring unprecedented challenges to the design and manufacturing of information infrastructure or various cyber-physical systems, the R&D institutions that are the first to make breakthroughs in overcoming these problems in the future will gain an unparalleled first-mover advantage, and pioneering manufacturers in this field will have the opportunity to become the leader in GFS products and market in a more complex, intelligent, and networked world.

1.5.4 Definition of GFS

If there is a robust construction model, which can not only keep the given model function within the safety threshold of quantifiable design under the perturbation of some traditional random factors or known unknown factors but also ensure the safety of the model’s functions under the action of cyberattacks based on unknown software/hardware vulnerabilities and backdoors in the model, then the model is considered to have the attribute of generalized safety and security. It is not difficult to prove that the GFS indicator can be expressed quantitatively using the generalized reliability dimension.

Obviously, GFS essentially needs to be obtained by the endogenous characteristics of “construction determining security.” Its problem domain is an area in which functional safety, cybersecurity, and information security are intertwined (as shown in Fig. 1.17). Its solution domain is how to ensure that the system function has the robustness of “quantifiable design, verifiable measurement” or cyber resilience (toughness) and a “trusted execution environment” for confidential or trusted computation under the circumstance of random failure and uncertain perturbation of

Fig. 1.17 Domains of three GFS problems



software/hardware, as well as under human attacks and disturbances based on known or unknown common ESS problems.

As we all know, the thinking perspective of the existing security defense paradigm is based on the accumulation of prior knowledge (knowledge base) such as threat perception and feature extraction, which can provide highly targeted security protection against known or even “known unknown” security threats but cannot effectively deal with “unknown unknown” network attacks based on common ESS problems or even ESS problems; using the precise analysis and location of vulnerabilities and backdoors, virus Trojans, and related utilization mechanisms and behavioral characteristics as the basic means, follow the “Headache Medicine.” Although the patching methodology of “fixing the problem by taking stopgap measures” can conditionally reduce network threats or attacks based on problems such as known vulnerabilities, backdoors or viruses, and Trojans through continuous evolution and iteration (debug), it can do nothing about the common ESS problems of targeted “on-patch, after-patch” or add-on defense measures and the possible new ESS problems in the target object caused by these measures. It is completely powerless; as for its “best effort” engineering practice standards, it is impossible to provide quantifiable design and verifiable measurement security indicators for the target system, both in theory and practice, even if active sensing, behavioral profiling, and intelligent prediction technology are introduced (with the help of artificial intelligence methods to analyze sandbox (data obtained by built-in means such as honeypots, intrusion detection, and embedded probes). The determination of attack behavior is still strongly related to the precise understanding of the target system’s software and hardware environment and the defender’s past experience and knowledge accumulation. Although attacks with clear characteristics or mechanisms can provide quantitative or semiquantitative security indicators (the impact of recognition errors such as false alarms or false alarms is not excluded), the theoretical level cannot be effective for unknown network attacks that are not recorded in the knowledge base or that cannot be effectively inferred, not to mention obtaining functional safety quality indicators that can be quantified from engineering and verification measurements.

It can be seen that there is an urgent need to carry out revolutionary changes in the current mainstream cybersecurity defense paradigm and classic functional safety specifications; propose new perspectives, new methodologies, and new practical norms for solving problems; and establish a new theory and method system, and through the system engineering, technical route of “Structure Determines Safety” has gained GFS attributes that integrate the intertwined problems of functional safety, cybersecurity, and even information security.

1.6 Difficulties in Defending Against CESS Problems

1.6.1 *Advanced Persistent Threats Are Difficult to Defend Against*

An APT (advanced persistent threat) attack is an intelligent cyberattack that refers to the use of advanced means of computer network attacks and social engineering attacks by relevant organizations or groups to make long-term and persistent infringement on specific high-value data targets or functional platforms. APT is a term coined by an information security analyst of the US Air Force in 2006. Generally speaking, APT has the following three characteristics.

Highly Advanced The attacker is an expert in hacking who can independently develop hacking tools, exploit vulnerabilities, and achieve the expected purpose of hacking through multiple combinations of a variety of methods and tools.

Persistent Infiltration The attacker infiltrates a specific target for a long time by launching sustained attacks while doing their best not to be detected.

Threat An APT attack is a man-made attack coordinated and directed by the organizer. With a clear goal, the hacking team is well-trained, well-organized, well-funded, and politically, economically, or militarily motivated.

Table 1.4 lists the main APT attacks and the vulnerabilities exploited.

It can be seen that almost every APT attack exploited the 0-day vulnerability, and some exploited N-day vulnerabilities. This shows that vulnerabilities have become an important means that APT attacks rely on, so vulnerabilities play a key role in cyberattacks and defense, serving as an important factor affecting the asymmetric game between the offensive and defensive sides.

1.6.2 *Uncertain Unknown Threats Are Unpredictable*

Former US Secretary of Defense Donald Rumsfeld once famously said: “(We know) there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns — the ones we don’t know we don’t know.”

References [64, 65] divide different types of knowledge into four quadrants according to the degree of mastery of models and data: known knowns, known unknowns, unknown knowns, and unknown unknowns, as shown in Table 1.5.

“Known knowns” (with models and data) represent the knowledge the system possesses, that is, known and independently verifiable facts about itself and the environment. Systems equipped with “known knowns” are able to detect, interpret, and act upon their observations in the environment. It is worth pointing out that even with a good model and excellent data, there is no guarantee that the entire behavior

Table 1.4 APT incidents and vulnerabilities exploited

Serial number	APT incidents	Vulnerabilities exploited	Time of occurrence	Impact
1	Operation of “APT-Q-36”	CVE-2013-3906 CVE-2014-4414 CVE-2017-8570	November 2009 March 2018 or earlier	Mainly stealing data from scientific research, education, and government agencies
2	Operation of “APT-Q-37”	CVE-2012-0158 CVE-2017-12824 vulnerabilities in Microsoft Formula Editor	November 2013 Early 2018	Stealing confidential information of a certain ministry, large central energy enterprises, the Pakistani government and personnel
3	Operation of “Equation Group”	CVE-2016-6366 CVE-2016-6367	2016	Controlled IPs and domain names are distributed in 49 countries, mainly concentrated in the Asia-Pacific region
4	Harvest Event (DX-APT1)	CVE-2015-1641 CVE-2012-0507 CVE-2013-0640	March 2015 or earlier	Stealing embassy address books as well as military and diplomacy-related documents
5	Petya ransomware incident	CVE-2017-0199	June 2017	Encrypting files or even causing system crashes
6	DarkHotel APT	CVE-2018-8174 CVE-2018-8242 CVE-2018-8373	May 2018	Stealing crucial information of Chinese corporate executives, defense industry, electronics industry, and other important personnel and institutions
7	Android 0-day vulnerability incident	One 0-day vulnerability in Android media driver v412	September 2019	More than 1 billion Samsung, Huawei, LG, and Sony smartphones are vulnerable to attacks, and attackers can use SMS to completely access the email in the device
8	SolarWinds software supply chain attack incident	SUNBURST, a backdoor embedded by hackers in SolarWinds Orion software update package	December 2020	Affecting the computer networks of government agencies, critical infrastructure, and some private enterprises within the United States, including the Department of Homeland Security (DHS)

(continued)

Table 1.4 (continued)

Serial number	APT incidents	Vulnerabilities exploited	Time of occurrence	Impact
9	Poisoning incident of Florida water works system	Stealing TeamViewer credentials	February 2021	Private companies related to Florida’s water industry raised the level of sodium hydroxide lye for poisoning
10	Large-scale cyberattack on Ukrainian government and bank websites	DDoS vulnerabilities targeting network devices	February 2022	Paralyzing many government agencies and banks in Ukraine

Table 1.5 Quadrants of knowledge

	With data	Without data
With models	Known knowns (previous knowledge)	Unknown knowns (Untapped knowledge) Research methods: logical reasoning, simulation
Without models	Known unknowns (Incomplete knowledge) Research methods: statistical analysis, machine learning	Unknown unknowns (Uncertainty)

of the system can be effectively predicted. Every model has a certain boundary of applicability, no data set is absolutely complete, and there are certain types of inaccuracies. In this quadrant, learning includes improving existing knowledge and detecting and deleting incorrect or outdated knowledge to ensure better decision-making.

“Known unknowns” (with data but without models), the most common situation, refer to the gaps in factual knowledge or professional knowledge that are likely to pose a threat to the system due to their own ambiguous behavior and state. For example, a system may have a large amount of observed data (about itself and its environment), but it may not be able to explain it. It needs to use statistical analysis to build a model based on this, or summarize it into an interpretable model, or generate a corresponding model through AI training, which can infer, predict, and observe subsequent data. In this quadrant, there is room for learning (from observation) so that system behavior can not only be detected but also explained.

“Unknown knowns” (with models but without data), seemingly contradictory and paradoxical, refer to untapped (or hidden) knowledge that, if unlocked, can improve the quality of decision-making. A system can migrate from “unknown knowns” to “known knowns” through shared knowledge. In fact, there is nothing particularly mysterious about the process in which something changes from being implicit to being explicit, for it lies at the core of mathematical knowledge. In mathematics, we start with a set of axioms and use logical reasoning tools to prove, step by step, the

implicit content of explicit axioms. Another possibility is computer simulation, i.e., building a simulated world in the computer and then creating different environments in the electronic world and checking the consequences of these changing conditions. Simulations often produce unexpected behaviors (sudden attributes) that make explicit what is implicit in the rules and background constraints imposed by the scenario. The model here is the structure and rules (axioms, so to speak) of the scenario, but there is no data. In fact, data must be generated by the simulator in the process of exploring the axioms that constitute the simulation.

“Unknown unknowns” (without models and without data) refer to potential uncertainties that are unknown to the system. Here is a truly unknown territory. How can one imagine an unknown that is strictly unknown? How does the system learn in this quadrant? One possible way is to explore and consider the changes of known events (events that have happened), observe the results of those invisible changes (both positive and negative), and learn from them. For example, if we consider the types of events that have happened and then consider the variants of the events that we have never seen before, then we will gradually enter the unknown unknown territory. It needs to be emphasized that “unknown unknowns” are unsolvable problems if seen from an epistemological perspective (unless one has a God’s perspective or superpower). However, in an engineering and technical sense, sometimes even if the cause and nature of a problem are not known, it can still be technically handled if its results or manifestations are known. For example, although we currently do not know the causal relationship of the analysis results of “black-box algorithms” of big data or artificial intelligence (which does not mean we give up exploration), this does not prevent the use of these results to solve other problems.

In cyberspace, known threats refer to the types of attacks with obvious characteristics that can be labeled and detected. Of course, known threats are all unknown threats at first, such as cross-site scripting attacks, brute-force attacks, misconfiguration, ransomware, watering hole attacks, phishing attacks, SQL injection attacks, DDoS attacks, and so on.

In email-based APT attacks, for example, the attacker controls the target by scouting for, scanning, and exploiting the vulnerabilities in Flash, Excel, etc. In this process, the attempts of scouting and scanning are known threats, against which many defense methods have been developed by the industry and academia. In the process of exploiting vulnerabilities, the 0-day vulnerabilities and backdoors used by the attacker fall into the category of known unknown threats. The types of vulnerabilities are known and the means of exploitation are not new, but the specific vulnerability to be exploited is unknown. This type of threat is generally difficult to detect and defend against, but the threat can usually be modeled and prevented. Unknown unknown threats more often refer to backdoors and unknown types of vulnerabilities. According to Reuters, Snowden-leaked documents said that with the approval of NIST, the National Security Agency (NSA), and RSA, an encryption company, reached an agreement worth more than 10 million US dollars, requiring that a backdoor be placed in the encryption technology widely used in mobile terminals, so that NSA can easily crack all kinds of encrypted data through the backdoor program of the random number generation algorithm Bsafe. If the report is

true, then the backdoor placed by NSA can be regarded as an unknown threat. This is because a backdoor implanted in an algorithm that has been proved to be mathematically safe is completely unpredictable. At the same time, we can see that mobile platforms have now become a regular channel for APT attacks. In September 2021, Apple released an emergency security update to fix two 0-day vulnerabilities: CVE-2021-30860 and CVE-2021-30858. There have been many incidents of using these vulnerabilities to attack iPhone and Mac devices before. In addition, by exploiting these vulnerabilities, attackers can install spyware Pegasus on the target iPhone, which is very powerful in monitoring key iPhone users.

As an asymmetric cyber deterrence capability, vulnerabilities and backdoors are strategic resources deliberately pursued, designed, and carefully reserved by countries advanced in network technologies. These countries are bound to convert their scientific advantages in cyberspace into technological and industrial advantages and gain the seller's advantages in "planting backdoors" and "hiding vulnerabilities" through technology exports, product supply, channel distribution, service provision, and market monopoly and then gain the strategic advantages of "one-way transparency of information" and "absolute freedom of action" in cyberspace.

With the development of the open-source community, backdoors are often added to open-source codes. Therefore, OpenBSD and OpenSSL have been questioned. For example, the Heartbleed vulnerability is set hidden and thus is very difficult to find. In addition, with the development of software/hardware technologies, it is not difficult for existing resources or technologies to support the setting of vulnerabilities in hardware and firmware. In 2021, Verkada, an American security camera company, confirmed that it suffered a large-scale hacking. In this incident, the hacker logged into the user's camera as a super administrator through the maintenance backdoor reserved by the manufacturer, accessed, and then released real-time video feeds from enterprises such as Tesla and Cloudflare. Furthermore, hackers can also execute arbitrary shell instructions through the reserved backdoor. This means that in the past few years, digital product manufacturers can perform any action at any time through preset backdoors without the user's knowledge [66].

1.6.3 Limited Effects of Traditional "Containment, Patching, and Repair"

In the past, the defense methods for vulnerabilities and backdoors were characterized by "containment, patching, and repair."

1.6.3.1 Inevitable Omissions Despite the Efforts to Minimize the Introduction of Vulnerabilities During the Development Process

To minimize the introduction of vulnerabilities in the development stage, Microsoft proposed the security development lifecycle (SDL) management model to guide the software development process. SDL is a security assurance process, which introduces the principles of security and privacy in all stages of development. Since 2004, SDL has been a mandatory strategy implemented by Microsoft. However, the implementation and operation of SDL is very costly, and only giant companies can afford to adopt it so far.

In addition to the SDL method, some lightweight tools have been developed to help reduce some common vulnerabilities in coding, including compiler-based source code analysis tools such as Clang Static Analyzer, which is a built-in static analysis tool for C, C++, and Objective-C source codes in Clang, the front end of the open-source compiler, and is used to check for general vulnerabilities (buffer overflow, format strings, etc.). Moreover, the gcc compiler supports the `-Wformat` security option and can be used to detect format string vulnerabilities in source codes. Once the compiler running these plug-in options detects a vulnerability, it will issue a warning and notify the software developer of the location of the suspicious vulnerability. Then, by searching for, locating, analyzing, and confirming the suspicious codes, the developer will be able to modify and eliminate the vulnerability.

Although “containing” vulnerabilities by any of the above methods or means in the development stage can reduce problems to some extent, the ESS contradiction cannot be completely eliminated, and “fish slipping through the net” is definitely unavoidable.

1.6.3.2 Busy Vulnerability Patching Despite Self-Checking Done in the Testing Stage

Vulnerability mining has experienced the development of manual mining, fuzzy testing, symbolic execution, intelligent mining, and other technologies. At first, vulnerability mining mainly relied on manual reverse analysis, which was time-consuming, labor-intensive, and difficult to be conducted on a large scale. Then, the fuzzy testing method tests the program by generating samples through random mutation, which improves the automation of vulnerability mining and becomes a widely accepted method in the industry. At present, famous fuzzy testing tools include AFL, LibFuzzer, honggfuzz, etc. In response to the testing requirements of high code coverage and the inherent blindness of fuzzy testing, security experts put forward a symbolic execution technology [67–70]. This technology symbolizes the input, simulates the execution of the program through symbolic expressions, and, when encountering conditional branches, collects constraints and solves the corresponding inputs of the two branches, thus obtaining the corresponding

relationship between the input and the path and effectively improving the code coverage of the testing process. Representative tools include SAGE (Scalable Automated Guided Execution) [71], KLEE [72], angr [73], etc. Recently, with the development of machine learning and other technologies, the application of intelligent methods in vulnerability mining has become a research hotspot. For example, in view of taint-propagation vulnerability, Yamaguchi et al. [74] in Germany proposed to use machine learning to automatically extract vulnerability patterns. Aiming at the problem of sample construction with high code coverage, Microsoft proposed to use deep learning to strengthen the fuzzy testing method. The experimental results show that the parser test coverage for ELF and PNG is improved by 10% compared with AFL. Experimental results show that compared with AFL, the parser test coverage for ELF and PNG is improved by 10%.

Although manufacturers and security researchers are committed to continuously improving their vulnerability mining capabilities, they have adopted the mode of self-checking and patching during the testing phase. However, not long after the software is released, third parties will report one vulnerability after another for patching. In addition, in recent years, there have been many cases of targeted attacks using wild vulnerability exploitation tools, with a large number of 0-day vulnerabilities in existence underground.

1.6.3.3 Nonstop Exploitation of Vulnerabilities Despite Constant Improvement of Vulnerability Mitigation Measures

Since vulnerability cannot be eradicated, security experts take mitigation measures to make it more difficult to exploit them, vulnerability exploitation and mitigation technologies have always been the focus of vulnerability attack and defense research, but most of the cases are that “as virtue rises one foot, vice rises ten.” The creation of a mitigation measure often leads to the birth of a bypass technology. An example is the confrontation process between the offensive and defensive ends of Windows platform stack protection technology, as shown in Table 1.6. At first, version 1.0 of GS protection was introduced into Visual Studio 2002, security cookies were inserted into the prologue function, and then cookies were checked in the epilogue. If any inconsistency was found, program execution would be terminated. In 2003, Litchfield [75] proposed a method of covering the bypass by structured exception handling (SEH). Later, the GS protection option version 1.1 and SafeSEH protection mechanism were introduced into Visual Studio 2003. In 2010,

Table 1.6 Attack on and defense of Windows platform stack

Vulnerability mitigation measure	Bypass technology
GS cookie protection	Method of overriding SEH handler
SafeSEH protection	Method of forging SEH linked lists
DEP protection	ROP technology
ASLR technology	Information leaking methods

Berre et al. proposed forging the SEH linked list to bypass SafeSEH protection. To prevent the execution of shellcode in the stack, data execution prevention (DEP) [76] was introduced. If the executed codes are located in a nonexecutable memory page, an exception will be thrown, and the process will be terminated. To bypass DEP protection, the technology of return oriented programming (ROP) [77] was invented, which uses the existing code snippets of a code fragment to realize the function of shellcode. To prevent attackers from accurately finding ROP gadgets, the ASLR mechanism was introduced into Windows to randomize the loading base address of DLL. Then, attackers invented an information-revealing method, by which they were able to accurately locate the ROP gadgets after revealing the loading base address of DLL.

In recent years of Pwn2Own Hacking Contests, hackers can still bypass the protection mechanisms of the latest versions of Windows, MacOS, and Ubuntu operating systems, which have been continuously reinforced with vulnerability mitigation measures, to gain the highest system privileges. This shows that although vulnerability mitigation measures increase the difficulty of vulnerability exploitation, they still cannot completely prevent successful attempts of vulnerability exploitation.

1.6.3.4 Cases of Bypassing the Protection Mechanism Occurring from Time to Time Despite Carefully Designed Detection Mechanisms such as Whitelists

In addition to continuous efforts to reduce the number of vulnerabilities and increase the difficulty of vulnerability exploitation, security experts also hope to restrict the subjects who exploit vulnerabilities. A well-designed whitelist is a feature-based detection mechanism. By default, programs that are not in the whitelist cannot run and data cannot pass. However, many whitelist bypass technologies have also emerged.

For example, starting with Vista, Windows introduced a user access control (UAC) mechanism that requires user confirmation when a program makes changes to the computer. UAC uses a whitelist mechanism to select trusted applications, but on Windows 7, attackers can exploit DLL hijacking vulnerabilities in whitelisted programs to bypass UAC protection. Windows 8.1 restricted some DLL hijacking, but attackers found a new DLL that can be hijacked. Windows 10 restricts more DLL hijacking, but it still cannot completely eliminate it. Another method of bypassing UAC through the program unloading interface has emerged. Some other security protection software also adopt a whitelist strategy and only allow whitelisted applications to run on the terminal. Bit9, for example, adopts a whitelist strategy, which only allows applications in the whitelist to run on the terminal so as to protect the security of the system, but it itself is vulnerable to attacks. Metasploit has released a DLL injection attack load for Bit9 Parity 6.0.x. In addition, manufacturers such as Bit9 have suffered intrusions, resulting in the leakage of keys and their own digital signature certificates being bypassed. Similarly, AppLocker, an application control

policy introduced in Windows 7, also uses application whitelisting to protect system security, but it itself has security vulnerabilities, and its whitelisting mechanism can be bypassed through `regsrv32.exe`, `presentationhost.exe`, etc.

1.6.3.5 Constant Introduction of New Exploitable Vulnerabilities Despite Timely Patching of Old Vulnerabilities

There may be the risk that a vulnerability is patched incompletely or that a bigger vulnerability is introduced after the existing vulnerability is patched. The outbreak of the Apache `log4j` vulnerability in December 2021 caused all security personnel to fall into a huge panic. Apache immediately released an emergency patch to fix the `Log4Shell` vulnerability in the `Log4j2.15.0` version. However, there were vulnerabilities in the hastily released patch, which could lead to denial-of-service (DoS) attacks under certain circumstances. The new vulnerability introduced by the patch, numbered `CVE-2021-45046`, may allow an attacker to control the input data of thread context map (also known as MDC) and in some cases use Java Naming and the Directory Interface (JNDI) lookup mode to craft malicious input data, thus resulting in DoS attacks. For this reason, Apache personnel must continue to update the `Log` to fix the new vulnerability introduced by the patch. But this does not mean that the security impact of `Log4j` is over. Researchers still do not know how many `Log4Shell`-related vulnerabilities currently exist and how they are related to each other. As the original vulnerabilities are continuously studied by more and more attackers, new `Log4j` vulnerabilities may pose a security threat in the future.

As late as the end of June 2021, security researchers were still discussing a vulnerability in the Windows “Print Spooler,” which they named `PrintNightmare`. `PrintNightmare` involves two vulnerabilities, `CVE-2021-1675` and `CVE-2021-34527`. Initially, there was a security vulnerability `CVE-2021-1675` in the Windows Print Spooler service, which allowed attackers to complete account privilege escalation on the machine with the vulnerability. After the staff released a patch for the `CVE-2021-1675` vulnerability, they thought the incident was over, but the more serious `CVE-2021-34527` vulnerability appeared subsequently. It was revealed that this vulnerability can not only be used to escalate account privileges but also achieve remote code execution. Security vulnerabilities that appear multiple times in the same product make it difficult for security maintenance personnel to fix and prevent them.

1.7 Summary of This Chapter

In a universal philosophical sense, the physical world is inherently uncertain. In the sense of human transforming nature, the role of technological invention is to transform uncertain problems into controllable probability problems under constraints. In cyberspace, “there is no absolute security,” but the author absolutely

believes that there are engineering methods that can transform generalized uncertain disturbances into controllable probability events or “prove self-innocence.”

If the essence of ESS problems is the expression of the inherent contradictions of things, then the essence of CESS problems is the expression of digital products or cyber-physical systems’ inherent contradictions, with some basic characteristics such as inevitability of existence, contingency of presentation, and temporal-spatial limitation of cognition. Their most prominent manifestation is that the underlying driving technology of the digital ecosystem has “genetic defects” in terms of endogenous security and safety. In particular, the ubiquity, the synergy between internal and external factors, the convertibility of contradictions, and the intertwining of common ESS problems in every corner of cyberspace make common ESS problems related to software/hardware vulnerabilities and backdoors become the most common and intractable GFS problems of digital products. What needs to be pointed out in particular is that the contradiction between vulnerabilities/backdoors and digital products at the security level means that CESS problems can only achieve a unity of opposites through evolution, transformation, or reconciliation, and it is impossible to completely eliminate the contradiction itself. In other words, all technical routes that violate the identity and opposition of contradictions, as well as technical measures that try to totally eliminate all vulnerabilities and backdoors by “debugging” or “patching,” will inevitably fall into a logical paradox at the philosophical level. At present, the mainstream add-on or plug-in cybersecurity defense measures cannot jump out of the “better late than never” mindset based on prior knowledge, which is largely due to the failure to fully recognize the contradictory nature of ESS problems, so the methodology of “closing the door and stopping the leak” followed cannot effectively deal with unknown and uncertain cyberattacks in mechanism, and there is no way to ensure that no new vulnerabilities will be introduced or caused while fixing an ascertained vulnerability. Because of this, it is even more impossible for the “best-effort” practice to achieve the technical goal of quantifiable design and verifiable measurement of security and safety. Therefore, the question of why digital products can have no GFS commitment has a seemingly absurd but very comfortable answer: “Because of the lack of theories and methods, it can only be selectively ignored technically!”

At present, the cybersecurity defense paradigm, which is composed of the thinking perspective of “learning from mistakes,” the methodology of “patch after patch,” and the practice of “doing the best,” not only has genetic defects in the defense system and mechanism against “unknown unknown” network attacks based on ESS problems but also is difficult to get rid of the dilemma of getting half the result with twice the effort in engineering. In the foreseeable future, human beings will continue to try to make fundamental repairs at the engineering level through add-on, built-in, embedded, or other “surgical” measures, so it is theoretically impossible to jump out of the vicious circle of patch after patch. To control the interaction between internal and external factors in the absence of prior knowledge, and reduce the possibility of unknown ESS problems being transformed into unknown ESS incidents, we urgently need to develop creative theories and subversive technologies, and only by doing so can we ensure that digital products have the

integrated ESS function to deal with potential threats or destruction such as human or nonhuman, random or uncertain disturbances, etc. To this end, we should not only change the thinking perspective, methodology, and practice norms of the traditional cybersecurity defense paradigm but also transform the underlying driving paradigm of the current digital ecosystem.

References

1. Hegel G, Wallace W. The logic of Hegel [M]. Oxford: Oxford University Press; 1975.
2. Birolini A. Quality and reliability of technical systems [M]. Berlin: Springer Berlin Heidelberg; 1994.
3. Wu Jiangxing. Cyberspace endogenous security and safety – mimic defense and generalized robust control [M]. Beijing: Science Press; 2020.
4. Xiao Qian, Li Xiulin, Wang Yongxiang. Principles of dialectical materialism [M]. 1st ed. Beijing: People's Publishing House; 1981.
5. Wu JX. Problems and solutions regarding generalized functional safety in cyberspace [J]. Secur Saf. 2022;1(1):1–13.
6. Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-design and Default. Publication April 13.2023. Cybersecurity and Infrastructure Security Agency.
7. Wu Jiangxing. Cyberspace's endogenous safety and security problem and the countermeasures [J]. Sci Sin Inf. 2022;52(10):1929–37.
8. Dorothy E. Cryptography and Data Security [M]. Boston: Addison-Wesley Longman Press; 1982.
9. Bishop M, Bailey D. A critical analysis of vulnerability taxonomies [J]. Comput Sci. 1996:1–14.
10. Krsul IV. Software vulnerability analysis [R]. Purdue University, West Lafayette; 1998.
11. Shirey R. Internet Security Glossary[R]. RFC 2828/Internet Engineering Task Force, 2000.
12. Stoneburner G, Goguen A, Feringa A. Risk management guide for information technology systems[J]. National Institute of Standards and Technology Special Publication 800–30, 2002.
13. Gattiker UE. The information security dictionary [M]. New York: Springer; 2004.
14. Wu Shizhong, Guo Tao, Dong Guowei, et al. Software vulnerability analysis technology [M]. Beijing: Science Press; 2014.
15. Wikipedia. Computation [EB/OL]. 2016. <https://en.wikipedia.org/wiki/Computation>
16. Nie Chujiang, Zhao Xianfeng, Chen Kai, et al. A microscopic vulnerability number prediction model [J]. Comput Res Dev. 2011;48(7):1279–87.
17. Marciniak JJ. Encyclopedia of software engineering [M]. New York: Wiley; 2002.
18. ff_good. Software vulnerability analysis [EB/OL]. 2020. https://blog.csdn.net/fufu_good/article/details/104154318
19. Pincus J, Baker B. Beyond stack smashing: recent advances in exploiting buffer overruns [J]. IEEE Secur Priv. 2004;2(4):20–7.
20. Bishop M, Dilger M. Checking for race conditions in file accesses. Comput Syst [J]. 1996;9(2): 131–52.
21. Halfond W G, Viegas J, Orso AF. A classification of SQL-injection attacks and countermeasures [C]. In: IEEE international symposium on secure software engineering, Washington, 2006. p. 13–15.
22. Shankar U, Talwar K, Foster J S, et al. Detecting format string vulnerabilities with type qualifiers [C]. In: USENIX Security symposium, 2001, vol. 10(16). p. 201–20.
23. Wang T, Wei T, Lin Z, et al. IntScope: automatically detecting integer overflow vulnerability in x86 binary using symbolic execution[C]. In: Network and distributed system security symposium, San Diego, 2009. p. 1–14.

24. Duchene F, Groz R, Rawat S, et al. XSS vulnerability detection using model inference assisted evolutionary fuzzing [C]. In: 2012 IEEE fifth international conference on software testing, verification and validation (ICST), Montreal, 2012. p. 815–17.
25. Heiland D. Web portals, gateway to information or hole in our perimeter defenses [EB/OL]. 2020. <https://www.doc88.com/p-384368730132.html>
26. CVE. Zend framework zend_log_writer_mail-classshutdownfunction permission and access control vulnerability [EB/OL]. 2016. <http://cve.scap.org.cn/CVE-2009-4417.html>
27. Feist J, Mounier L, Potet ML. Statically detecting use after free on binary code. J Comput Virol Hacking Tech [J]. 2014;10(3):211–7.
28. Kim Y, Daly R, Kim J, et al. Flipping bits in memory without accessing them. ACM SIGARCH Comput Archit News [J]. 2014;42(3):361–72.
29. Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds [C]. In: Proceedings of ACM conference on computer and communications security, Chicago, 2009. p. 199–212.
30. Gras B, Razavi K, Bosman E, et al. ASLR on the line: practical cache attacks on the MMU [C]. NDSS. 2017;42(3):1–15.
31. Tesler LG. Networked computing in the 1990s [J]. Sci Am. 1991;265(3):86–93.
32. Abbott RP, Chin JS, Donnelley JE, et al. Security analysis and enhancements of computer operating systems [M]. Washington: National Bureau of Standards; 1976.
33. Bisbey R, Hollingsworth D. Protection analysis project final report [M]. Marina Del Rey, CA: Information Sciences Institute, University of Southern California; 1978.
34. Aslam T. A taxonomy of security faults in the UNIX operating system [M]. West Lafayette: Purdue University; 1995.
35. Neumann P G. Computer-related risks[M]. Addison-Wesley Professional, New York, 1995.
36. Cohen FB. Information system attacks: a preliminary classification scheme [J]. Comput Secur. 1997;16(1):26–49.
37. Krsul I, Spafford E, Tripunitara M, et al. Computer vulnerability analysis [R]. West Lafayette, IN, USA: COAST Laboratory, Purdue University; 1998.
38. Landwehr CE. A taxonomy of computer program security flaws [J]. ACM Comput Surv. 1994;26(3):211–54.
39. Bishop M. A taxonomy of UNIX system and network vulnerabilities [R]. Davis, CA: Department of Computer Science University of California at Davis; 1995.
40. Du W, Mathur A P. Categorization of software errors that led to security breaches [EB/OL]. 1997. <https://www.semanticscholar.org/paper/Categorization-of-Software-Errors-that-led-Du/b5a98424b12c9af6b6cd706ec13b3b0366e7da2d>
41. Jiwnani K, Zekowitz M. Susceptibility matrix: a new aid to software auditing [J]. IEEE Secur Priv. 2004;2(2):16–21.
42. Project Zero. Global trends in 0-day vulnerability exploitation in 2021 [EB/OL]. 2022. <https://www.freebuf.com/vuls/331078.html>
43. Rampage Communications. The number of IoT and networked devices has reached 22 billion, but where are the benefits? [EB/OL]. 2020. <https://baijiahao.baidu.com/s?id=1634289492007203002&wfr=spider&for=pc>
44. Zhang Penghui. Research on computer hardware vulnerability analysis methods [D]. Changsha: National University of Defense Technology; 2016.
45. Moghimi D, Sunar B, Eisenbarth T, et al. TPM-FAIL: TPM meets timing and lattice attacks [EB/OL]. 2019. <https://arxiv.org/abs/1911.05673>
46. Ravichandran J, Na W, Lang J, et al. PACMAN: attacking ARM pointer authentication with speculative execution [C]. In: Proceedings of the 49th annual international symposium on computer architecture. New York, 2022. p. 685–98.
47. Garbelini M, Bedi V, Chatopadhyay S, et al. BrakTooth: causing havoc on bluetooth link manager via directed fuzzing [C]. In: Security symposium, Singapore, 2022. p. 1–18.

48. Lennert W. Glitched on earth by humans: a black-box security evaluation of the SpaceX Starlink user terminal [EB/OL]. 2023. <https://i.blackhat.com/USA-22/Wednesday/US-22-Wouters-Glitched-On-Earth.pdf>
49. Wikipedia. Pentium FDIV bug [EB/OL]. 2016. https://en.wikipedia.org/wiki/Pentium_FDIV_bug
50. Wikipedia. Pentium F00F bug [EB/OL]. 2016. https://en.wikipedia.org/wiki/Pentium_F00F_bug
51. INTEL-SA-00075 [EB/OL]. 2016. <https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00075.html>
52. Lipp M, Schwarz M, Gruss D, et al. Meltdown [EB/OL]. 2018. <https://graz.pure.elsevier.com/en/publications/meltdown-reading-kernel-memory-from-user-space>
53. Kocher P, Genkin D, Gruss D, et al. Spectre attacks: exploiting speculative execution [C]. In: Proceedings of 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, 2019. p. 1–10.
54. Vicarte J, Flanders M, Paccagnella R, et al. Augury: using data memory-dependent prefetchers to leak data at rest [C]. In: Proceedings of 2022 IEEE Symposium on Security and Privacy, New York, 2022. p. 1491–505.
55. People's Daily. Regulations on security protection of critical information infrastructure [EB/OL]. 2022. <https://www.gbjm.gov.cn/n1/2021/0818/c409080-32197544.html>
56. Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets [EB/OL]. 2014. <https://arxiv.org/abs/1406.2661>
57. Yampolskiy R V, Spellchecker M S. Artificial intelligence safety and cybersecurity: a timeline of AI failures [EB/OL]. 2016. <https://arxiv.org/abs/1610.07997>
58. Wu Jiangxing. Development paradigms of cyberspace endogenous safety and security [J]. Sci Sin Inf. 2022;52(02):189–204.
59. Yu Nenghai, Hao Zhuo, Xu Jiajia, et al. Review of cloud computing security [J]. Acta Electron Sin. 2013;41(002):371–81.
60. Wang X, Yin YL, Yu H. Finding collisions in the full SHA-1 [J]. Cryptologia. 2005;3621(10): 17–36.
61. Ahmad I, Kumar T, Liyanage M, et al. Overview of 5G security challenges and solutions [J]. IEEE Commun Stand Mag. 2018;2(1):36–43.
62. Cadar C, Dunbar D, Engler D R. Klee: Unassisted and automatic generation of high-coverage tests for complex systems programs [C]. In: Proceedings of the 8th USENIX conference on Operating systems design and implementation, California, 2008. p. 209–24.
63. Upstream. Upstream security's 2020 global automotive cybersecurity report [EB/OL]. 2020. <https://upstream.auto/upstream-security-global-automotive-cybersecurity-report-2020/>
64. Chhetri M B, Uzunov A, Vo B et al. Self-improving autonomic systems for antifragile cyber defence: challenges and opportunities [C]. In: Proceedings of 2019 IEEE international conference on autonomic computing, Sweden, 2019. p. 18–23.
65. Casti J. Four faces of tomorrow [EB/OL]. <https://www.oecd.org/governance/risk/46890038.pdf>
66. Security Geek. Be wary of backdoor security in IoT devices [EB/OL]. 2022. <https://www.51cto.com/article/650042.html>
67. Boyer RS, Elspas B, Levitt KN. SELECT: a formal system for testing and debugging programs by symbolic execution [J]. ACM SIGPLAN Not. 1975;10(6):234–45.
68. Clarke LA. A program testing system [C]. In: Proceedings of the 1976 annual conference, ACM. Houston Texas, 1976. p. 488–91.
69. Moura DL, Rner N. Satisfiability modulo theories: introduction and applications [J]. Commun ACM. 2011;54(9):69–77.
70. Cadar C, Engler D. Execution generated test cases: how to make systems code crash itself [C]. In: Proceedings of 20th ACM symposium on operating systems principle, 2005. p. 1–14.
71. Godefroid P, Levin MY, Molnar D. SAGE: whitebox fuzzing for security testing [J]. Practice. 2012;55(3):40–4.

72. Cadar C, Dunbar D, Engler D R. Klee: unassisted and automatic generation of high-coverage tests for complex systems programs [C]. In: OSDI. Berkeley, 2008. p. 209–224.
73. Wang F, Shoshitaishvili Y. Angr-the next generation of binary analysis [C]. In: 2017 IEEE cybersecurity development (SecDev). IEEE; 2017. p. 8–9.
74. Yamaguchi F, Maier A, Gascon H, et al. Automatic inference of search patterns for taint-style vulnerabilities [C]. In: 2015 IEEE Symposium on Security and Privacy (SP), 2015. p. 797–812.
75. Litchfield D. Defeating the stack based buffer overflow prevention mechanism of microsoft Windows 2003 server [EB/OL]. 2003. <https://www.semanticscholar.org/paper/Defeating-the-Stack-Based-Buffer-Overflow-Mechanism-Litchfield/69d3d6c113bd78f4b1d214314f4baba6ed72f085>
76. Andersen S, Abella V. Data execution prevention [EB/OL]. 2004. <http://h10032.www1.hp.com/ctg/Manual/c00317515.pdf>
77. Andersen S, Abella V. Part 3: memory protection technologies [EB/OL]. 2009. <http://technet.microsoft.com/en-us/library/bb457155.aspx>

Chapter 2

Cyberspace Endogenous Security and Safety Defense Vision



Since cyber-physical systems are a bridge connecting the physical world, the digital world, and human activities, they are widely used in important industries and fields such as public communication and information services, energy, transportation, water conservancy, finance, public services, e-government, and national defense science and technologies. They are related to all aspects of the modern society, and with the increasing abundance of digital products, smart services in cyberspace are becoming more and more advanced, the number of software and hardware codes is increasing, the complexity is growing exponentially, and the application scale of cyber-physical systems is expanding with each passing day. In this context, the problems of CESS emerge one after another, just like the “Sword of Damocles” hanging over the digital society. Unlike general information systems, once a cyber-physical system is attacked, it will directly or indirectly cause unpredictable and incalculable physical damage. In particular, the cyber-attacks related to vulnerabilities and backdoors have become a threat to the functional safety of all cyber-physical systems or infrastructure containing digital elements. Common CESS problems, like “lingering ghosts,” have permeated almost every corner of cyberspace, deeply endangering the safety of people’s lives and property and bringing unprecedented challenges to human beings in maintaining or ensuring the healthy and sustainable development of the digital society [1].

A large number of cybersecurity incidents have shown that most security threats in cyberspace are caused by external factors such as human attacks and are formed through the interaction between internal causes such as individual or common ESS problems of the target object itself. Unfortunately, so far, the traditional mode of thinking and technical routes of cybersecurity has rarely been able to break away from the traditional pattern of “making up for the weaknesses to the best of one’s ability” [1] that relies on the accumulation of prior knowledge and has even failed to fully recognize that “ESS contradictions in the field of cybersecurity can only evolve, transform or reconcile, but cannot be completely eliminated.” Trying to exhaust vulnerabilities and backdoors by iterative engineering methods such as patching vulnerabilities, sealing backdoors and killing virus Trojans, or expecting

to find and destroy the attack chain in advance by setting up non-systematic or structured defense measures such as honeypots, sandboxes, probes, encryption authentication, intrusion detection, moving targets, or attempting to solve ESS problems by “piling up” various additional protective measures, built-in or embedded hierarchical security monitoring mechanisms (drawing on the idea of biological symbiosis), or threat perception means based on big data/artificial intelligence is often futile or at most gets half the results with double the effort. Homogeneous transfer of contradictions in the same spatial-temporal dimension cannot guarantee that at the philosophical level, fixing security defects or introducing “plug-in” security functions will not introduce new common ESS problems, thus inevitably falling into the logical paradox of homogenization and circular transfer of security contradictions. Then, how can we crack the uncertain threat or destructive effect based on CESS problems (especially unknown attacks) without relying on prior knowledge or freshness of the knowledge base? If we cannot achieve a breakthrough in theory, it will be difficult to realize the transformation of the underlying driving paradigm of digital ecology and the vision of cyber resilience engineering.

To make matters worse, blindly piling up various add-on or plug-in, built-in, or embedded security technologies on the user side or application side cannot fundamentally jump out of the current cybersecurity dilemma, and with the rapid increase of the complexity of application system deployment and the comprehensive investment in digital facilities in the whole life cycle, the stability and effectiveness of the system will continue to be affected by man-made attacks or uncertain disturbances that have never been seen before. Ironically, on the one hand, the design and manufacturing side of digital products “selectively ignores” ESS problems or deliberately evades cybersecurity commitments, which brings “endless troubles and disasters” to the users while transferring cybersecurity responsibilities and risks to them; on the other hand, billions of digital products and a large amount of software codes are introduced every year at an alarming growth rate. Relatively speaking, the limited technical progress of traditional security defense, such as “vulnerability mining, backdoor sealing, virus Trojan killing, and threat perception,” is as powerless as a mantis trying to stop a chariot.

Fortunately, people have finally realized that even software and hardware codes designed with the most stringent security measures cannot completely avoid vulnerability problems, and the responsibility for cybersecurity risks must be transferred to those manufacturers who “selectively ignore” the ESS problems of digital products. Even if they have the freedom to pursue superior functional performance and profitability of their products, they must be forced by law and market access conditions to assume their due social responsibilities when they are unwilling or intentionally neglect to fulfill their safety/security commitments to consumers or infrastructure operators. The cyber world needs more secure digital products, not more security products (whose security logic is not self-consistent). Therefore, in order to transform the underlying driving paradigm of digital ecosystem, we must first transform the cybersecurity paradigm.

Based on the theoretical framework of paradigms, this chapter classifies previous cybersecurity technologies based on paradigms, pointing out that the current

cybersecurity defense cannot effectively deal with artificial or deliberate “known unknown” and “unknown unknown” network attacks at the paradigm level. It puts forward a general deconstruction analysis model for the Impossible Triangle of Defense against Unknown Threats—Incomplete Intersection Principle (IIP), which can qualitatively analyze and explain why the vast majority of existing cybersecurity technologies cannot get rid of the dilemma of relying on prior knowledge. This chapter also puts forward the vision of CESS defense and related systems, mechanisms, and characteristics and creates the thinking perspective, methodology, and practice norms of the CESS defense paradigm.

2.1 Current Security Defense Paradigm Problems

2.1.1 Concept of the Paradigm

The concept and theory of the paradigm were put forward by Thomas Kuhn, a famous American philosopher of science, and systematically elaborated in *The Structure of Scientific Revolutions* published in 1962 [3]. He pointed out that a paradigm is essentially a theoretical system and theoretical framework, within which the theories, rules, and laws are universal and are the coordinates, reference system, and basic way of carrying out scientific researches, establishing a scientific system, and applying scientific ideas. A paradigm has three basic characteristics. First, a paradigm is recognized within a certain scope; second, it is a whole composed of basic laws, theories, applications, and related instruments or equipment. Third, it can contribute to reproducible and successful templates for scientific research and technological development.

The author believes that a paradigm is a world view and methodology in the development of science and technologies and a method for raising and solving problems. Different from Kuhn’s view, although paradigms differ in ways of thinking, theoretical methods, and technical routes, their substitution is not like “the change of dynasty.” Their thinking perspectives and prerequisites are different of course, but they should also be symbiotic in coexistence and inherit each other in iterative development, just as the law of parity conservation in the macroscopic world and the law of parity non-conservation in the microcosmic world, both of which are the basic theories of mankind’s understanding of the world. However, there is no doubt that paradigm changes often occur when the original paradigm cannot solve new problems. Therefore, the evolution history of cyberspace security and safety technologies, just like the history of scientific researches, has both evolutionary and transformative development stages.

Jim Gray, an American scientist who won the Turing Prize in 1998, proposed dividing scientific research into four paradigms in his 2007 speech titled “Revolution of Scientific Methods,” and this paradigm classification has now become a highly accepted consensus [2]. According to Jim Gray, the four paradigms come as follows:

The first paradigm of science is testing/experimentation or measurement, which is mainly characterized by recording and describing natural phenomena. Representative events include chemical elements and electromagnetic phenomena, photoelectric effects, celestial movements, and the discovery of elementary particles. However, these studies were obviously affected by the limitations of the experimental conditions of the time, making it difficult to accomplish a more precise understanding of natural phenomena.

The second paradigm of science is theoretical analysis. Representative results include Newton's three laws, Darwin's biological evolution, Maxwell's electromagnetism, Planck's quantum theory, Einstein's theory of relativity, etc. With the increasing difficulty and cost of verifying theories, the established scientific experiments are no longer adequate.

The third paradigm of science is numerical simulation and emulation, which is to implement a specific calculation on a computer, very similar to a physical experiment. In short, it is to carry out mathematical modeling of experimental objects (the current fashionable terms are digital twin and metaverse) and then use the computer to do simulation and emulation. For example, it is difficult to carry out experimental researches on the laws of celestial movement, numerical weather forecasting, and the effects of nuclear weapons through real physical scenes, but it is subject to the impact of computer algorithms and the speed of computing power improvement.

The fourth paradigm of science is the scientific research based on big data and intelligent computing, which is used to discover new laws hidden in massive, heterogeneous, and diversified data, reveal new mechanisms, and identify the characteristics and behaviors of target objects. This paradigm, also known as the big data and artificial intelligence-driven science (BD/AI for science), is characterized primarily by a move away from a single-minded focus on causality and causal mechanisms and toward a focus on the characteristics of things in themselves or their intrinsic interconnections. However, this statement has not yet achieved academic consensus.

In fact, no matter what kind of development paradigm, its thinking perspective and methodology are unique, and its practice norms have universal significance in a certain range and even over a long period of time. Different paradigms can learn from each other, evolve iteratively, and can be inherited and developed.

2.1.2 Classification of Cybersecurity Paradigms

Although there is no consensus in the field of cybersecurity on whether there has been a "development paradigm" so far, the author believes that cybersecurity has objectively experienced three milestone development stages. This section attempts to use the paradigm classification method to re-examine or interpret the basic characteristics of these three stages of development from the perspective of the thinking level and basic concept.

1. **Functional safety development paradigm based on redundant configuration and majority voting** (referred to as the first paradigm). Its thinking perspective is to solve the problem of physical or logical failure of network terminals, nodes, and network system software and hardware. The assumption is that both physical and logical failures are random and probabilistic and that there is no deliberate disturbance of human factors even if there is uncertainty in software performance. Its theoretical basis is functional reliability and robustness based on statistics [3]. Its methodology is to introduce temporal redundancy for repeated processing, spatial redundancy for primary backup or load sharing, isomorphic or heterogeneous redundancy based on majority voting in critical paths or channels, nodes or components, and architecture. Its practical specification is a resilient functional design based on a redundant architecture that can be quantitatively or semi-quantitatively expressed by means of probabilistic tools, focusing on preventing the impact of uncertain common-mode disturbances. However, with the continuous penetration of digital, networked, and intelligent technologies into the traditional field of functional safety, especially the rise of the software-defined hardware (SDH) technology [4], not only is the simple assumption of random physical failure of hardware difficult to establish but also has the same uncertain errors as software. Moreover, the “known unknown” security threats and “unknown unknown” network attacks based on vulnerabilities and backdoors make the traditional semi-quantitative expression of functional safety (and even cyber resilience, a term that is getting popular in recent years) face severe challenges [5].
2. **Authorization security development paradigm based on encryption and authentication** (referred to as the second paradigm). Its thinking perspective is to ensure, without relying on any prior knowledge, that authorized users operate in compliance by means of authorization management or encryption authentication and use software/hardware facilities or information services or data resources safely [6]. Its precondition is that the encryption and authentication algorithms are mathematically safe and credible, and the host software/hardware execution environment which supports or carries the related algorithms is “trusted.”

Its practical specification is based on the cryptography engineering theory and key distribution method to manage the privacy, integrity, and availability of network facilities, information services, and data resources. In terms of engineering modes, the protected core resources of cyber-physical systems or digital devices need to be attached with related encryption and authentication software and hardware facilities. The challenge it faces is the “internal-external collaborated” cyberattacks caused by the vulnerabilities and backdoors in the “trusted execution environment” carrying the encryption authentication algorithm. The “perfect secrecy” of “one-time pad” put forward by Shannon’s information theory is difficult to be established in engineering without a “trusted execution environment” (see Chap. 3 for details).
3. **Cybersecurity development paradigm based on detection and analysis** (referred to as the third paradigm). There have been three stages with different focuses in history, namely, the virus Trojan killing stage, the software/hardware

vulnerability discovery and repair stage, and the attack behavior feature perception and blocking stage. The focuses of different stages are, respectively, (1) detecting and removing malicious codes inserted/implanted in cyberspace terminals, nodes, or system software based on knowledge base and updating the malicious code knowledge base regularly and iteratively; (2) patching the security defects found in the target system's software/hardware code design, establishing a vulnerability library and the corresponding query and storage mechanism, and introducing different levels and categories of active/passive defense technologies to disrupt the injection of virus Trojans or reduce the exploitability of unknown vulnerabilities and backdoors in the target object; (3) adopting precise attack behavior perception and characteristic profiling technologies to discover or reject network attacks through on-site analysis means such as sandboxes, honeypots, firewalls, and intrusion perception and non-real-time means such as intelligent detection (including big data and artificial intelligence) based on iterative knowledge base, so as to perceive or warn threats and attacks as early as possible, increase the difficulty of establishing attack chains, or reduce the reliability of attack chains.

The theoretical and technical prerequisites for the above three stages are, respectively, the following: (1) The problem codes such as virus Trojans and their operating characteristics are known or about which there is prior knowledge or inferences and assumptions about their assumed characteristics can be made. (2) The possible attack range or attack surface and exploitable attack resources are static, deterministic, and similar. (3) The behavioral characteristics of the attacker and all legal operations and compliance behaviors of the protected object are clearly understood.

The corresponding practice specifications are as follows:

1. Trying to automatically detect and delete malicious codes, alarm/isolate it or request manual intervention, introduce intrusion detection and tolerance technology, and establish and iterate virus Trojans or malicious behavior signature libraries
2. Establishing and keeping the improvement of a knowledge base of vulnerabilities and backdoors, formulating software/hardware code security design and engineering specifications, developing vulnerability analysis and positioning technologies, proactively exploring or discovering vulnerabilities, and trying to reduce the accessibility of attack surfaces or the exploitability of attack resources
3. Using real-time or non-real-time means such as built-in probes, honeypots, sandboxes, and operation logs to collect data of suspected problem scenes as much as possible; establishing a global threat perception system and detecting, suppressing, or preventing possible attacks with the help of new technologies such as black/white lists; and using new technologies such as black/white lists, big data, and artificial intelligence to discover, suppress, or prevent possible attacks [7].

The common principle is that additional or external detection or protection facilities are needed for the protected system.

The common problem of the above three paradigms of cybersecurity development is how to effectively cope with unknown cyber-attacks based on “internal-external collaboration,” unknown vulnerabilities, backdoors, and virus Trojans and how to provide GFS performance that can be quantifiably designed and verifiably measured.

In recent years, people have gradually realized that under increasingly advanced APT (including AI-based) attacks, it is impossible to achieve “foolproof” defense, and it is difficult to stick to the bottom line that “the system will not be compromised.” As a result, the focus of the work is once again locked in the direction of strengthening “mission assurance,” and it is proposed that the digital system itself should not only be able to cope with random or uncertain disturbances but also have the ability to perceive any potential threats and sabotage and that the digital system should have not only resilient functionality under non-human disturbances but also “cyber resilience” under network attacks. However, to achieve this goal, if there is no breakthrough in basic theory, engineering technologies will still be unable to cope with “unknown unknown” ESS defects. This book will focus on this problem and propose targeted solutions.

2.1.3 Defense Paradigms Need to Be Urgently Changed

Since CESS problems, especially common ESS problems, have become the biggest “public hazard” plaguing the digital society in the network age, it is not an exaggeration to call them the “source of trouble” in the governance of a digital society. However, the current mainstream cybersecurity defense paradigm seems powerless in this regard.

2.1.3.1 Outdated Thinking Perspective

In addition to the protection measures of encryption and authentication, what is mainly adopted so far has been “remedial” passive defense based on the accumulation of prior knowledge such as threat perception and feature extraction (including active defense such as honeypots, sandboxes, and intrusion detection). Although effective security protection can be provided against known or even “known unknown” security threats (the risks that may occur are known, but their timing and possible degree of harm that are unknown), for “unknown unknown attacks” (what attacks will occur is unknown, and their timing and scope of impact are also unknown), which are a common ESS problem of digital products, especially “internal-external collaboration” network attacks based on unknown backdoors, the current technical route of “pursuing zero problems” from the thinking perspective of “knowing what it is and why it is so” is impossible to achieve logical

self-consistency at the theoretical level, not to mention whether it can exhaust all unknown threats and sabotages in engineering practice.

2.1.3.2 Methodological Confinement

The current methodology of “patching” or “additional defense” based on various knowledge bases attempts to use “additional” security protection or “bodyguard-like” services or “debugging” means through continuous evolution and iteration at the engineering and technical level to reduce or even completely eliminate the harm of network attacks based on vulnerabilities and backdoors. Neither in theory nor in practice can this methodology completely eliminate all the potential security defects of the “additional” defense facilities themselves. Moreover, the new common ESS problems that appear in patching or after patching may also cause unexpected security threats or damage. The methodology of “patching” aims at cyclic migration of security contradictions within the physical or logical space of the same target object and often falls into the dilemma of patching a known vulnerability and then generating several new unknown vulnerabilities. The nature of contradiction or even its expression space has not changed, which obviously goes against the philosophical principle of the evolution, transformation, and reconciliation of contradictions. Even if active security technologies such as Moving Target Defense (MTD) [8] are introduced with dynamic (random), diversified (heterogeneous), and redundant elements, there is still no way to deal with unknown network threats or attacks based on internal-external collaboration. It should be pointed out that although “perfect secrecy” based on cryptographic technologies can provide security protection that does not rely on prior knowledge against non-internal-external collaboration network attacks, it is still unable to solve the problems about “perfect security” including the multi-intertwined problems of functional safety, network security, and even information security (see Chap. 3 for details).

2.1.3.3 Non-quantifiable Security

Because add-on or plug-in defense specifications are based on dealing with known or “known unknown” threats, the technical goal of resilience engineering that is characterized by “patching vulnerabilities” must not only accurately perceive the attack behavior but also fully understand how the attack chain is established, what hardware and software resources are used, what the attack mechanism is, how to locate the problem code segment, whether the whole attack scene can be reproduced, and other known conditions necessary for “patching” engineering, as well as the effectiveness evaluation method in engineering implementation and regression test of related functions of the system. In this process, it is not always possible to “know what is happening and why it is happening,” so “a lost sheep may not necessarily lead to a mended sheepfold,” “a fall into the ditch may not necessarily make a person wiser,” and “doing one’s best” can only be one of the options in various kinds of

helpless situations. It is impossible for such defense specifications to obtain any substantial security defense effect against unknown network attacks, let alone provide quantifiable design and verifiable, measurable security metrics.

This triggered a series of scientific problems. For example, how can we prove that the “patch-after-patch” and “patch-on-patch” types of add-on security protection are safe and trustworthy? How can we prove that each “patch” itself is safe and trustworthy? Due to the universality of common ESS problems, how can existing security technologies prove their innocence? How can we prove that the “trusted execution environment” required for confidential or trusted computing is trustworthy? [7]

Both the theory and practice show that although some ESS contradictions cannot be eliminated, they may be reconciled or reach a certain degree of compromise or form a certain unity of opposites. For example, strict physical isolation of the network (including cutting off all possible physical external communication channels such as sound, light, electricity, magnetism, and infrared) makes it impossible for an external network attack to trigger security problems such as unknown backdoors and vulnerabilities in the internal network, that is, to achieve the effect that “external attacks are impossible to access the isolated network or resources under the attack surface are not exploitable.” However, mainstream additional security defense measures such as “backdoor sealing,” “active detection,” and “encryption authentication” always make ESS problems or contradictions cyclically migrate within the same target object, the same spatial-temporal relationship, and the same resource environment, with the invariability of “adding more water when flour is in excess and adding more flour when water is in excess,” or use basic security (or reliability) elements such as randomness, variety/diversity, and dynamics in a non-integrated manner to implement costly “Moving Targets Defense (MTD)” [8]. Even so, these technical arrangements can neither fundamentally deal with “unknown unknown” security threats (especially “internal-external collaborative” network attacks), nor can they avoid the increasing defense cost, the result of which is a continuous deterioration of the target system’s service functionality and technical-economic performance throughout its life cycle.

2.2 Impossible Triangle of Defense Against Unknown Threats

Generally speaking, the so-called impossible triangle is essentially a universal ternary deconstruction analysis model, which can be widely used to qualitatively analyze the accessibility of core functional indicators of anything or any target object. In layman’s terms, the deconstruction model of “impossible triangle” points out that it is impossible to achieve all the goals or meet all the needs through “not only...but also...and...” without making any compromise or giving up part of the

benefits [9]. Venn diagrams and ternary intersection formulas are often used to express the intuitive graphics and calculations of element sets and their relationships.

With the help of the above tools, this section creates an “Impossible Triangle” deconstruction model (or Venn diagram expression) in the field of cybersecurity, aiming at whether defense against unknown security threats or breaches (hereinafter referred to as defense against unknown threats) is effective, so as to reveal why cybersecurity technologies so far cannot effectively defend against internal-external collaborative network attacks based on “unknown unknowns.” The basic method is to qualitatively analyze the technical significance of the three elements of security defense, namely, dynamics/randomness, variety/heterogeneity, and redundancy, to defend against unknown threats. Next, taking these three basic properties as the vertex elements D, V, and R of the “Impossible Triangle,” we qualitatively analyze the role of the three vertex angles (including pairwise intersection) in defending against unknown threats and find out what the main problems are. Then, further observation and analysis can lead to a conclusion: as long as the three elements are in a triangular structural relationship, they will not be able to defend against unknown threats. In other words, we can intuitively obtain the conjecture of whether there is ESS in the “DVR intersection.” That is to say, there may be some kind of “structure determining security” intersection relationship between the three elements, so let’s see if we can get the system engineering gain of “ $1 + 1 + 1 > 3$.” In order to facilitate the understanding of this method, we first introduce it from Krugman’s impossible triangle principle and the impossible triangle theorem of distributed systems, then put forward the incomplete intersection principle (IIP), and conduct qualitative analysis combined with representative cybersecurity defense technologies.

2.2.1 *Krugman’s Impossible Triangle*

The concept of the impossible triangle/impossible trinity theory originated from an economic concept [10]. It was founded in 1999 by Paul Krugman, a professor of economics at MIT, based on the Mundell-Fleming model and the empirical analysis of the Asian financial crisis, as shown in Fig. 2.1. The three vertices of the triangle represent the free capital flow, monetary policy independence, and exchange rate stability, respectively.

Its basic connotation is that “a country cannot simultaneously achieve free capital flow, independent monetary policy and exchange rate stability at the same time.” In other words, a country can only have two of them, but not all three at the same time. If a country wants to allow free flow of capital while demanding an independent monetary policy, it will be difficult to maintain exchange rate stability. If exchange rate stability and capital mobility are required, then independent monetary policies must be given up [10, 11]. The Mundell-Fleming model requires that the two most important conditions should be in place: first, the monetary policy must be independent of the fiscal policy; second, the country must have well-developed capital and money markets.

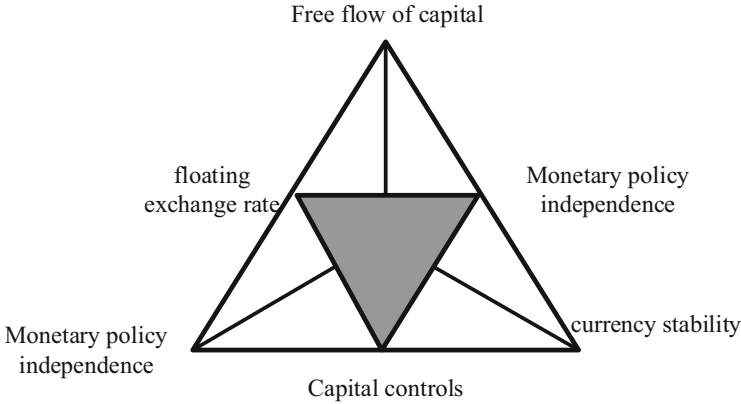


Fig. 2.1 Krugman's impossible triangle

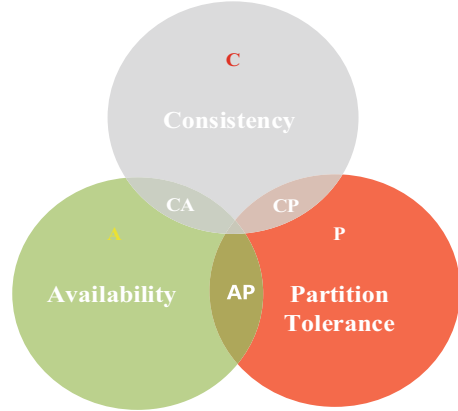
According to the “impossible triangle,” a country can only choose three policy combinations between free capital flow, monetary policy independence, and exchange rate stability. First, to maintain free capital flow and monetary policy independence, exchange rate stability must be sacrificed; second, to maintain exchange rate stability and monetary policy independence, capital flow must be restricted; third, to maintain free capital flow and exchange rate stability, monetary policy independence must be given up.

The limitation of the above theory is that it must be based on strict assumptions, that is, complete free capital flow, complete currency independence, and complete exchange rate stability, and at the same time it also implicitly assumes that such a country must have developed capital and money markets, which is obviously not true for most countries [11].

2.2.2 Brewer's Impossible Triangle

Brewer's impossible triangle, also known as the impossible triangle of distributed systems or CAP (consistency, availability, partition tolerance) theorem or principles, is a conjecture put forward by Eric Brewer at the PODC (Principles of Distributed Computing) Seminar organized by ACM in 2000 based on his own engineering practice (as shown in Fig. 2.2). Two years later, Seth Gilbert and Nancy Lynch of Massachusetts Institute of Technology proved it theoretically [19], and it has since become an accepted theorem in the field of distributed computing. Two years later, Seth Gilbert and Nancy Lin of MIT Scholars such as Nancy Lynch theoretically proved it, and since then it has been a recognized theorem in the field of distributed computing.

Fig. 2.2 Impossible triangle of distributed systems (Venn diagram)



“CAP” in the term of the CAP impossible triangle theorem is taken from the first letters of three English words or phrases: consistency, availability, and partition tolerance.

1. “Consistency” means that in a distributed system with m nodes, all nodes have the same data at the same time.
2. “Availability” means that when m nodes are all available, they can all respond to service requests within a given time.
3. “Partition tolerance” refers to fault tolerance, that is, the loss of any information in the system or the failure of message delivery or the downtime of nodes will not affect the continuous operation of the system.

The CAP theorem points out that in a strict temporal sense, C, A, and P cannot exist at the same time in a distributed system. In other words, consistency (C), availability (A), and partition tolerance (P) cannot be obtained simultaneously in a distributed system, and only two of them can be chosen at most. Qualitative analysis is as follows:

1. Why can’t we realize C when having A and P? Assuming that there are m available nodes in a distributed system (i.e., P is guaranteed), if we want to back up data, we need to back up the same data on all m nodes to maintain data consistency, which will consume the processing time of nodes. Therefore, service availability cannot be guaranteed during the data backup time. In other words, if service requests can be responded to within a given time, there will be no guarantee that the data on all nodes will always be consistent.
2. Why can’t we realize A when having C and P? In a distributed system with m nodes, if data consistency is to be maintained at all times, the requirement for service response time cannot be met all the time, that is, availability cannot be guaranteed.
3. Why is it meaningless to have C and A? On the one hand, the requirement for always ensuring data consistency on m nodes is contradictory to the requirement for ensuring that service requests can be responded to within a given time; on the

- other hand, a distributed system without fault tolerance makes no sense in engineering practice.
4. Why is P necessary in distributed systems? According to the current software and hardware capabilities, there must be network message packet loss or transmission errors and node downtime. However, a distributed system is required to provide consistent and available services when a node or network partition fails, so fault tolerance P becomes a necessity.
 5. Why is it OK to choose either C or A in the distributed system? The reason is that the combination of A and P does not guarantee that the data on m nodes are always consistent at any time, while the combination of C and P cannot guarantee that the service response delay is the same all the time.

Engineering significance of the CAP theorem: Compared with the previous simple conclusion that “low delay and sequential consistency cannot be satisfied at the same time” [12], CAP analysis is more specific and helps engineers carefully consider the performance indicators of key functions when designing a distributed system, rather than wasting time to build an ideal system.

2.2.3 Three Elements of CyberSecurity Defense

Figure 2.3 shows the network attack chain. Obviously, the effectiveness and reliability of the attack chain are very dependent on the static nature, certainty, and similarity of the target system, where a change in any link may prevent the network attack from achieving the expected purpose. In other words, the attack chain is actually very fragile. For example, introducing random, dynamic, and diverse mechanisms in the stages of system detection, vulnerability mining, and system breakthrough can effectively combat covert penetration; introducing heterogeneous redundant majority-based ruling mechanisms in the stages of system control, information acquisition, and system damage can deal with latent or dormant attacks. It

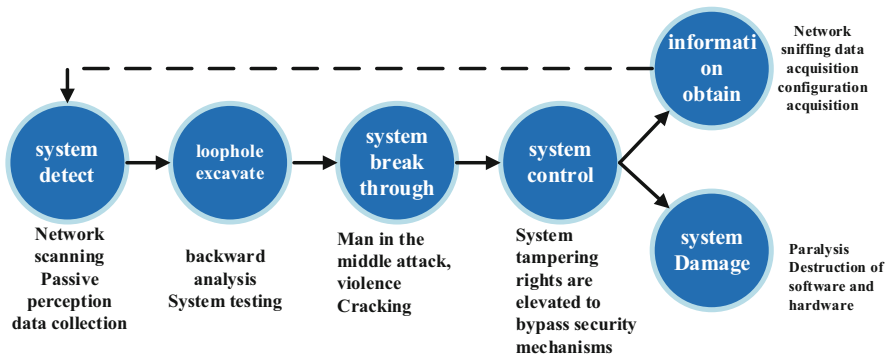


Fig. 2.3 A typical network attack chain

can be seen that dynamics/randomness, variety/heterogeneity, and redundancy are the core elements of defense (including resilience/toughness). We should not only study the defensive function of each element but also be clear about the security nature, application premise, and possible defects or deficiencies of various combinations of these elements and recognize what kinds of faults or attacks can be effectively prevented by various current security technologies under what conditions; whether these three elements are enough to solve the defense problem of unknown security threats; whether emergent properties' security defense gains can be obtained through structural effects; and whether the impossible triangle's general deconstruction analysis tools can be used for qualitative analysis in the cybersecurity field. In the final analysis, it is necessary to find out whether there is an endogenous "antibody" in cyberspace. In other words, the question of whether the concept of CESS can be established in theory should be answered in the end.

2.2.3.1 Dynamics Brings Uncertainty

On the one hand, the certain, static, and similar nature of the traditional information system or cyber-physical system (CPS) gives attackers temporal and spatial advantages, making it possible for them to repeatedly sniff and analyze the ESS problems of the target system and even find a way to break through the defense of the target system by means of social engineering. On the other hand, it gives attackers an asymmetric cost advantage over the defender, allowing them to use the same attack means to act on similar or identical target systems on a large scale. One of the most representative dynamic defense technologies is "Moving Target Defense (MTD)" proposed in the Networking and Information Technology Research and Development (NITRD) Program of the United States, which aims to increase the difficulty and cost of attackers and limit the exposure of vulnerabilities and the chance of successful attacks through various constantly changing construction, evaluation, and deployment mechanisms and strategies [8]. In other words, it aims to increase the uncertainty of the target object from the attacker's perspective as much as possible. MTD can be implemented in different ways at different levels of the target object and can be roughly divided into five categories: dynamic operating environment, dynamic software, dynamic network, dynamic data, and dynamic platform. Its core technologies include variable IP address, variable number of channels, variable routing security and IP protocol, randomness of network and host identity, randomness of address space, randomness of instruction set, randomness of data, etc.

It should be noted that essentially and theoretically, random phenomena are a kind of unpredictable dynamic movement with no rules to follow. That is to say, repeated experiments under the same condition may or may not produce the same result, or even if we know the past of a thing, we cannot be completely sure about its future. As in the game of tossing a coin, we know that each toss will result in either heads or tails, but there is no way to expect that a particular toss will definitely result in heads or tails. Randomness is an ideal representation of dynamics. In fact, the idea of dynamic defense has been around since ancient times. *Sun Tzu's Art of War – The*

Use of Spies states: “In war, those who are good at attacking will attack where the enemy’s defenses are weak, and those who are good at defending will protect areas that are vulnerable to enemy attacks.” This statement emphasizes the importance of dynamic change in controlling the battlefield.

2.2.3.2 Diversity Brings Dissimilarity

In a philosophical sense, diversity refers to “the plurality of beings, or the manifoldness of reality.” Empedocles (490–430 BC) and others advocate a kind of qualitative pluralism, believing that “beings” are not monistic but pluralistic and that there are qualitative differences between the elements of existence.

The engineering community believes that a function can be realized through a variety of technical methods and expressed through a variety of hosts or carriers. This is what is meant by the saying “All roads lead to Rome.”

In the biological world, there is a fog of diversity, also known as mimic disguise. In 1998, Mark Norman of the University of Melbourne, Australia, discovered a new species of octopus in the waters of Sulawesi, Indonesia. Its scientific name is striped octopus, also known as the mimic octopus. It is the top mimicry camouflage master that has been discovered in the biological world so far. Research shows that it only actively changes its body color and texture but also imitates the shape and behavior of other creatures, making it completely invisible on the gravel seabed and in the coral reef environment. Without relying on additional camouflage, the striped octopus can at least simulate 15 kinds of sea creatures. It takes structurally similar objects as references; conceals its external appearance (including morphology and behavior) through emulation or simulation of color, texture, appearance, and behavior; and creates cognitive dilemmas or cognitive misunderstandings for predatory objects with apparent features or functions, so as to gain survival advantages and safety [1].

In short, the diverse manifestations of things often cause observers to fall into cognitive dilemmas. Similarly, the apparently diverse expressions of target objects can cause great perceptual or cognitive obstacles to cyberattackers. Therefore, the use of diversity mechanisms is to change the similarity of the target system environment, so that attackers cannot locate and exploit attack resources such as vulnerabilities, backdoors, and virus Trojans or directly apply or replicate certain successful attack experience to the attacks in similar target scenarios.

It is important to emphasize that, in general, an important difference between plurality and diversity is functional equivalence but not homology. Usually, diversity refers to multiple manifestations of the same entity, also known as multiple variants of the same entity. There are many similarities or common causes between the variants. The mimicry phenomenon in the biological world is a classic representation of this form. Although there are equivalent functions between diverse entities, there are few or even no homologous components in the same elements. For example, the Windows operating system and the Linux operating system have almost no similar vulnerabilities [13]. Diversity and plurality are ultimately

manifested in the dissimilarity in the implementation algorithm of component elements. For attackers, it is much more difficult to find identical or common-mode vulnerabilities among multiple variants than to find homologous vulnerabilities among multiple variants.

2.2.3.3 Redundancy Brings Reliability

Redundancy in the field of engineering technologies refers to the use of more than one approach or method to complete a specified function or to repeatedly perform the same operation in different time periods. Spatial redundancy design, also known as redundant design, is generally to add more than one set of channels and working components that can complete the same function at the place or link where the system or equipment plays a key role in completing the task. The purpose of redundant design is to ensure that the system or equipment can still work normally or continue to provide services when this part has errors or failures, reduce the probability that the system will enter a state of failing to fulfill its mission, and improve the reliability of the system or its components. In this sense, redundant design is both a fault-tolerant design and a technical method of “structure determining safety.” Its application scenario has the following characteristics:

1. Scenarios in which system mission reliability requirements cannot be met by conventional methods such as improving the quality and basic reliability of components or materials (commonly known as ceiling scenarios)
2. Scenarios when it is difficult to accurately estimate and verify system mission reliability due to the use of new materials, new processes, or use in unknown environmental conditions
3. Links or key parts decisive for the success or failure of system missions
4. Scenarios where the correctness of the processing results of a complex system cannot be determined and majority-based ruling is made by using the processing results of multiple functionally equivalent systems or the same system in different time periods

All in all, redundant design not only improves the reliability of “unknown unknown” scenarios but also obtains the recognizability of “unknown” differential-mode scenarios through the relativity ruling mechanism under certain conditions.

In the fields of functional safety, cybersecurity, and even information security, the three elements of dynamics (D), variety (V), and redundancy (R) are often applied in combination, but so far, no application example of the integrated combination of the three elements has been found in technical practice, perhaps because it is difficult to directly realize the complete intersection of D, V, and R in engineering.

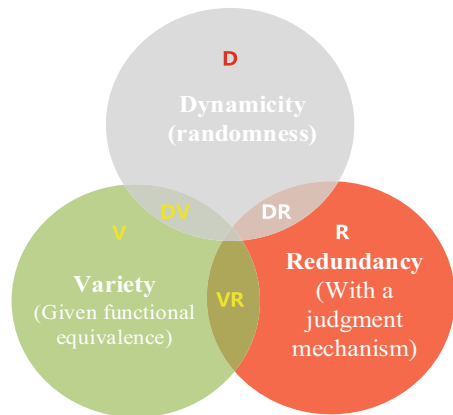
2.2.4 *Incomplete Intersection Principle (IIP)*

As mentioned above, there are three core technical elements (D, V, and R) in the field of cybersecurity defense, namely, dynamics/randomness, variety/heterogeneity, and redundancy, which are crucial for increasing uncertainty and guarding against unknown security threats or breaches. Among them, randomness can be regarded as a special case of dynamics. Likewise, heterogeneity can be considered a special case of variety. Unfortunately, the functional safety, cybersecurity, and information security industries have long lacked a systematic theoretical understanding of how to utilize the combination of the three elements to defend against unknown threats. They do not know what combinations are sufficiently necessary for obtaining ESS, nor do they understand what logical structures are necessary for obtaining ESS functionality that does not rely on (but does not exclude) prior knowledge. In this subsection, the author will use the “Impossible Triangle” universal deconstruction analysis model based on Venn diagrams and ternary intersection expressions to conduct a ternary deconstruction analysis of the above-mentioned scientific problems and apply the impossible triangle of protection against unknown threats—incomplete intersection principle (IIP) to explore why existing cybersecurity technologies can neither effectively respond to human and non-human disturbances nor reject “known unknown” or “unknown unknown” network attacks in real time.

Incomplete intersection principle (IIP): If there is no intersection between D, V, and R, or there are only intersections between any two of them, then it is impossible to deal with internal-external collaborative network attacks based on unknown vulnerabilities and backdoors inside the target object. In other words, without relying on prior knowledge, as long as the three elements of D, V, and R cannot completely intersect, it is impossible to prevent unknown security threats in cyberspace. Figure 2.4 shows the DVR ternary deconstruction model (Venn diagram).

Based on IIP, the following is a qualitative analysis of relevant security technologies:

Fig. 2.4 DVR ternary deconstruction model



2.2.4.1 Situation Where There Is No Intersection Between D, V, and R

When there is no intersection between D, V, and R:

D domain: Although the dynamics of only a single target object can bring apparent uncertainty, it has no security effect on preventing internal-external collaborative attacks based on unknown vulnerabilities, backdoors, or virus Trojans in the target object.

R domain: Simple redundancy or isomorphic redundancy is mechanically ineffective in preventing attacks, common-cause failures, and trial-and-error attacks based on unknown common-mode vulnerabilities and backdoors.

V domain: Without a dynamic scheduling mechanism, variety cannot provide security protection.

2.2.4.2 Situations Where There Are Only Intersections Between Any Two of D, V, and R

In these cases, there are only $D \cap V$, $D \cap R$, and $V \cap R$ intersections. The qualitative analysis is as follows:

1. $D \cap V$ intersection. Typical technical practices include Moving Target Defense (MTD [3]).
 - (a) Prerequisites:
 - There are diverse target objects with functional equivalence.
 - Security problems such as vulnerabilities and backdoors in diverse target objects are unknown to defenders.
 - (b) Defense principle: Randomly selecting functionally equivalent target objects to provide current external service functions.
 - (c) Defense effect: Increasing the attacker's uncertainty in exploiting known vulnerabilities or detecting unknown vulnerabilities.
 - (d) Main problems:
 - As long as there is a backdoor unknown to the defender in diverse target objects, the attacker can at least achieve the goal of stealing or tampering with sensitive information through internal-external collaboration.
 - Theoretically, the difficulty of vulnerability detection or exploitation is related to the number of target objects that can be exploited, the scheduling algorithm, and the intensity of scene changes. This means that high system overhead is required to provide sufficient uncertainty, which is often unbearable in engineering practice.
2. $D \cap R$ intersection. The typical technology has dynamic isomorphic redundancy, which is also a traditional method to break through the “ceiling” of reliability.

(a) Prerequisite and working principle:

- Only for random failures or uncertain errors caused by natural factors.
- The target object consists of n redundant bodies with exactly the same structure and function.
- The ruling or consensus mechanism can distinguish and mask redundant bodies with differential-mode performance.
- The dynamic scheduling mechanism can switch or remove redundant bodies with differential-mode performance.

(b) Defense effect: Under the condition of $f \leq (n - 1)/2$ (n is the number of redundancies, and f is the number of redundant bodies that allow differential-mode problems to exist at the same time), all the redundant bodies with differential-mode performance in the target object can be masked.

(c) Problems: The effectiveness of defense against cyber-attacks or common-cause failures based on unknown common-mode vulnerabilities and backdoors in most redundant bodies (although the probability of occurrence is low under normal conditions) cannot be established mechanistically.

3. $V \cap R$ intersection. The typical technology is dissimilar redundancy structure (DRS [3]), which is also known as heterogeneous redundancy (HR) in the field of reliability.

(a) Prerequisites:

- The target object consists of multiple functionally equivalent non-similar or heterogeneous redundant bodies.
- Vulnerabilities and backdoors in redundant bodies are black boxes for both attackers and defenders.
- The number of redundant bodies that cannot be used at any time is $f \leq (n - 1)/2$ (n is the total number of redundant bodies).

(b) Defense principle: Multiple heterogeneous redundant bodies work in parallel with a majority-based ruling mechanism. Whenever a redundant body with differential-mode performance is found, regardless of the original cause for the problem and the mechanism of its occurrence, it will be blocked without interrupting the current service.

(c) Defense effect: Compared with isomorphic redundancy, the probability of common-cause failure is much lower.

(d) Existing problems:

- Since the number of heterogeneous redundant bodies and deployment plans of the target system are static, deterministic, and similar, the attack surface is also deterministic and similar.
- If there are vulnerabilities and backdoors known to the attacker but unknown to the defender distributed on multiple redundant bodies, the attacker can use the “blast” method one by one, so that the number of currently available redundant bodies cannot meet the minimum conditions

of majority-based ruling, and the system can only go into a downtime state; the attacker can also use the “penetration” method to realize the “hibernation standby” attack of arbitrary functions. In other words, $V \cap R$ has neither stability robustness nor quality robustness under attacks based on backdoors and vulnerabilities unknown to the defender [14].

The above analysis shows that in a defense system, if any one of the elements of dynamics (D), variety (V), and redundancy (R) is missing, or if the three elements are always in an incompletely intersecting state, then the system will not be able to defend against internal-external collaborative attacks based on internal unknown security problems.

It is not difficult to infer that IIP, as a general ternary deconstruction analysis tool, can qualitatively examine whether any target object or technical product has the GFS capability to prevent unknown security threats without relying on prior knowledge, especially whether it has the ability to deal with internal-external collaborative network attacks based on the ESS problems of the target object.

2.2.5 Examples of IIP Analysis of Existing Security Technologies

2.2.5.1 Encryption/Authentication

Any encryption/authentication algorithm belongs to a subset of the DVR domain. Although the algorithm itself can be static, the key must be dynamic, following the theoretical principles of “one-time pad” and “perfect secrecy” and not relying on prior knowledge. In addition, multiple groups of keys are heterogeneous and redundant, following the principle of randomness as much as possible. Therefore, a good encryption algorithm itself should have ESS attributes. It needs to be emphasized that although the security of the encryption/authentication algorithm itself does not rely on any prior knowledge, it does not mean that its implementation or execution carrier or trusted execution environment (TEE) also has ESS attributes. The common ESS problems of host software and hardware are often the biggest hidden trouble of the encryption/authentication algorithm. For example, no advanced algorithm can cope with internal-external collaborative “bypass” attacks based on vulnerabilities and backdoors in the execution environment.

2.2.5.2 Blockchain Technology

The blockchain technology based on distributed, decentralized, and consensus mechanisms also belongs to a subset of the DVR domain in terms of the algorithm itself. In theory, the number of nodes in the blockchain is not fixed, allowing dynamic addition or exit, and there is neither homogeneity nor diversity

requirements for nodes. A consensus mechanism based on redundancy is adopted among distributed nodes. Therefore, the blockchain as an endogenously safe and secure algorithm may be established in theory, but it is difficult to ensure that the software and hardware host systems of each node of the blockchain also have ESS in practice. For example, if more than 51% of nodes use the same x86 or ARM architecture CPU, or operating systems such as Windows and Linux, or various derived software formed by some open-source community and same-source codes, then the common ESS problems in these software and hardware may theoretically lead to the ineffectiveness of the Byzantine General-type consensus algorithm. In addition, the impossible triangle—CAP principle of distributed systems—is also a big challenge to the large-scale application of the blockchain.

2.2.5.3 Zero Trust Technology

The zero trust technology has attracted much attention since its birth and is even considered a disruptively innovative concept in the field of cybersecurity. In essence, zero trust as a method, an architecture, a strategy, a systematic arrangement, and a continuous security protection goal based on a prior knowledge base, also belongs to a subset of the DVR domain. Since the technical architecture and deployment strategy are static, similar, and deterministic once determined, there is no redundancy in theory although the implementation of trust authentication links at all levels does not exclude diversity or heterogeneity. Therefore, the lack of D and R elements makes its effectiveness strongly related to the richness of prior knowledge, and it is mechanically impossible to deal with “unknown unknown” security threats or damage. In addition, if there are common ESS problems in the software and hardware host system of each certification link, especially in the case of using single-source hardware and software components, an unknown attack targeting a hidden or unknown same-source vulnerability or backdoor may severely damage the zero-trust security system that seems to be fortified layer by layer but expressed in fragments. For example, when most links use the same CPU or operation with backdoors and vulnerabilities unknown to the defender, the common-mode security problems cannot be eliminated.

2.2.5.4 Trusted Computing/Confidential Calculation

Theoretically, trusted computing/confidential calculation is not a technical example in the DVR domain. Because its effectiveness is predicated on having a complete or relatively complete knowledge base, an indispensable software and hardware certificate authentication mechanism and a private system startup and loading mechanism, it cannot deal with “unknown unknown” security threats. In addition, the redundant element R does not have functional equivalence, and it also lacks the dynamic element D and the diversity element V. More embarrassingly, since the birth of this technology, there has been an unavoidable “soul torture”—how to prove that

there are no common ESS problems in the trusted root and its related control system or the TEE of software and hardware. In other words, it extremely relies on the trustworthiness and dependability of the Enclave.

2.2.5.5 Various Additional, Built-in, Embedded, or Intrinsically Safe Traditional Technologies

Except for encryption technologies, the effectiveness of almost all traditional security technologies relies on the accumulation of prior knowledge and theoretically cannot deal with “unknown unknown” threats or attacks. Therefore, these technologies are not problems that need to be discussed in the DVR domain. In terms of the implementation of relevant software and hardware technologies, neither theories nor technologies so far can be logically self-consistent as for how to ensure that there are no common and individual ESS problems.

In a word, IIP helps to qualitatively analyze whether various security algorithms (concepts, schemes) and implementation carriers (hardware and software technologies) can effectively deal with security threats or internal-external collaborative attacks based on “known unknowns” or “unknown unknowns” without relying on prior knowledge. It needs to be re-emphasized that even if an algorithm or method has ESS, it does not mean that its software and hardware implementation carrier/host has the ability to defend against unknown threats or damage. These are two different categories of problems.

Undoubtedly, IIP is the only criterion to determine whether a target object has ESS or not, and it gives us a reliable deconstruction analysis method to distinguish various types of digital products labeled as being endogenously safe and secure.

2.3 The Vision of Cyberspace Endogenous Security and Safety Defense

As a natural inference, if we can change the thinking perspective of the current cybersecurity defense paradigm from the passive defense paradigm of “mending the sheepfold after a sheep is lost” to one that does not rely on (but does not exclude) previous knowledge or can be aimed at “unknown” threats and sabotage, from the “patch-after-patch” methodology to a system engineering theory based on the principle of “structure determining security” or “mimic disguise,” and from the practice of “doing our best” to the engineering goal of quantifiable design and verifiable measurement of security, then we may completely get rid of the troubles of various vulnerabilities, backdoors, and other security problems and adopt new methodologies and practice specifications in the newly created cognitive space and problem domain to quantitatively solve the multi-intertwined common ESS problems of functional safety, network security, and information security, fundamentally

break through the confinement of the current cybersecurity defense paradigm, and acquire both non-specific and specific immunity like vertebrates in the biological world [15]. Such immunity not only provides the “surface defense” capability independent of prior knowledge but also has the ability of “point defense” based on the precise identification of antigens, i.e., providing protogenetic or endogenous security. It is also expected that cyber-physical systems or digital products can obtain ESS gains based on system architecture, so as to achieve the ideal cyber resilience engineering goal of making the systems or products as strong as “reinforced concrete.” Taking the “creation of a trusted execution environment coexisting with ESS problems in cyberspace with genetic defects” as a prerequisite, we can solve the multi-intertwined problems of network security, functional safety, and even information/data security in the underlying driving environment of the digital ecosystem in a well-structured way, which is called “Cyberspace Endogenous Security and Security (CESS).” In subsequent sections and chapters of this book, we will refer to it simply as “CESS” or “ESS” for short, unless otherwise stated.

2.3.1 Desired ESS Structure

The ESS vision is similar to the communication technology of “providing reliable transmission services in noisy channels,” creating a trusted execution environment in cyberspace with genetic defects, and making digital products obtain integrated security gains based on the structure through the change of the underlying driving paradigm of digital ecosystem. To this end, we have the following expectations for the properties or attributes that ESS structures should have:

1. The ESS structure should be open. Based on this structure, any digital facility or cyber-physical system’s non-security-related functions (missions or services) and security-related functions can be realized simultaneously, “known unknown” or “unknown unknown” common ESS problems are allowed to exist in the architecture, and the inherent attributes and security effects of the architecture are of universal significance for transforming or resolving the contradiction of ESS and achieving a unity of opposites.
2. The ESS structure should be able to obtain structured security gains in an integrated form and provide a highly reliable, available, and trustworthy solution for quantifiable design and verifiable measurement and for the solution of GFS problems of digital systems, including multi-intertwined functional safety, network security, and information security problems.
3. The ESS structure should be able to integrate diversity, dynamics, redundancy, and other defense elements or functions to endow digital products or cyber-physical systems with cyber-resilient ESS in an inseparable and integrated manner.
4. The ESS structure should be able to effectively disrupt any differential-mode trial-and-error attacks or blind attacks from a mechanical perspective and can

bring stability robustness and quality robustness to the service functions of cyber-physical systems or digital products.

5. The ESS structure should be able to play the role of a “reinforced concrete skeleton” for digital technology products, naturally accept the professional safety protection technologies that have been or may be possessed in the future in the form of “concrete,” make the target system have cyber resilience like “reinforced concrete,” and give integrated quantitative design and verification indicators.
6. The ESS structure should be insensitive to the specific details of GFS problems such as software and hardware vulnerabilities and backdoors existing in the architecture. Theoretically, the structural effect can mechanically inhibit differential-mode functional safety, cybersecurity, and even information/data security incidents within the structure, regardless of the number. Even if an attacker attempts to exploit common-mode vulnerabilities in the structure, it is extremely difficult to realize blind collaborative escape under non-collaborative conditions.

2.3.2 *Desired ESS Mechanism*

The security mechanism expected by the ESS vision should have the following characteristics:

1. The ESS mechanism should be able to transform random or uncertain failures and security events caused by human or non-human factors into simple differential-mode or perceivable common-mode perturbations in the DVR domain, which can be expressed by probability tools, and can be avoided in real time by using mature reliability theories and technologies, without the need to understand the specific causes for and mechanisms of generalized uncertain disturbances.
2. The ESS mechanism does not attempt to fundamentally eliminate the ESS contradictions in the architectural environment, nor does it expect to “zero out” the security problems within the architecture. It just expects to automatically and iteratively transform the abnormal disturbance or perturbation scenario in the current architectural environment into a service scenario that is insensitive to the current disturbance behavior (at this time, there is no need to pay attention to the performance in other scenarios) or undermine the effectiveness and stability of the current attack chain by changing the running state, reallocating resources, cleansing the environment, reconfiguring the system, and introducing heterogeneous defense elements, so as to achieve the effect of “fighting the enemy with force, blocking the water with earth.”
3. The effectiveness of the ESS mechanism should not depend on (but does not reject) the richness and freshness of any prior knowledge (base) about the attacker, including the known information such as causes for security problems, triggering mechanisms, behavioral characteristics, and available resources, as well as the reliability and effectiveness of other security measures such as additional, built-in, and embedded ones. Moreover, the ESS mechanism should

be able to make it more difficult for “internal-external collaborative attacks” to crack or identify the environment of mimic disguise.

4. GFS empowered by the ESS mechanism should be able to provide cyber-physical systems or digital products with a trinity of high reliability, high trustworthiness, and high availability in terms of non-security functions and security-related cyber resilience, and it should have the stability robustness and quality robustness that can be quantifiably designed and verifiably measured. Its security effectiveness should be weakly related to the technical ability, past experience, and timeliness of maintenance of operators.

2.3.3 *Desired ESS Features*

The desired ESS features are as follows:

1. The structural empowerment characteristics of ESS should be like the role of chemical bonds in determining the hardness of carbon-based materials and should have an integrated “point-to-surface” defense function similar to the non-specific and specific immune mechanisms of vertebral organisms [15]. At the structural level, it should be inseparably integrated or fused with the intrinsic service functions of cyber-physical systems or digital products.
2. Within the ESS empowerment architecture, any network attacks based on vulnerabilities, backdoors, and virus Trojan, as well as random or uncertain failures, can be transformed into problems that can be handled in real time by the existing reliability and automatic control theories and methods through the inherent dimensionality-reduction mechanism of the architecture, without the reliance on (but not rejecting) the support of any previous knowledge about vulnerabilities, backdoors, and virus Trojans and without the prerequisite requirement for real-time accurate perception and positioning of local perturbation, disturbance, and their causes.
3. The design and engineering implementation of the ESS technology should not impose overly stringent security constraints on the hardware and software components and allow the use of functionally equivalent commodity-level soft and hard components produced by the open supply chain. It should not only economically support the demand of incremental markets for the cost performance of ESS-empowered digital products but also effectively support the transformation and upgrading of non-ESS devices or systems in the existing markets. The additive cost of ESS-empowered systems should be significantly superior to the total investment cost of life-cycle functional safety and network security protection of traditional digital products, with much better cost performance in safe use and maintenance.
4. ESS should be able to naturally integrate or accept various additional cybersecurity technologies in a structured technical environment, so that the architecture-empowered system can economically obtain endogenously safe and secure cyber

resilience like “steel-reinforced concrete,” and there is no strict safety requirement for “self-proof of innocence” except for traditional quality indicators.

5. ESS should be able to skillfully utilize the complementary relationship between the elements of dynamics, diversity, and redundancy with the help of structural technologies, so as to reduce the complexity of realizing the cyber resilience of the target system or its key links and core components, avoid the overall technical economy of the target system being affected by simply emphasizing one element, and replace the traditional global static heterogeneous redundancy mechanism with a local DHR mechanism.
6. Compared with traditional security technologies, the empowerment of the ESS architecture should endow cyber resilience engineering with security indicators that can be quantifiably designed and verifiably measured and the ability to deal with unknown threats and sabotage.

2.4 CESS Defense Paradigm

In the previous chapter, we put forward the general concept, basic characteristics, and definition of ESS problems, as well as common ESS problems, GFS problems, and the difficulty of guarding against ESS problems. With the help of the paradigm theory and concepts, this section will construct a structure of the “paradigm” on the basis of the previous “seemingly stacked and disorderly” cybersecurity technical route, so as to discover the “new structure of ESS.” It should be emphasized that the ESS defense paradigm is not and cannot be the expected approach to “absolute security.” In fact, as early as 1931, the famous Incompleteness Theorem put forward by Austrian mathematician Kurt Gödel pointed out: “Any formal system containing first-order predicate logic and elementary number theory has a proposition that can neither be proved true nor false in the system” [16]. In other words, even a logically self-consistent formal system is incomplete, and there will always be uncertain “blind spots.” Because the computer is an instantiated expression of Gödel’s computable model, it cannot be complete and cannot be expected to be “absolutely secure.” However, imperfection in the philosophical sense does not reject the role of technological invention, and under certain conditions, uncertain problems can be transformed into controllable probability problems. Just as communication technologies can solve the problem of reliable communication about error probability in noisy (incomplete) channels, CESS constructs a paradigm about “transforming human or non-human uncertain disturbances into controllable probability events,” so that the cyber resilience of digital products can be “quantitatively designed and verifiably measured.” Therefore, the problem of “self-proof of innocence” is technically transformed into the security quality that can be expressed by probability or the problem of whether it can pass the third-party’s “white-box plug” test (see Sect. 3.11 for details).

2.4.1 Prerequisites and Constraints

2.4.1.1 ESS Problems Are Only a Subset of the CyberSecurity Problem Domain

CESS problems only refer to contradictory problems that are directly related to the intrinsic functions of the target object and have explicit or implicit side effects. Problems that are not in an inherent contradictory relationship with the intrinsic functions do not fall into the category of ESS problems. In other words, even if all ESS problems can be solved, it is impossible to solve all the security problems in the entire cyberspace.

2.4.1.2 ESS Defense Focuses on Solving Common Problems

As can be seen from the above, because of the need to find a universal solution, the ESS defense domain and the solution domain for common ESS problems are highly overlapped. In other words, the core of ESS defense is to find the universal solution to common ESS problems. Individual problems in the domain of ESS problems are not in the scope of discussion of common problems, but they do not affect the derivation of solutions to common problems or the role of “stones from other mountains in polishing jade,” as shown in Fig. 2.5. In addition, all security problems outside the ESS boundary or attack surface [17], that is, the defense effects outside the ESS framework, are uncertain (may or may not be effective). For example, an ESS-empowered cipher machine empowered can provide a “Trusted Execution Environment (TEE)” for encryption or authentication algorithms. It is weakly or even irrelevant to whether there are security problems in the environment, but it cannot do much about the security flaws in the design of cryptographic algorithms. However, it is theoretically possible to deal with the potential security risks caused by individual ESS problems in the big AI model by redundantly deploying multiple software versions (such as AI software versions based on multiple models) with

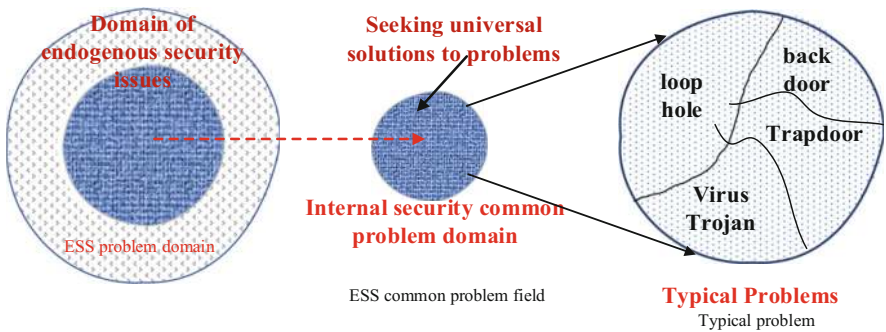


Fig. 2.5 Solution to common ESS problems

equivalent intrinsic functions and different degrees of design defects in the endogenously safe and secure TEE. Likewise, the unknown individual security defects at the level of pure functional design should be able to obtain non-specific and specific immunity similar to that of vertebrates within the ESS framework through differentiated deployment of software and hardware versions with equivalent service functions.

Empowerment of the ESS architecture not only provides security defense or generalized robust functions for components, nodes, network elements, devices, and equipment but also endows systems, platforms, networks, and even larger service systems with ESS functions.

2.4.1.3 GFS Problem Domain and Solution Domain

According to the definition of generalized functional safety, the area in which functional safety problems, network security problems, and even information/data security problems are intertwined is the domain of generalized functional safety problems in an incremental sense (problems outside this area are the respective concerns of functional safety, cybersecurity, and information security), as shown in Fig. 2.6. Since the current cybersecurity itself has not found a reliable solution to network attacks based on unknown vulnerabilities and backdoors in the target system, it is inevitable to introduce common CESS problems when applying digital, intelligent, and networked technologies in the field of functional safety [18]. Therefore, cracking common ESS problems and solving GFS problems have the same technical goal. In other words, the intertwined problems of functional safety, cybersecurity, and information security are usually difficult to solve independently. In this sense, the CESS defense goal is highly consistent with the GFS goal.

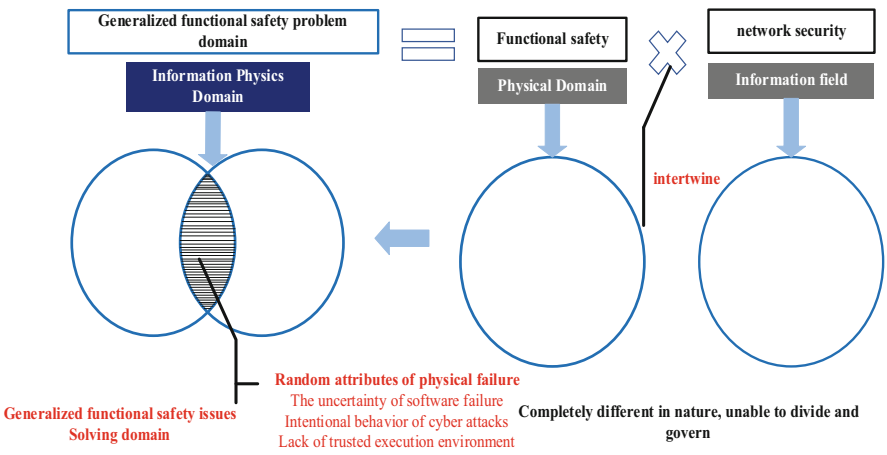


Fig. 2.6 GFS problem domain and solution domain

2.4.1.4 Technologies That Do Not Rely on Yet Do Not Reject Prior Knowledge and Traditional Security Paradigms

For defenders, the existence of “known unknown” uncertain disturbances or “unknown unknown” network attacks is relative to the prior knowledge (base) accumulated. Since the ESS defense theory can transform random or uncertain failures and complex disturbances caused by intentional or non-subjective reasons into simple differential-mode or perceivable common-mode or common-cause disturbances that can be expressed by probability in the DVR space, mechanically there is no need to rely on the abundance and freshness of various knowledge bases as a prerequisite for effective defense, so functional safety protection independent of prior knowledge can be implemented. It should be emphasized that for differential-mode or perceivable common-mode (common mode expressed in differential mode) perturbations in the DVR domain, we do not need to care about causes for or mechanisms of the perturbations in real time, even if the subsequent error correction processing mechanism only needs to meet the prerequisite of “knowing what they are.” However, ESS defense does not reject the evolution from the stage of “knowing what it is” to the stage of “knowing why it is.” In fact, with the help of technologies such as big data and AI online or background analysis, we can not only discover some “0day” vulnerabilities, backdoors, virus Trojans, and other problems with high “gold content” [19] but also achieve accurate change of problem scenarios, reduce the convergence time of dynamic feedback iteration algorithm, and improve the stability robustness and quality robustness of the system based on the knowledge obtained through learning and analysis. In addition, the differentiated deployment of traditional cybersecurity technologies can also significantly enhance the heterogeneity in the DVR domain. We will reveal the exponential relationship between heterogeneity and ESS gains in the subsequent sections.

2.4.1.5 The Application Scope of Quantifiably Designed and Measured ESS

It needs to be emphasized that the effectiveness and robustness of ESS defenses and the trustworthiness of the execution environment have a quantifiably designed and verifiably measured security indicator system only within the area covered by the structure or model. If the security defense system deploys the ESS-empowered system at a pass or strategic location, the system’s security gains can also be obtained due to the reasonable defense deployment. In other words, the structure can be either node-empowered or system-empowered. For example, the application of endogenously safe and secure native cloud introduced in Chap. 6 of this book is a kind of point-to-surface empowerment. It uses node empowerment for core components such as resource management, database management, and maintenance management, while platform-level system empowerment is used for servers and data memories with a relatively large number of configurations. This, of course,

presupposes a heterogeneous deployment or arrangement of diverse resources with functional equivalence of these services (a requirement that is not onerous in today's diverse supply markets).

2.4.2 *Change of Thinking Perspectives*

How to effectively inhibit the random “known unknown” perturbation based on common ESS problems and the uncertain “unknown unknown” disturbances including network attacks without relying on (yet not rejecting) the prior knowledge, accurate perception, and behavior analysis of the attacker, especially how to deal with unknown “internal-external collaborative” network attacks, is the fundamental difference between the ESS architectural design and the traditional cybersecurity defense paradigm: the former is the security technology of digital product design on the manufacturing side; the latter is the add-on defense technology on the user side. The two paradigms are not at the same research level at all.

Qian Xuesen, a famous scientist and CAS academician, pointed out in his famous work *The Theory of Systems Engineering* [20]: “Starting with the whole complex problem, it is considered that the whole is greater than the sum of its parts, the parts are inferior but the whole can be optimized.” When people encounter the “ceiling” problem in the conventional reliability improvement means, they find that as long as random hardware failures and uncertain software errors do not trigger common-cause failures of most redundant components at the same time, isomorphic redundant structures can greatly improve the reliability and availability of the target system in normal circumstances. Of course, the common-cause failure of isomorphic redundant systems is still a problem that cannot be selectively ignored in theory and engineering. Therefore, heterogeneous redundant structures that significantly reduce the probability of common-cause failures have been invented, although their design, engineering implementation, and maintenance are usually quite costly under the condition of functional equivalence.

The enlightening significance of heterogeneous redundant structures is that when faced with random failures or uncertain errors, functionally equivalent heterogeneous redundant structures, through relevant ruling and scheduling mechanisms, can effectively deal with “known unknown” or “unknown unknown” disturbances caused by natural or non-human factors. However, the prerequisites for the effectiveness of heterogeneous redundant structures in dealing with network attacks are often difficult to establish. The reason is that from a cybersecurity perspective, heterogeneous redundant structures have inherent defects in such aspects as stability, certainty, and similarity. As long as the attacker can understand the software/hardware vulnerabilities, backdoors, and other information of the relevant redundant components, they may cause functional downtime of the target system or achieve dormant standby common-mode escape through trial-and-error methods such as “blasting” or “penetration” one by one. This is why the classical dissimilar redundancy structure (DRS) can only be used as the “last barrier” rather than the “bottom

line of mission assurance” of an intrusion-tolerant system. From a quantitative analysis of the GSPN (generalized stochastic Petri net) model of DRS, it is not difficult to draw the conclusion that the DRS structure does not have stability robustness and quality robustness in the face of artificial attacks [3]. However, from a global perspective, the dissimilar redundancy structure can indeed transform the random perturbations and uncertain disturbances in the structure into differential-mode or perceivable common-mode failures without the impact of human factors; it can also transform the uncertain disturbances caused by human factors in the structure into simple differential-mode or perceivable common-mode incidents in the case of unknown network attacks, although such transformation still lacks the necessary robustness. For this reason, we refer to dissimilar redundancy structures as “imperfect” GFS structures.

2.4.3 *Methodological Innovation*

Developing functionally equivalent heterogeneous redundant structures and related strategic ruling and feedback iteration mechanisms (see Chap. 3 for details) can transform uncertain perturbations or unknown network attacks at the individual level into differential-mode (or perceivable common-mode) reliability incidents with “known unknown” probabilistic attributes at the group level, although in this case we cannot yet determine the underlying cause for or the mechanism of the differential-mode disturbances in real time (theoretically, it is not difficult to analyze it through background or offline processing). In other words, in the heterogeneous redundant space, we only need to perceive the scene of differential-mode or perceivable common-mode disturbances in real time to carry out scene iterative transformation or reconfigure the defense environment. Its effectiveness is weakly related or even irrelevant to the randomness or uncertainty of differential-mode disturbances. Theory and practice can prove [21] that the structure of DVR intersection based on strategic ruling and feedback control (see Chap. 3 for details) can not only perceive differential-mode or perceivable “known unknown” disturbances or “unknown unknown” network attacks at the individual level in the absence of prior knowledge but also make the differential-mode disturbance scene in the current operating environment move or transform in spatial-temporal dimension through comprehensive use of various technologies such as error correction, shielding, cleansing, reconfiguration, strategic ruling, or feedback iteration. Therefore, it can effectively deal with collaborative or non-collaborative trial-and-error attacks including APT (or dormant standby) attacks or common-mode escape and ensure the quality robustness and stability robustness of the transformed intrinsic function. In the next chapter, we will introduce in detail the core content of the methodology of the ESS defense paradigm and explain why the structure of DVR complete intersection can be an enabling architecture for the cyber resilience of cyber-physical systems or digital products.

2.4.4 *Update of Practice Norms*

Different from the traditional “best-effort” cybersecurity practice norms, in terms of technical goals, the new engineering paradigm emphasizes that whether for differential-mode or perceivable common-mode “known unknown” or “unknown unknown” perturbations, or for differential-mode or perceivable common-mode disturbances caused by human or non-human factors, the structure of DVR complete intersection should be able to give security indicators that can be “quantifiably designed and verifiably measured.” Theoretically, if the structure of DVR intersection is “absolutely heterogeneous” and the attacker cannot grasp the security defects of most executors in the current operating environment in real time, the impact of differential-mode events can be 100% inhibited and the probability of common-mode escape can be reduced to nearly zero [10]. However, there are no absolute things in the world, so “absolute heterogeneity” cannot be realized in the pursuit of “perfect security.” As a reasonable compromise, as long as the relevant components in the current operating environment can meet local security requirements, coupled with working state transition or scene migration technologies such as cleansing, recovery, intelligent reconfiguration, and ruling-based dynamic feedback control, it is possible to destroy or weaken the constructability, effectiveness, and reliability of the attack chain, and it is not difficult to obtain an iterative effect based on the adaptive intelligent fast convergence algorithm. As mentioned repeatedly before, if additional cybersecurity products with different defense mechanisms are strategically deployed in the environment of a complete DVR intersection structure, the engineering implementation complexity of heterogeneous redundant structures can be significantly reduced, the technical economy of the products can be improved, and exponential security and reliability gains can be obtained [1], without the need to put forward too stringent requirements on the security or trustworthiness of the relevant software and hardware. This natural and non-exclusive integration property is the unique endowment of the DVR complete intersection. Obviously, the new code of practice is to take the DVR intersection structure as the “reinforced concrete skeleton” of various cyber-physical systems or digital facilities and take the related intrinsic function technologies and the existing or future network security technologies as the “concrete material” to cast a “reinforced concrete-like” cyber-resilient service system that can be quantifiably designed and verifiably measured.

It should be pointed out that the DVR technology structure with ESS is naturally a “trusted execution environment (TEE)” that does not require the guarantee of an “absolutely secure root of trust,” without relying on prior knowledge of security defense. In other words, if there is a highly secure version of application software (such as some encryption, sensitive information processing, and other trusted programs) that supports cross-platform operation and needs a confidential or trusted computing platform which will not affect its safe operation, then the endogenously safe and secure processing system can be a “perfect TEE” because it can completely shield all the differential-mode security problems existing in it.

2.4.5 *Rejecting Trial-and-Error Attacks*

In the premise of reliability or traditional functional safety, there is generally no cyber resilience requirement that “for all potential damages, attention should be paid to their adversarial nature” [21]. In modern cyber-physical systems or digital products, however, this is precisely an indispensable cyber resilience function. Unfortunately, the “king of traditional functional safety”—dissimilar redundancy structure (DRS)—lacks stability robustness and quality robustness and, therefore, cannot withstand the continuous trial-and-error or blind attacks of the APT class. On the contrary, the structure of DVR complete intersection may mechanically subvert the premise that the background of trial-and-error attack is unchanged, so that any trial-and-error algorithm (including APT attack) cannot achieve the effect of “entropy reduction” in theory. This apparent “trial-and-error non-convergence” effect is unique to the structure of DVR complete intersection.

It needs to be particularly emphasized that the safe area of the DVR structure is limited to the given boundary. In this area, the common ESS problems (including some individual ESS problems under certain conditions) in the architecture can be effectively controlled. But for the security threats or sabotage outside the architecture, the defense effect is uncertain, and the attack surface (AS) is the security boundary. In other words, the “mimic disguise” of DVR structure only encrypts and camouflages all the “dark functions” in the target object except the intrinsic functions at the structural or environmental level, forming the so-called double-blind effect. It is “transparent” to the target system’s intrinsic functions or services and does not need to be decrypted. For potential “dark functions,” however, if the attacker cannot decrypt the mimic disguise, he will not be able to achieve the purpose of exploitation (see Chap. 3 for details). It can be seen that if the intrinsic function itself has security defects, the security efficacy of DVR structure will be uncertain. For example, even if we can control the security threats such as vulnerabilities, backdoors, and virus Trojans in the TEE that hosts the encryption algorithm, we still cannot overcome the design defects of the encryption algorithm itself. In a general sense, ESS defects in the design of intrinsic functions cannot be solved by the ESS design of the structure, just as no matter how smart a judge may be, he cannot correct the flaws in legal design.

2.5 Summary of This Chapter

Common CESS problems are almost ubiquitous, and their inherent contradictory nature makes them only possible to evolve, transform, or reconcile, but impossible to be fundamentally eliminated. Unfortunately, the current mainstream cybersecurity defense paradigms cannot effectively control the impact of uncertain disturbances based on common CESS problems due to flaws in thinking perspective, methodology, and practice norms. The key is that their thinking perspectives cannot get rid of

the constraints of the theoretical framework of “mending the sheepfold after a sheep is lost,” which relies on threat perception or the accumulation of relevant knowledge. The core is that the “patch methodology” does not change the basic nature of the main contradiction in the process of solving the problem of vulnerabilities and backdoors, but just turns one problem into another with the same nature, making it difficult to form a unity of opposites in the spatial-temporal dimension. Its “best-effort” engineering practice norms cannot achieve the goal of safety and security that can be quantifiably designed and verifiably measured for “unknown unknown” disturbances.

Using the innovative principle of the impossible triangle of defense against unknown threats—the IIP theorem—we can not only qualitatively analyze whether a given cybersecurity technology instance has ESS attributes but also roughly understand whether the target object can cope with generalized uncertain disturbances including unknown internal-external collaborative network attacks without relying on (but not rejecting) prior knowledge.

This chapter clearly defines and interprets the author-proposed vision of CESS defense from three levels, system structures, security mechanisms, and basic characteristics, and answers the questions of what CESS is, what characteristics it has that are different from those of the current security technology, and what theoretical and technical goals it expects to achieve. The CESS defense paradigm created provides a new perspective for the defense against generalized uncertain disturbances, a new methodology to deal with the global problem of “unknown unknown” network attacks, and an innovative way to overcome the lack of integrated structural gains in the current cyber resilience engineering design on the manufacturing side. The paradigm also provides user-side information security or data security with a “safe enclave” required for confidential calculation or trusted computing through the structure of complete DVR intersection even if there are untrustworthy factors in the “trusted execution environment.” In the next chapter, we will see how the CESS defense paradigm points out a new direction for and the design of digital products with cyber-resilient functions; how to fundamentally resolve the sharp contradictions between “openness and security, advancement and reliability, autonomy and controllability, functional safety and network security or information security” in the Internet era; and how to completely reverse the current strategic predicament of “easy to attack but difficult to defend” in cyberspace from the bottom driving theory and methodology of digital ecosystems and from both the manufacturing side and user side.

References

1. Wu Jiangxing. Cyberspace endogenous security and safety – mimic defense and generalized robust control [M]. Beijing: Science Press; 2020.

2. Hey A, Tansley S, Tolle K. Jim gray on escience: a transformed scientific method [EB/OL]. 2009. <https://www.semanticscholar.org/paper/Jim-Gray-on-eScience%3A-a-transformed-scientific-Hey-Tansley/80fa526c43dbb2fcbf725dcfd579cb214788624e>
3. Wu Jiangxing. Development paradigms of cyberspace endogenous safety and security [J]. Sci Sin Inf. 2022;52(02):189–204.
4. Kreutz D, Ramos FMV, Verissimo PE, et al. Software-defined networking: A comprehensive survey [J]. Proc IEEE. 2015;103(1):14–76.
5. Wu Jiangxing. Thoughts on the development of novel network technology [J]. Sci Sin Inf Sci. 2018;11(08):1102–11.
6. Li Haiquan, Li Jian. Computer network security and encryption technology [M]. Beijing: Science Press; 2001.
7. Li Haige, Zuo Hao. Research on the Application of Artificial Intelligence Technology in Network Security [C]. Proceedings of International Conference on Computer Network, Electronic and Automation, 2023. p. 24–28.
8. Jajodia S, Ghosh AK, Swarup V, et al. Advances in information security [M]. New York: Springer; 2011.
9. Gilbert S, Lynch N. Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services [EB/OL]. 2002. <https://dl.acm.org/doi/10.1145/564585.564601>
10. Gao Yuwei. Interpretation of international financial hotspots: impossible trinity of policy objectives [EB/OL]. 2013. https://www.boc.cn/aboutboc/ab8/201303/t20130328_2210231.html
11. Impossible Triangle [EB/OL]. <https://baike.baidu.com/item/%E4%B8%8D%E5%8F%AF%E8%83%BD%E4%B8%89%E8%A7%92/8396713>
12. Paper: don’t settle for eventual: scalable causal consistency for wide-area storage with COPS [EB/OL]. 2011. <http://highscalability.com/blog/2011/11/23/paper-dont-settle-for-eventual-scalable-causalconsistency-f.html>
13. Garcia M, Bessani A, Gashi I, et al. Analysis of OS diversity for intrusion tolerance. Softw Pract Exp [J]. 2013;44(6):735–70.
14. Ren Q, Wu JX, He L. Performance modeling based on GSPN for cyberspace mimic DNS [J]. Chin J Electron. 2020;29(4):738–49.
15. Wu JX. Cyberspace endogenous safety and security [J]. Engineering. 2022;15(8):179–85.
16. Muze C. On correct understanding of the Godel’s incompleteness theorem [J]. J Hunan UnivSci Technol (Social Science Edition). 2008; 11(2):513–534.
17. Wu Jiangxing. Cyberspace mimic defense – generalized robust control and endogenous security [M]. Beijing: Science Press; 2018.
18. Wu JX. Problems and solutions regarding generalized functional safety in cyberspace [J]. Saf Secur. 2022;1(2):1–13.
19. Wang Qing. Oday security: software vulnerability analysis techniques [M]. Beijing: Publishing House of Electronics Industry; 2011.
20. Qian Xuesen. On systems engineering [M]. Shanghai: Shanghai Jiaotong University Press; 2007.
21. Ross R, Pillitteri V, Graubart R, et al. Developing cyber-resilient systems: a systems security engineering approach [EB/OL]. 2021. <https://csrc.nist.gov/publications/detail/sp/800-160/vol-2-rev-1/final>

Chapter 3

The Principle and Structure of Cyberspace Endogenous Security and Safety



As can be seen from the previous chapter, the incomplete intersection principle (IIP) points out that if the functions of DVR in a cybersecurity defense system cannot be completely intersected, it will be impossible to have the structural capabilities to prevent unknown security threats (especially internal-external collaborative network attacks). In other words, no matter what kind of security technology is added to the target system or how many built-in, embedded, endosymbiotic, and other non-integrated defense measures are deployed, it is theoretically impossible to effectively resist “unknown unknown” security threats or “internal-external collaborative” cyberattacks. This chapter needs to prove whether the “DVR complete intersection” can, in mechanism, transform generalized uncertain disturbances into differential-mode or perceivable common-mode security incidents that can be described by probabilistic tools in the DVR domain, i.e., to theoretically analyze and prove whether this “DVR transformation” has the basic attributes mentioned in the vision of CESS. In a more general sense, it is to prove whether the inference that “structure determines security” is true at the theoretical level. Then, there is also a discussion about what kind of model or algorithm can technically realize DVR complete intersection. It is found that the introduction of a state or output feedback control mechanism on the basis of a dissimilar redundancy structure (DRS) can form a feedback loop consisting of all links, elements, or variables in the structure of DVR complete intersection that are well connected and causally related to each other, where a change in any one of them will cause a change in other links or elements, thus forming a feedback loop and controlling movement. Therefore, the feedback control is mechanically able to adjust variety (V) and redundancy (R) in real time through the dynamics (D) of iterative convergence, thus meeting the requirements of the DVR complete intersection model. In view of this, the author invented the dynamic, heterogeneous, and redundant (DHR) architecture (the same papers also called “the mimic architecture (MA)” or “mimic defense”) [1] based on the principle that “structure determines security.” It solves multiple intertwined functional safety, network security, and even information/data security problems such as unknown threats or attacks in the architectural environment with an integrated effect. Then,

this chapter introduces in detail the structure and function of the DHR architecture, the mechanism of ESS, the environmentally encrypted attack surface (AS), and the engineering expression of ESS and proves from the perspective of cryptography that, compared with the “perfect secrecy” model of Shannon’s information theory, the ideal DHR model, which does not rely on prior knowledge, essentially puts forward an innovative method to obtain “perfect safety” functions through “structural coding/environmental encryption.” This enriches the connotation and extension of the ESS theory. Finally, a general testing and “golden standard” of measurement of ESS functions and performance—the white box plug method—is given.

3.1 A Conjecture About DVR Complete Intersection

Assuming that a system has a structure of $D \cap V \cap R$ intersection (hereinafter referred to as DVR complete intersection) as shown in Fig. 3.1 and meets the following prerequisites:

1. It does not rely on any prior knowledge about the attacker.
2. It is based on completely heterogeneous variety (only established at the theoretical level).
3. $f \leq (n - 1)/2$ (Note: n refers to the number of available executors at any time, and f refers to the total number of redundant bodies that are allowed to have differential-mode faults at any time).

Then, it is not difficult to infer that this structure has ESS defense capabilities that do not rely on any prior knowledge for the following differential-mode (including perceivable common-mode) problems in the DVR domain:

1. Differential-mode security problems caused by random or uncertain disturbances.
2. Differential-mode security problems caused by human or nonhuman factors.
3. Any differential-mode problems caused by the interweaving of functional safety, cybersecurity, and even information security problems.

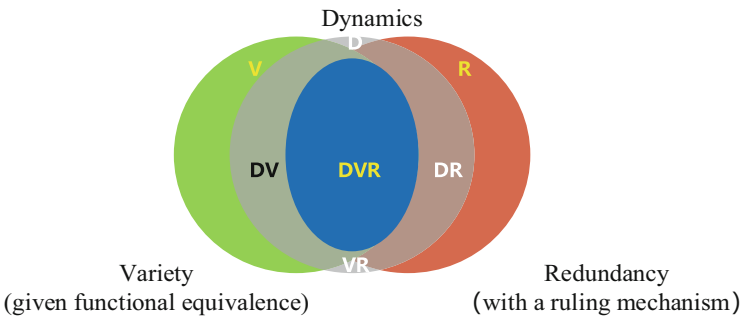


Fig. 3.1 DVR complete intersection (Venn diagram)

It needs to be emphasized that in theory, if the heterogeneity between redundant components can be guaranteed at any time in the DVR domain, it is usually impossible to generate common-mode disturbances in the domain. In engineering practice, random hardware failures and uncertain software errors caused by nonhuman factors can support this assumption with a high probability. However, under special or extreme conditions, network attacks caused by human factors may cause common-mode disturbances or escape incidents (perceivable or unperceivable):

1. All the heterogeneous redundant components with equivalent functions in the structure have backdoors preimplanted by the attacker with the same function or trapdoors known only to the attacker.
2. The attacker masters or understands the attack resources such as differential-mode vulnerabilities and backdoors in most (greater than or equal to the threshold of majority-based ruling) of the heterogeneous redundant components in the structure and has the ability to exploit these resources dynamically and collaboratively through the attack surface to achieve common-mode escape across heterogeneous redundant bodies or execution scenarios.
3. There are software and hardware code design defects with the same logical function in multiple heterogeneous redundant components within the structure, and the defender is not aware of them (such as unknown security problems in open-source software and hardware codes or cross-platform diversified versions).

Therefore, the complete intersection conjecture (CIC) of DVR can be expressed as follows: In a digital system, if there is a functional safety or cybersecurity (or even information/data security) threat unknown to the defender, whether it comes from random or uncertain perturbations or from human or nonhuman disturbances, it can always be expressed as a differential-mode or perceivable common-mode local incident within the DVR domain.

3.2 ESS Existence and DVR Transformation

1. ESS-Existence Conjecture (ESS-EC).

If a structure or algorithm has a completely intersecting expression of the three elements of dynamics (D), variety (V), and redundancy (R), then differential-mode attacks based on unknown vulnerabilities, backdoors, and virus Trojans in the structure can be controlled based on the endogenous effects of the structure even in the absence of prior knowledge, and differential-mode disturbances caused by random or uncertain factors can be inhibited. Section 3.3 of this chapter will prove that the DVR complete intersection conjecture (CIC) and ESS-existence conjecture (ESS-EC) are theoretically established and have the same characteristics.

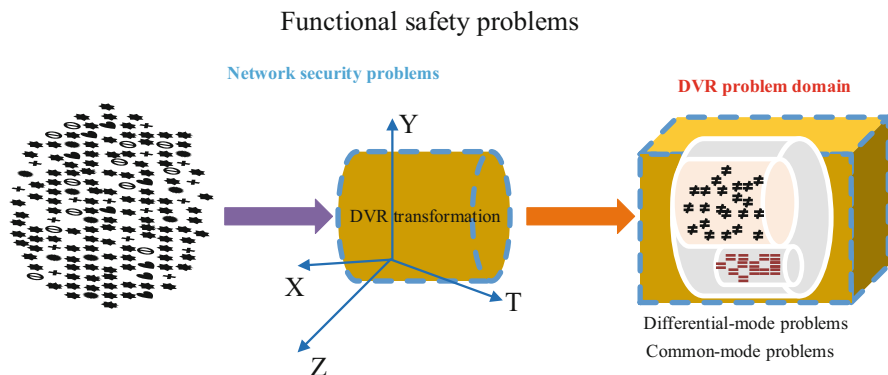


Fig. 3.2 DVR transformation in the four-dimensional space

2. DVR Transformation.

According to ESS-EC, if the structure where D, V, and R completely intersect or the $D \cap V \cap R$ intersection can transform the internal disturbances or perturbations of various network security and functional safety problems into a differential-mode or common-mode expression of DVR intersection in a four-dimensional space, the multiple intertwined problems of functional safety, network security, and even information/data security within the structure can be solved integrally without any prior knowledge. Figure 3.2 gives a schematic diagram of this transformation mapping relationship, where $XYZT$ represents the four-dimensional space of three-dimensional entities + time under the intersection of DVR.

It needs to be emphasized that DVR transformation aims to hide the diversity, complexity, causes, and mechanisms of perturbations or disturbances in the domain of functional safety and network security problems, so that the perceivable security incidents mapped to the DVR domain have only two simple dynamic expressions: differential-mode or common-mode incidents (strictly speaking, there is a difference between unperceivable and perceivable common-mode incidents). This dimensionality reduction processing method can deal with human/nonhuman, random/uncertain, and differential-mode/common-mode disturbances or network attacks in real time with the help of mature cyber resilience and automatic control theory and method. In this case, we only need to pay attention to how to obtain the high-probability differential-mode results based on the necessary diversity law of cybernetics [2] and the relatively true axiom [3] in the shortest possible time and put the more difficult problems such as accurate positioning and accurate analysis, which rely heavily on prior knowledge and complex analytical ability, in a relatively minor position, and even deal with them in the background in an offline way.

However, as a conjecture, ESS-EC assumes that there is some kind of algorithm model or structure that can transform the disturbance or perturbation based on unknown common ESS problems into events expressed in a differential mode or

common mode in the DVR domain, so ESS-EC should be valid in theory. Nevertheless, even if it can be proved theoretically, there is still no answer to the questions of what is the model of DVR transformation algorithm or structure, what is the logical structure, whether it can be realized in engineering, whether there is a solution to the common-mode problem, etc. Therefore, how to organically connect dynamic, heterogeneous, and redundant functional sets through an integrated structure has become a problem that needs in-depth thinking and research. Professor Albert Einstein once said: “Problems cannot be solved in their original dimension, but can only be presented. Only by adding dimensions can current problems be solved.” If we confirm that there is no solution in the current spatial dimension, it is possible that a new way to solve the problem can be obtained with the help of the physical law of increasing dimensionality. In the next section, we will look at how to create a theoretical model that can realize DVR complete intersection.

3.3 Deriving DVR Transformation Structure from Heterogeneous Redundancy

ESS problems, especially common ESS problems, in cyber-physical systems or digital products cannot be changed by people’s will or consciousness nor can they be completely solved by supply chain security strategies. Problems such as vulnerabilities, backdoors, and trapdoors in “untrusted supply chains” can never be completely eliminated through independent controllable technology pathways. Therefore, it is necessary to create an ESS defense paradigm based on the rationale that “structure determines security.” Just as a triangle in the Euclidean space is geometrically stable, DVR transformation is to construct an algorithm model that can organically integrate dynamics, diversity, and redundancy and has integrated security gains at the structural level. It can transform complex, changeable, and unpredictable multi-intertwined problems of functional safety, network security, and information security into technical problems that are monotonous in type and simple in nature and can be expressed by probabilistic tools and easily handled by mature theoretical methods.

William Ross Ashby, a British neurophysiologist and cyberneticist, put forward the law of necessary diversity (also known as Ashby theorem) in his book *Introduction to Cybernetics* published in 1956, which means that “in nature, only diversity can destroy diversity.” In other words, the law of necessary diversity means the adaptability and evolutionary ability of nature, especially the ability to cope with uncertainties in the development process. The DRS based on the law of necessary diversity and the consensus mechanism of relatively true axiom can transform the ESS problems such as random failure, uncertain errors, known and unknown security perturbation, manmade and non-manmade disturbance, etc. into probabilistic events with differential-mode or common-mode properties expressed in the VR domain. However, due to the inherent structural stability, certainty, and similarity of

DRS, it is unable to cope with unknown network attacks based on internal-external collaboration or persistent trial-and-error cyberattacks. In other words, the heterogeneous redundant structure does not have the expected stability robustness of the DVR complete intersection set. Nevertheless, studying the DRS mechanism of classical heterogeneous redundant structure helps to find a new structural model that meets the requirements of DVR complete intersection ($D \cap V \cap R$).

3.3.1 True Relatively Axiom and Dissimilar Redundancy Structure

“The relatively true axiom” [4] (which is also called the consensus mechanism by some researchers) means that “everyone has shortcomings of one kind or another, but it is rare that most people make exactly the same mistake in the same place at the same time when performing the same task independently.” In the 1970s, in the field of reliability or resilience engineering, the law of necessary diversity was combined with the relatively true axiom, and the dissimilar redundancy structure (DRS), also known as heterogeneous redundancy (HR) system, was created. It was first successfully applied to flight controllers in the aviation field, and its abstract model is shown in Fig. 3.3. Under certain preconditions or constraints, the random failures of its soft and hard components with different forms of distribution or the statistically significant uncertain failures caused by unknown design defects or errors can be transformed into differential-mode or common-mode incidents that can be expressed with probability by the consensus mechanism, so that engineers can not only improve the dependability of the target system by improving material properties, enhancing component quality, and developing new technologies but also break

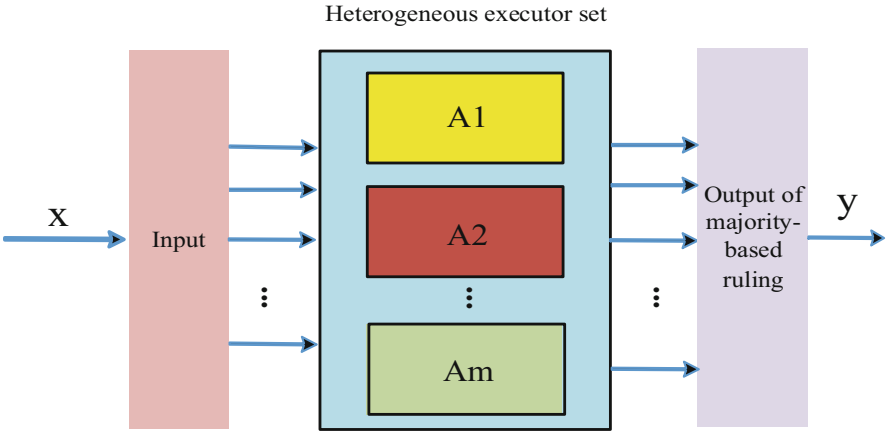


Fig. 3.3 An abstract model of DRS

through the “ceiling” of dependability and availability through the innovation of system architecture technology.

Theoretically, if the condition of “absolute heterogeneity” between redundant executors can be satisfied and the disturbances and perturbations in the structure or model are random, there may only be the expression of differential-mode security incidents in DRS. However, the requirement for “absolute heterogeneity” is impossible to meet in engineering practice, and there is still no fundamental breakthrough in theory and methodology as for how to quantitatively design and evaluate heterogeneity. In the development of its B-777 twin-engine wide-body aircraft autopilot control system, Boeing Company of the United States used multiple design teams with different technical backgrounds and a “back-to-back” R&D approach, who managed to replace the harsh conditions of “absolute heterogeneity” with “dissimilar redundancy,” which was treated differently in engineering management and design specifications. Practice shows that this move has indeed achieved the expected technical results but led to a high financial cost of multiversion maintenance throughout the life cycle [5].

Long-term engineering practice has also found that although both heterogeneous redundancy and isomorphic redundancy have common-cause (common-mode) failures, it is indeed difficult, in a statistical sense, for physical random disturbances to trigger the same common-cause failures or faults caused by common-mode defects of multiple independent physical entities at the same time. Even if the design bugs of the same software version are in a isomorphic redundant environment, the probability of common-cause failures is still within the tolerable range of most application scenarios because of different working conditions or operating environments of the executor (e.g., in master-slave or load-sharing or cold/hot backup mode). Therefore, isomorphic redundancy, which is less difficult to realize technically, has become the mainstream functional safety empowerment structure, while the heterogeneous redundant structure and DRS can only be used in some special applications (such as aircraft, aerospace systems, high-tech weapons, equipment, etc.) that are stringent in security requirement and insensitive to whole-lifecycle operation cost.

3.3.2 Qualitative Analysis of the Ability of DRS to Defend Against Unknown Attacks

At the end of the twentieth century, some researches proposed using DRS as an intrusion tolerance technology [6] and “the last line of defense to stop fish from slipping through the net.” Here, we might as well qualitatively examine the effect of DRS in cybersecurity defense.

For convenience, firstly we assume that the elements of m executors in DRS meet the prerequisite of “absolute heterogeneity”; second, we assume that there are common ESS problems in any executor A_i , and then we assume that unknown network attacks based on vulnerabilities and backdoors in the executors are

inevitable, and the attack surfaces are located at x and y ends. Therefore, at least the following three situations may occur:

1. An input sequence at the x end may cause the element A_i in the m executors to produce a different output response from other elements. Majority-based ruling considers this to be a differential-mode incident whose output can be shielded. At this moment, the DRS actually does not know what kind of attack sequence exists at the x end, and the reasons and mechanisms that cause executor A_i to generate a differential-mode state are even less clear. At this time, there are two possibilities: one is that the unknown design defect of A_i executor is triggered by random disturbances of nonhuman factors and is manifested as a differential-mode error or fault; the other is that the unknown design defect is deliberately triggered by a network attack, which is manifested as the appearance of a response to differential-mode attacks. However, analyzing the operating mechanism of DRS, we can find that it does not care whether the design defect of A_i is known or unknown, nor does it care whether the input sequence at the x end is a compliant operation sequence or a hidden attack sequence, as long as it can perceive the existence of executor A_i with differential-mode output response in the structure. In other words, no matter what differential-mode common ESS problems exist in m executor elements, no matter whether the disturbance is random/uncertain, known/unknown, manmade/non-manmade, and regardless of the compliance of x input sequence, as long as executor A_i is expressed in the form of differential-mode output, it can be found and shielded by the majority-based ruling mechanism.
2. If the attacker can master the differential-mode common ESS defects in multiple ($\geq$ the threshold of majority-based ruling) executors, and the precondition of $f \leq (m - 1)/2$ (f is the number of differential-mode output executors and m is the total number of heterogeneous executors) is no longer established by “blasting” attacks one by one according to the above method before the problem executors are cleansed and resumed, then the target system based on DRS will not be able to avoid the risk of downtime.
3. An input sequence at the x end may activate the attack resources such as the dormant backdoors and trap doors in an executor element A_i (e.g., performing some sensitive data packaging operations for ending), but it does not show different output responses from other actors, so the majority-based arbitrator is completely insensitive at this time. In the same way, it is assumed that the attacker can control the common-mode backdoor functions in most executors, complete the same dormant-standby operation according to the above steps, and then simultaneously activate the executors in dormant-standby state using a compliant input sequence instruction at the x end. At this point, the attacker can take advantage of the blind zone of majority-based ruling to achieve the desired attack purpose in the mode of “arbitrator side-channel attack” [3]. It should be pointed out that “absolute heterogeneity” among m executors is impossible to realize in practical engineering, so the problem of common-mode escape in the blind zone of majority-based ruling can never be ignored.

It is not difficult to see that although the “last line of defense” based on the intrusion tolerance of DRS can indeed suppress the impact of differential-mode common ESS problems in the structure, on the whole, the operating environment of heterogeneous executors and the exploitable network attack resources such as vulnerabilities and backdoors in DRS are still static, certain, and similar, and the mode of parallel deployment of executors usually does not change the accessibility of the attack surface. Therefore, under the condition of having necessary prior knowledge, accurate attacks can be launched one by one against the DRS system to make the “mission assurance” prerequisite of $f \leq (m - 1)/2$ difficult to maintain, or realize the common-mode escape of sensitive information based on “dormant standby,” and make successful attack experience inheritable, the attack method reproducible, and the attack effect sustainable. In other words, in the face of network attacks based on common ESS problems, the static and deterministic DRS cannot break through the constraint of IIP or do not have the attribute of complete intersection required by DVR transformation. Therefore, it inevitably has genetic defects in structure or algorithm, and it is impossible to have stability robustness in dealing with unknown network attacks [3].

3.3.3 *Enlightenment and Discovery*

Although DRS does not have stability (time) robustness and quality robustness against unknown network attacks, the above analysis does not prevent us from gaining the following three enlightening insights:

1. DRS can map all kinds of common ESS problems and complex and changeable generalized uncertain disturbances in A_i to differential-mode or common-mode problems with probabilistic attributes in the static heterogeneous redundancy (VR) space (as shown in Fig. 3.4), and it is not necessary to accurately understand the mechanisms of common ESS problems and the characteristics of uncertain disturbances and attacks as a prerequisite, thus opening up a new way of “structure determining security” for network security defense without relying on vulnerability (backdoor) knowledge base, virus Trojan library, and attack feature library (e.g., the hardness of graphite and diamond is determined by the structure of chemical bonds between carbon atoms). However, DRS obviously still fails to achieve the expected ESS goals of DVR transformation.
2. Under the premise that “absolute heterogeneity” cannot be guaranteed, if attackers can grasp certain attack resources in advance, when the number of redundancies is n , they can still exploit the blind zone where the majority-based ruling mechanism is unable to determine whether $(n - 1): 1$ or $1: (n - 1)$ is a “true differential-mode” problem or a “perceivable common-mode escape” and adopt the strategy of continuous dormant-standby infiltration or internal-external collaborative attacks to realize unperceivable stealing of sensitive information based on arbitrator side-channel attack [7] or perform other operations desired by the

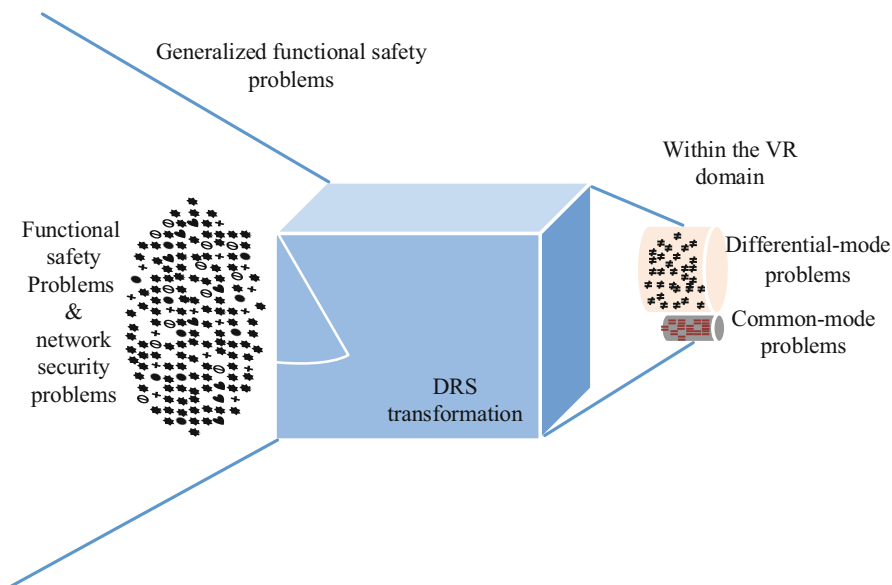


Fig. 3.4 Transformation from DRS to the VR domain

attackers. This not only directly threatens the “mission assurance” function of the target system but also threatens or destroys the confidentiality, integrity, and availability of data resources stored or processed in DRS.

3. Because of the static nature of DRS, the certainty of the operating mechanism and attack surface, and the difficulty in meeting the requirements for global statics and heterogeneity in engineering practice, attackers can grasp the common ESS problems of most executors in advance, and when the average time of recovery/cleansing operation of the executors is longer than the interval of trial-and-error attacks, they can “paralyze” most of the executors one by one using the method of probability-based trial-and-error attacks, thus causing system downtime. Referring to the concept of information entropy [8], the static nature of DRS makes the initial information entropy have no self-sustaining mechanism, so it can only reduce the entropy under persistent trial-and-error or blind attacks until the initial information entropy is so low that the attack chain can reliably play its expected role. At this point, the DRS will completely lose its intrinsic function or safety function. For example, if the attacker has mastered the ESS defects of most executors in the structure in advance and reached the state of $f \geq (n - 1)/2$ before the differential-mode executors completed the cleansing and recovery operations, system downtime and collapse will be inevitable [9], and the “dormant-standby” information security attack cannot be inhibited.
4. Although the arbitrator can detect differential-mode or perceivable common-mode disturbances in the structure, they will not automatically disappear if the diversified or redundant scenes in the current operating environment cannot be changed or reset in real time. In other words, it is necessary to have a dynamic

deployment/cleansing/reconfiguration mechanism of executor scenes based on the awareness of the operating environment.

3.3.4 *DVR Complete Intersection to Be Realized by Feedback Control*

According to the feedback principle of cybernetics, in a feedback control system, the control effect exerted by the control device on the controlled object is taken from the feedback information of the controlled variable and is used to constantly correct the deviation between the controlled variable and the input variable, so as to complete the task of controlling the controlled object. Its operating mechanism is that the control device and the controlled object form a feedback loop in which all links, elements, or variables are well connected and causally related to each other, where a change in any one of them will cause a change in other links or elements, thus forming a feedback loop and controlling movement. So, feedback control has the dynamics of iterative convergence, as shown in Fig. 3.5. It should be pointed out that the feedback function based on causality is to achieve a common functional goal, so feedback not only establishes a close relationship between causality and purpose but also reflects the internal relationship between stability and adaptability. William Ross Ashby put forward the concept of “ultrastability” in his book *Design for a Brain* [10], in which he wrote: “When the old stability of a system is inevitably destroyed, if there is an (adaptive) mechanism to re-establish the stability, such a

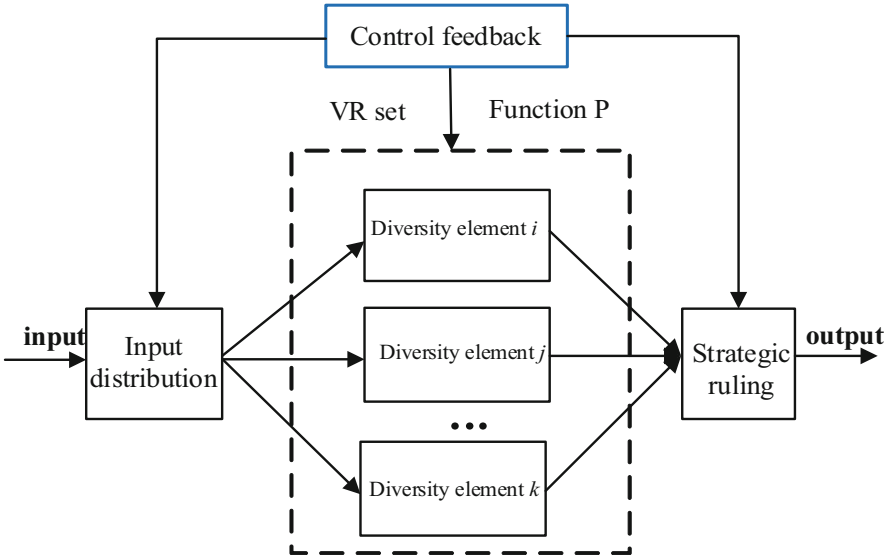


Fig. 3.5 Feedback control can form DVR complete intersection

system can be called an ultra-stable system.” Facing the systematic requirement of restraining the influence of uncertain disturbance in structural environments, the key of feedback control of an ultra-stable system lies in whether there is sensitive, accurate, and powerful feedback.

The significance of classical output feedback control is to use the relevant output as the feedback variable to form a feedback law and achieve closed-loop control of the system, thereby achieving the desired system performance indicators. The significance of state feedback control is that each state variable of the system is multiplied by the corresponding feedback coefficient, which is fed back to the input end added with the reference input, and the sum is used as the control signal of the controlled system. The significance of fuzzy output feedback control is to describe and model the uncertain nonlinear system with the fuzzy T-S model and then realize the design of system control. For example, fuzzy modeling is performed for a class of uncertain nonlinear time-delay systems to obtain the design of fuzzy state feedback control and observer-based dynamic output feedback control. The fuzzy controller is used to prove that the uncertain fuzzy time-delay system is asymptotically stable.

It is not difficult to deduce that if a heterogeneous redundant scenario dynamic scheduling mechanism based on output feedback control is introduced into DRS, and the state of strategic ruling is used as the feedback variable to form a feedback control law, it will undoubtedly change the inherent defects of statics and certainty of DRS and form a causal feedback loop among variety, redundancy, and dynamics, so that the original independent or dispersed DVR elements can form a complementary relationship.

However, this inference also needs to solve at least the following three thorny engineering implementation problems:

First, it is impossible to distinguish between “true differential-mode problems” and “perceivable common-mode escape” only by relying on the majority-based ruling strategy of DRS, that is, there is a “blind zone of ruling” as mentioned earlier. For example, in a triple-modular-redundancy (TMR) DRS system under natural perturbations, when the arbitrator finds that there are a few different scenarios in the output of the executor, they can be regarded as differential-mode incidents in most cases. However, the possibility of differential-mode attacks or perceivable common-mode escapes cannot be ruled out under human disturbances. In particular, as long as the attacker can master the vulnerabilities/backdoors in two executors, it will not be difficult for him to put the DRS system into a “dormant-standby” state and realize desired collaborative attacks or common-mode escapes by orchestrating or coordinating the compliance sequence of the shared input channel. Therefore, it is necessary to introduce a multi-relativity iterative ruling mechanism. For example, the historical performance weight of each executor, the security design level of software and hardware, the system security protection level, and the current software version number are normalized into parameters that can be used for iterative ruling.

Second, due to differences in the operating environments of heterogeneous redundancy (even isomorphic redundancy), it is impossible to achieve strict temporal synchronization in the output responses of each executor. In other words,

although the executors are functionally equivalent, the output response time is uneven relative to the input excitation. Therefore, under the harsh conditions of output response time, it is difficult to ensure that the arbitrator can give feedback control parameters in real time or determine the output of an executor as the system response output. In addition, the executor may also generate suggestive or protective alarms during operation, which often occur before the output of ruling. Therefore, the arbitrator also needs to make a strategic ruling based on the normalized information. For example, injection attacks may cause illegal instruction alarms; overflow attacks may cause program or data space overflow alarms; and improper stealth attacks may cause illegal process alarms, execution file security certificate authentication alarms, various security software-related alarms deployed differently on the executor, and software/hardware fault alarms or prompts given by the operating system. In addition, some software that uses random parameters, such as the initial serial number of the TCP, encryption authentication, system function calls, etc., are strongly related to the executor's software/hardware configuration, resource scheduling, and operating state. Even if the executors run diversified versions generated by the same source program, they cannot completely avoid inconsistent differential-mode output results, which are not caused by uncertain disturbances (except that the output results are chronologically different), so there are still many challenges in engineering implementation simply using the feedback control mechanism based on output ruling.

Third, no matter what feedback control mechanism is used, once it is activated, it will issue commands to change the current operating environment or resource configuration. For example, it may designate the problem executor to cleanse the operating environment or change the current operating state or even implement scenario migration and replacement operations. There are multiple options and the question of how to achieve optimization (involving feedback convergence time or system security degradation duration). As long as the executor carries services based on the state machine principle, once it is changed, the problem of data file consistency or program state "catch-up" will occur. Moreover, before "synchronously" catching up with other executors, the system security is always in an $N-1$ degraded state (N is the number of heterogeneous redundant bodies or operation scenarios), although this should have been a problem that the "CAP impossible triangle" of distributed computing systems needs to deal with.

In short, the introduction of a feedback control mechanism is probably the only mechanism to achieve complete intersection of D, V, and R (which needs to be further proved in theory) and can significantly increase the apparent uncertainty of the DVR complete intersection (which can also be regarded as a structural encryption effect; see Sect. 3.6 for details), and it will also bring many new thorny problems at the engineering and technical implementation level.

3.3.5 *Discovery of the DVR Transformation Structure*

According to the above discussion, if we can introduce an output state feedback control mechanism into DHR or DRS to ensure the dynamic balance of initial information entropy, we will be able to largely avoid or alleviate the problem of its lack of stability robustness and quality robustness under network attacks. In other words, we should not only introduce dynamic feedback control mechanisms such as strategic ruling, feedback control, and iterative convergence to obtain the effect of entropy nondecreasing but also naturally accept and differentially deploy additional security defense elements such as antivirus, backdoor blocking, vulnerability patching, encryption authentication, intrusion detection, and intrusion prevention measures so as to significantly reduce the difficulty of engineering implementation caused by heterogeneous design and technically and economically enhance the heterogeneity within the executive set and obtain exponential security gains. In this way we should, in theory at least, be able to fundamentally overcome the genetic defects of stability and certainty of the DRS operating environment, so that attackers can neither fully understand the common ESS defects in the operating environment of the empowered objects nor grasp and exploit the available attack resources in the current execution environment in real time, making it even more difficult to achieve “coordinated common-mode escapes in the DHR environment under uncooperative conditions.” The error-correction or shielding mechanism of this structure should make it difficult for attackers to perceive or recognize the target environment and evaluate the attack effect. Its strategy iteration mechanism of cleansing, recovery, and dynamic reconfiguration and operation scenarios should enable the defenders to “effectively deal with the current generalized uncertain disturbances at the right time and in the right running scenarios” and achieve the adaptive defense goal of “fighting the enemy with force, blocking the water with earth.” Its feedback control mechanism should be able to form a causal feedback loop among diversity, redundancy, and dynamics, so that the originally independent or dispersed defense elements can form a mutually reinforcing relationship, and the complexity of engineering implementation can be effectively reduced. It is expected that this innovative ESS-empowered structure will have GFS that can be quantifiably designed and verifiably measured. The author named this ESS-empowered structure DHR architecture [11], as shown in Fig. 3.6. Obviously, the DHR architecture belongs to one of the DVR complete intersection algorithms (its uniqueness cannot be proved yet), as shown in Fig. 3.2. Next, we will deduce the ESS coding or environmental encryption theory based on the DHR architecture model and Shannon’s channel coding theory and prove the existence ESS.

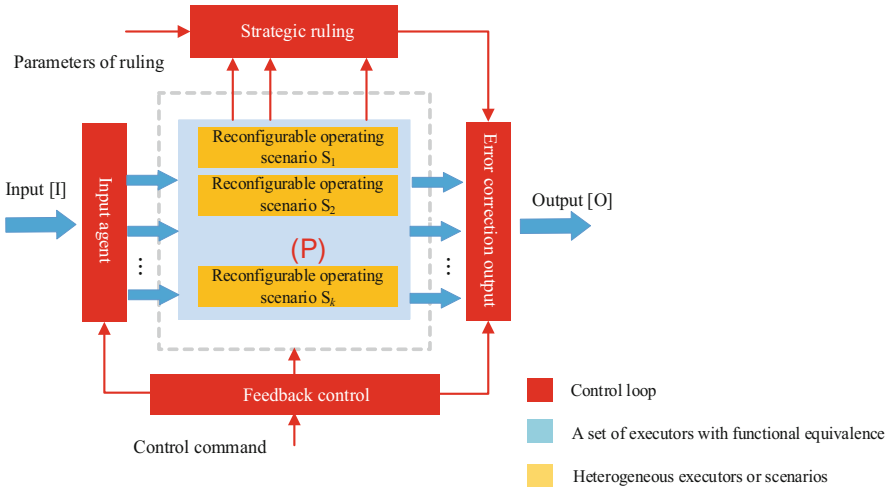


Fig. 3.6 DHR architecture model

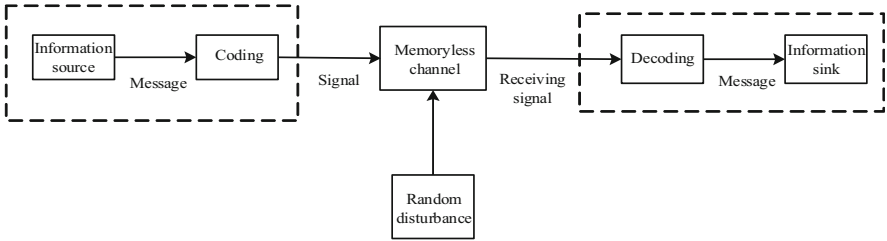


Fig. 3.7 Shannon's dependable communication model

3.4 Principle of Existence of ESS Coding

Can we get the inference that “structure determines safety” from the axiom that “structure determines function, performance, and efficiency”? In other words, in terms of cyberspace security and safety, it is necessary to prove whether there is a structure or architecture that can effectively resist persistent “known unknown” or “unknown unknown” threats or “internal-external collaborative” sabotage without relying on prior knowledge and additional security measures. Before Shannon put forward the famous channel coding theorem [12] in 1949, the problem of dependable communication under noisy conditions was a typical “known (wrong communication data caused by noise) unknown (error-correction coding model)” problem. Shannon’s dependable communication model is shown in Fig. 3.7. Shannon’s second theorem (noisy channel coding theorem) proves that the dependable transmission of messages can be realized by adding properly designed redundant codes to the original message to be transmitted and then reconstructing (decoding) the original message at the receiver under the assumption that memoryless channels

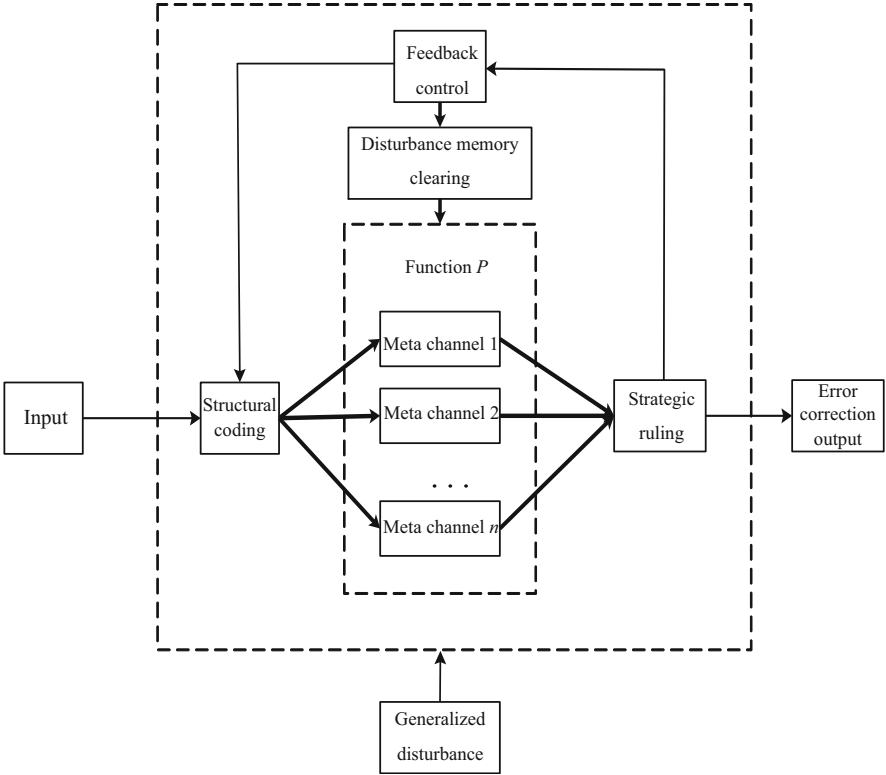


Fig. 3.8 An endogenously safe, secure, and dependable information processing model

and random noises lead to random errors. The theorem also proves that there is a communication method in which the probability of communication errors can infinitely approach zero, which points out a new direction or new way of modern dependable communication based on channel coding/decoding technologies.

Random or uncertain disturbances and manmade or non-manmade disturbances caused by common CESS problems can be described as problems of reliably, continuously and correctly processing information by information systems, where information processing includes the calculation, storage, and communication or transmission of information. To facilitate discussion, here we convert the DHR architecture model in Fig. 3.6 into an endogenously safe, secure, and dependable information processing model and assume that the model's function P is a function of the structural coding of n heterogeneous meta-channels (x), i.e., $P = f(x) = f(x_i) = f(x_1 \cup x_2 \dots \cup x_n)$, where $x_1 \neq x_2 \dots \neq x_n$, as shown in Fig. 3.8. Different from the assumption that memoryless channels and random disturbances lead to random errors in Shannon's dependable communication model, on the one hand, a network information system generally has computing, storage, and communication or transmission capabilities and can be abstracted as a channel with ESS problems and

memory or a general Turing machine; on the other hand, random communication noises, random physical failures, and network attacks can be collectively called generalized disturbances, which can lead to known or unknown, random, or non-random errors.

The impact of CESS problems on information systems is manifested in various known and unknown faults, which generate random or nonrandom errors or failures when activated by generalized disturbances. Attacks against the target object are usually not independent random incidents but are expressed as Boolean quantities rather than probability values in a mathematical sense. This makes traditional attack-resistance analysis difficult to describe with the help of stochastic process tools. However, for a system that meets the requirements of DVR complete intersection, certain or uncertain errors caused by generalized disturbance activating executor vulnerabilities, backdoors, virus Trojans, and dependability errors can be transformed into some distributed differential-mode/common-mode probability incidents at the system level without relying on any prior knowledge and additional security measures. By dynamically adjusting the DVR parameter configuration, the impact of system failure can be controlled within an acceptable range. In other words, through DVR transformation, the target system can normalize its dependability errors and the attacks on its meta-channels into probabilistic differential-mode or common-mode problems at the system level, thereby making it possible to solve the intertwined functional safety, network security, and even information security problems in an integrated manner and use the mature probability theory and statistical methods for quantitative analysis.

If viewed from Shannon's redundant coding theory, the DVR structure can be spatially and temporally expanded into a "structural code" based on the DHR architecture and can also be regarded as an "environmental encryption." The purpose is to combat the impact of random or nonrandom "structure disturbance noise (SDN)" similar to channel noise at the structural level. However, the analysis objects of Shannon's channel coding theory are "random disturbance errors and memoryless channels without processing capability," while the heterogeneous redundant feedback control scenario of DVR is equivalent to "random or nonrandom disturbance errors and channels with memory and processing capability." Therefore, Shannon's theory and method cannot be directly used to quantitatively analyze the safety or generalized robustness of the DVR architecture. It is necessary to develop a structural error-correction coding theory that meets the requirement of DVR complete intersection from the error-correction coding theory and theoretically prove that under generalized disturbances, dependable/continuously correct information processing services can be ensured by creating appropriate structural coding for specific defective memory channels. The wording "dependable/continuously correct" means that the steady-state probability of errors in information processing is small enough when there is any random or artificial additive interference in the architecture of digital products with ESS attributes by adopting appropriate structural coding and decoding steps. In short, the theory of existence of ESS coding is composed of the lemma, theorem, and mathematical proof of the existence of ESS coding, thus creating a new method of "dependable/continuously correct"

information processing by memory channels/universal Turing machines that have ESS problems, under generalized disturbances. In this sense, Shannon's second theorem is just a special case of achieving dependable communication under the condition of "random disturbance errors and memoryless channels."

3.4.1 *The Concept of ESS Coding*

A typical information system has five basic attributes: functionality, usability, performance, cost, and dependability. Among them, dependability is the ability of a component, device, or system to complete specified functions under the specified conditions within a predetermined time.

From the perspective of dependability and robustness control, the safety-related definitions in this chapter are given as follows.

Definition 3.1. Generalized disturbances are the results of interaction between internal and external factors that may cause changes in system functional safety and cybersecurity characteristics or parameters, including random and nonrandom perturbations and artificial or nonartificial disturbances, such as communication noise, software and hardware failures, cyberattacks, etc.

Definition 3.2. Fault is a state in which the system components cannot complete the functions required by the design specifications, and it is the cause of errors. When a fault causes an error, it is active; otherwise, it is dormant.

Definition 3.3. Error is a manifestation or state when a component fault in the system is activated. When an error reaches the system service interface and changes the service, it causes the system to fail.

Definition 3.4. Failure is a state when a service provided by the system deviates from the correct one.

Definition 3.5. Security is a big concept. In this chapter, security refers to the ability of the target system to continue to provide services that can be reasonably trusted under generalized disturbances. It can be quantitatively evaluated by the probability of correct service or the probability of system failure.

Definition 3.6. Meta-channel is a subchannel that has the same function and performance and has transmission and processing capabilities within the system.

Definition 3.7. Structural coding refers to a coding scheme that meets certain structural requirements to achieve specific functions and performance.

Definition 3.8. Error-correction decoding refers to a set of strategic rulings based on relativity algorithms, parameters, and related information.

Definition 3.9. Feedback control means that the system forms a corresponding robust control strategy according to the result of strategic ruling, combined with the

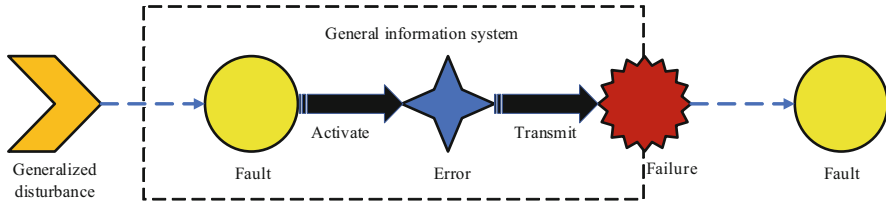


Fig. 3.9 Action chain of generalized disturbance

given algorithm and parameters in the channel or through dynamic iterative mechanism, and acts on related links such as structural coding, memory elimination, and error-correction output.

Definition 3.10. Memory elimination refers to the timely elimination of errors or failures with memory effects caused by communication noise, reliability failure, and human disturbance by the system through feedback control strategy.

Definition 3.11. Coding of endogenous security structure (“ESS coding” for short) is a security defense coding scheme that meets the requirements of endogenous security structure and uses multiple meta-channel combination coding methods to correct errors or failures caused by uncertain disturbances on memory channels, which mainly includes functional units such as structural coding, meta-channels, strategic ruling, error-correction decoding (output), feedback control, memory elimination, etc.

Definition 3.12. Reliability disturbance refers to random physical faults, errors, or uncertain software failures caused by environmental factors or equipment aging.

Definition 3.13. Attack disturbance refers to random or nonrandom faults, errors, or failures caused by the attacker triggering the ESS problems in the system. For the convenience of analysis, it is assumed that when the attacker can accurately locate and exploit the vulnerability and backdoor in the system structure, the attack disturbance is nonrandom (which is also the case in practice); when the attacker needs to implement trial-and-error or blind attack, the disturbance is random.

The security definition in this section covers equipment dependability, service dependability, and system attack resistance. It is a security indicator to measure generalized disturbance suppression capability. For general information systems in cyberspace, the chain of action from generalized disturbances in the environment to system faults, component errors, and service failures is shown in Fig. 3.9. Generalized disturbances may activate system faults and cause component errors. When errors are transmitted to the system service interface and affect the correctness of the service, the system will fail, which then may lead to the faults in other information systems related. Because disturbances, faults, and errors cannot be completely avoided, the defense route based on ESS coding is to correct local errors caused by generalized disturbances by creating a structural coding method in the DVR domain without relying on prior knowledge and additional security measures and to

block the development chain from faults to errors/failures caused by generalized disturbances, thereby finally reducing the failure probability of the system and comprehensively solving the intertwined function safety, network security, and even information security problems. In the next section, based on the definition given in this section, we will specifically study the similarities and differences between network security defense and dependability assurance and propose a new structural error-correction coding method that can solve the ESS problems based on reliability-related theories.

3.4.2 *Mathematical Model of ESS Defense*

This section takes the error probability (failure probability)/usability probability of the system being attacked as the core indicator and makes a quantitative security analysis under the condition of preventing unknown security threats through the corresponding model of the DVR incomplete intersection principle (IIP), so as to derive the security defense scheme of DVR complete intersection and give the conclusion of corresponding quantitative analysis.

3.4.2.1 **Model Assumptions and Definition**

The following assumptions are given for the entire security defense process of cyberspace information systems.

Assumption 3.1. For a single exploitable vulnerability in the system, it takes the attacker time $T_s > 0$ to activate the vulnerability and generate an output error through disturbance, that is, it takes a certain time to exploit a single vulnerability; for the exploitation of multiple vulnerabilities, considering the heterogeneity of executors, the attacker needs time $T_i > 0$ to exploit these vulnerabilities collaboratively and produce common-mode output errors.

Assumption 3.2. There is a backdoor in the system that can actively contact the attacker, and the time required for the backdoor to be activated and cause a system error is $T_s = 0$, that is, no time is needed to exploit the backdoor; for a backdoor that requires internal-external collaboration, considering the heterogeneity of the executors, the attacker needs time $T_i > 0$ to generate an output error, that is, it takes some time for the backdoor to be collaboratively exploited.

Assumption 3.3. The system components that implement various functions/services in the system are called executors, which are assumed to vulnerabilities and backdoors at the same time. The output errors/failures of the executor caused by generalized disturbances can be with or without memory (i.e., the effects after the executor is attacked can be cumulative or noncumulative), and the memory of the executor output error/failure caused by generalized disturbances can be cleared.

Assumption 3.4. Attack disturbances against a single executor arrive nonrandomly, and generalized disturbances interfere with the executor in a certain pattern (e.g., the attacker exploits vulnerabilities to achieve attacks that arrive with a negative exponential distribution, and the interference success rate continues to increase over time, or the attack arrives with a probability of 1 by exploiting backdoors).

Assumption 3.5. When the system is not cleansed and repaired, the effects of attacks on the executor can be accumulated.

Definition 3.14. The relationship between time t and the probability $P_s(t)$ of successful attacks on the executor is mathematically expressed as:

$$P_s(t) = \begin{cases} p = 1 - \frac{1 - e^{(-t+T_s)}}{1 - e^{T_s}}, & 0 \leq t \leq T_s \\ 1, & t > T_s \end{cases} \quad (3.1)$$

Based on the above assumptions, in this section we will mathematically analyze dynamic heterogeneous nonredundant, dynamic isomorphic redundant, static heterogeneous redundant, and dynamic heterogeneous redundant models under the condition of generalized disturbance.

3.4.2.2 Analysis of the Dynamic Heterogeneous Nonredundant Model

When the attacker knows an exploitable vulnerability $v_i \in Z_v^i$ in executor i , where Z_v^i is the set of exploitable vulnerabilities $Z_v^i \in Z_v = \{Z_v^1, Z_v^2, \dots, Z_v^m\}$ in executor i , and the dynamic transformation space is m , then:

At $\forall$ moment $t > 0$, the probability of unsuccessful attacks is $P_e(t) = \frac{1}{m}P_s(t)$.

In view of the existence of backdoors in the executor that actively collaborates with the outside world, no matter how large the space m is,

At $\forall$ moment $t > 0$, the probability of unsuccessful attacks is $P_e(t) = 1$.

3.4.2.3 Analysis of the Dynamic Isomorphic Redundant Model

When the attacker knows an exploitable vulnerability $v_k \in Z_v^k$ in executor k , where Z_v^k is the set of exploitable vulnerabilities $Z_v = \{Z_v^1, Z_v^2, \dots, Z_v^k, \dots, Z_v^n\}$ in executor k , and the dynamic transformation space is m , then:

At $\forall$ moment $t > 0$, the probability of unsuccessful attacks is $P_e(t) = \frac{1}{m}P_s(t)$.

In view of the existence of backdoors in the executor that actively collaborates with the outside world, no matter how large the space m is, the isomorphic redundancy vulnerability ignores the ruling mechanism through collaboration, and,

At $\forall$ moment $t > 0$, the probability of unsuccessful attacks is $P_e(t) = 1$.

3.4.2.4 Analysis of the Static Heterogeneous Redundant Model

According to Definition 3.14, the attacker implements vulnerability or backdoor-based collaborative attacks with a probability of $P_s(t)$, and the time for the exploitable vulnerability in executor r to be successfully exploited is T_s^r , that is:

At $\exists$ moment $t = T_s^r$, the attack arrives, and at $\forall$ moment $t > T_s^r$, the probability of output error of the architecture without feedback repair is $P_e(t) = 1$.

So, when the attacker attacks m executors one by one, the failure time of executor i is $T_s^i \in T_s = \{T_s^1, T_s^2, \dots, T_s^m\}$, that is:

At $\exists$ moment $t = T_s^f$, $T_s^f \in T_s$, invalidating the ruling, or $T_s^f = \max\{T_s^1, T_s^2, \dots, T_s^m\}$ makes all executors fail.

Therefore, when $t > T_s^f$, the probability of unsuccessful collaborative attacks is $P_e(t) = 1$.

3.4.2.5 ESS Structure Model Analysis

Through the above analysis of security defense solutions corresponding to the incomplete intersection of the D domain, V domain, and R domain, the defense against unknown security problems will inevitably develop in the direction of the integration of dynamics, variety, and redundancy. The following is a quantitative security analysis of the ESS architecture with complete intersection of the three domains. Figure 3.10 is a mathematical model of the endogenously safe and secure

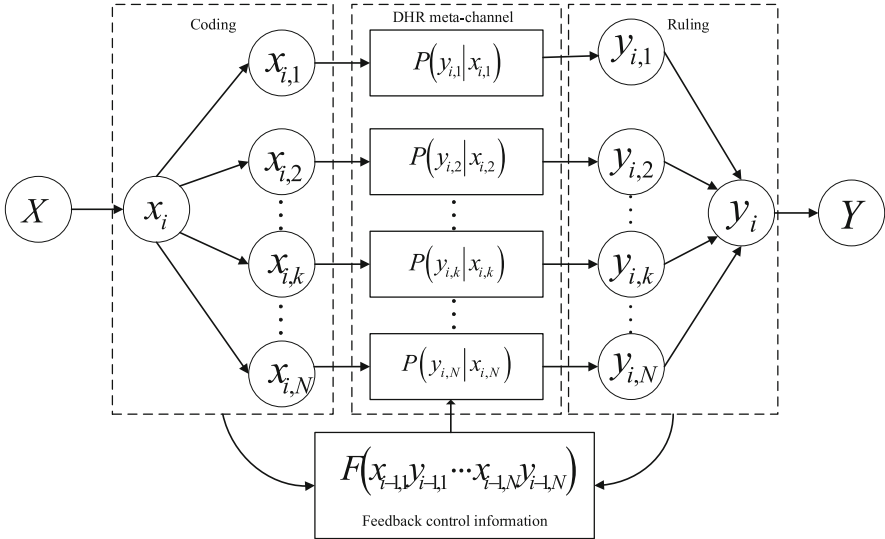


Fig. 3.10 The mathematical model of endogenously safe and secure transmission and processing structure

transmission and processing structure created based on the DHR architecture model illustrated in Fig. 3.6.

If the attack arrives at a rate of $\lambda(t) > 0$, then for $k \geq 1$ groups of meta-channel combinations, it is necessary to deploy mechanisms on heterogeneous meta-channels against collaboratively triggered vulnerabilities and backdoors. It is assumed that there are mathematical relationships between probability $P_c(t)$ and time t for the following three types of executors to be successfully attacked in collaboration:

$$P_c(t) = \begin{cases} p = 1 - \frac{1 - e^{(t-T_c)}}{1 - e^{-T_c}}, & t \leq T_c \\ 1, & t > T_c \end{cases} \quad (3.2)$$

$$P_c(t) = \begin{cases} p = 1 - \frac{1 - e^{(T_c-t)}}{1 - e^{-T_c}}, & t \leq T_c \\ 1, & t > T_c \end{cases} \quad (3.3)$$

$$P_c(t) = \begin{cases} p = \frac{t}{T_c}, & t \leq T_c \\ 1, & t > T_c \end{cases} \quad (3.4)$$

It is advisable to deploy differential-mode meta-channels with a probability of $P = \frac{k}{C_M^N}$ for the first time.

It can be proved that the ESS architecture based on DHR and feedback control makes the structure and memory of the system uncertain and ensures the randomness of the failure rate of the system in the entire time domain [1].

3.4.3 ESS Coding Existence Theorem

ESS requires dependable or continuously correct processing of information under generalized disturbances in a channel with ESS problems, memory, and computing power. Among them, ESS problems include known/unknown vulnerabilities, backdoors, and dependability faults of the system; generalized disturbances include random/nonrandom disturbances such as manmade attacks, natural environmental impacts, and uncertain failures. Errors caused by generalized disturbance activating ESS problems include random/nonrandom errors. For memory channels that meet the requirements of DVR complete intersection, Lemma 3.1 and Lemma 3.2 of ESS coding below respectively prove the randomness of errors caused by random/nonrandom generalized disturbances at the system level. ESS coding Theorem 3.1 proves the equivalence of this theorem and Shannon's second theorem under the condition of random disturbance of memoryless channels. Shannon's second theorem is no longer valid under the condition of random/nonrandom disturbance of meta-channels with memory. Theorem 3.2 of ESS coding proves the existence of ESS under this condition, that is, there is a memory channel structural coding

method that meets the requirements of DVR complete intersection, and the method can make the average error probability of system output arbitrarily small under generalized disturbances (the probability of successful disturbances approaches 0 infinitely, and the probability of system usability approaches 1 infinitely).

3.4.3.1 Randomness Lemma of Memory Channels with Random Disturbances

It has been proven that the ESS architecture scheme based on DHR and feedback control makes the system structure and memory uncertain [1], which ensures the randomness of the system failure in the entire time domain.

The randomness lemma of memory channels is as follows:

Lemma 3.1. According to the conditions of memory meta-channels with random disturbances, the randomness of failure of the ESS architecture is analyzed as follows:

When the disturbance of the meta-channel randomly arrives:

At $\forall$ moment t , the probability that the disturbance successfully causes an error in the meta-channel output is $0 < P_e < 1$.

At $\exists$ moment $t = T_s$, the random disturbance successfully causes an error in the output of the meta-channel. In addition, the meta-channel has memory, then:

At $\forall$ moment $t > T_s$, the probability of meta-channel output error is $P_e = 1$.

It can be proved that the ESS architecture scheme based on DHR and feedback control can ensure that system failures in the entire time domain are random under the condition of random disturbances of memory meta-channels [1].

3.4.3.2 Randomness Lemma of Memory Channels with Nonrandom Disturbances

In view of the condition of memory meta-channels with nonrandom disturbances (manmade disturbances), the randomness of failure of DHR channels is analyzed as follows.

Lemma 3.2. When the disturbance of the meta-channel arrives nonrandomly:

At $\forall$ moment $t > T_e$, the probability that the disturbance successfully causes an error in the output of the meta-channel is $P_e = 1$.

At $\exists$ moment $t = T_s$, the random disturbance successfully causes an error in the output of the meta-channel. In addition, the meta-channel has memory, then:

At $\forall$ moment $t > T_s$, the probability of meta-channel output error is $P_e = 1$.

It can be proved that the ESS architecture scheme based on DHR and feedback control can ensure that system failures in the entire time domain are random under the condition of nonrandom disturbances of memory meta-channels [1].

3.4.3.3 The First Theorem of Existence of ESS Coding

The generalized disturbance of the meta-channel arrives randomly. For any random disturbance, the probability of the meta-channel output being wrong is $P_e < 1$. Due to the memoryless nature of the meta-channel, the probability of meta-channel output being wrong is $P_e(t) < 1$ for a random disturbance at any time t . Therefore, under the condition of random disturbance of memoryless meta-channels, the n meta-channels in the ESS architecture satisfy the constraint premise of Shannon's second theorem that the channel noise is random and the channels expand n times without memory [12].

The sample space for input request X is $x = \{0,1\}$, and the sample space for output response Y is $y = \{0,1\}$. Based on the probability theory and mathematical statistics methods, the first theorem of ESS can be proved [1].

Theorem 3.1. Disturbance (noise) arrives randomly, and the processing capacity of the discrete memoryless executor (meta-channel) $[P(y|x)]$ is C . If the transmission rate of the input request is $R < C$, as long as the number of executors n is large enough, an input processing set consisting of $M = 2^{nR}$ sequences with a processing efficiency of R can always be found in the input coding set X^n , and the average probability of output errors can be arbitrarily small under certain rules for strategic ruling.

In the case of memoryless channels with random disturbances, the channel noise is random, and all the meta-channels in the architecture are memoryless. Shannon's second theorem requires that the channels be extended n times without being memorized, the noise of each extended channel be random, and all channels be memoryless. Therefore, in terms of the conditions to be met, the first theorem of ESS under the condition of memoryless channels with random disturbance is equivalent to Shannon's second theorem under the condition of memoryless channels with random disturbance [1, 12].

3.4.3.4 The Second Theorem of Existence of ESS Coding

Theorem 3.2. Disturbance arrives randomly or nonrandomly, and the processing capacity of the discrete memory channel $[P(y_s|x_s)]$ with the DHR and feedback control structure is $C(t) > 0$, where $\forall t > 0, C(t) \in [C_s, C_0]$. If the transmission rate of the input request is $R(t) < C(t)$ at moment t , as long as the number of executors n is large enough, an input processing set consisting of $M = 2^{nR(t)}$ sequences with a processing efficiency of $R(t)$ can always be found in the input coding set X^n , and the average probability of output errors can be arbitrarily small under certain rules for strategic ruling.

The first theorem, the second theorem, and two lemmas of ESS coding together constitute the ESS-existence theorem (ESS-ET). It puts forward and proves the

existence of ESS structure error-correction coding that can make the probability of successful generalized disturbances infinitely small.

Under the condition of generalized disturbances including human attacks, probabilistic and statistical methods and the theory of existence of ESS coding are used to prove the feasibility of the basic method of realizing system service dependability based on DHR structural error-correction coding. Therefore, ESS coding is generalized dependability coding, which enlightens us that we can learn from traditional reliability-related methods and indicator systems for the quantitative design, testing, and evaluation of ESS systems. It can be inferred that the usability probability under the condition of generalized disturbances is the leading indicator of ESS systems. Furthermore, supporting indicators related to ESS functions (unknown threat perception, defense functions, etc.) and performance (ESS service quality, etc.) can be put forward. Then, the ESS of information systems can be quantitatively designed by the method of top-down, layer-by-layer decomposition or can be quantitatively evaluated by bottom-up, weighted-aggregation white box plug method. In short, the theory of existence of ESS coding lays a mathematical foundation for the defense against unknown threats in cyberspace and points out the development direction for the quantitative design, testing, and evaluation of ESS systems.

3.4.4 Strategic Ruling Feedback Control and Common-Mode Disturbance

The DVR structure can eliminate the random disturbance effect by initializing or cleansing heterogeneous executors under the condition of random disturbance. Under the condition of manmade disturbance, if the effect of common-mode attack is obtained by the attacker, the rate of success of common-mode attacks will increase cumulatively, that is, once the same heterogeneous executor set appears randomly next time, the attacker can reproduce the effect of common-mode attacks in the same way in a certain period of time. The following is a mathematical analysis of the limitations of the DVR structure in resisting unknown threats.

3.4.4.1 DVR Model and Common-Mode Analysis

Let us assume that the attacker launches a collaborative attack on executor i in the DVR structure with a probability of

$$P_s^i(t) = 1 - \frac{1 - e^{(-t+T_s^i)}}{1 - e^{T_s^i}} \quad (3.5)$$

wherein the total number of heterogeneous executors is $m > 2n + 1$ and the number of services is $2n + 1$. Then,

At $\exists$ moment $t = T_s^r$, the attack arrives. At $\forall$ moment $t > T_s^r$, the probability of output error of executor r is $P_s^r(t) = 1$.

So when the attacker launches cumulative attacks on $2n + 1$ service executors, the time of failure of executor i is $T_s^i \in T_s = \{T_s^1, T_s^2, \dots, T_s^{2n+1}\}$. Then

At $\exists$ moment $t = T_s^f$, $T_s^f \in \mathbf{T}_s$, causing the ruling result of the current service executor to be wrong, or $T_s^f = \max\{T_s^1, T_s^2, \dots, T_s^{2n+1}\}$, causing all service executors to make errors and result in wrong results.

Therefore, when the dynamic transformation time is $T \geq T_s^f$, the effects of current common-mode attacks can be accumulated, and the rate of success of the attacker's next attack on the DVR structure is

$$P_e(t) \geq \frac{C_{n+1}^{n+1} C_{m-n-1}^n}{C_m^{2n+1}} = \frac{(m-n-1)!(2n+1)!}{m!n!} \quad (3.6)$$

When the number of services is $2n + 1 = 3$ and the total number m is 4, 5, 6, and 10, respectively, once a common-mode attack successfully occurs, the attacker will be able to successfully implement the next common-mode attack with a high probability of $P_e(t) \geq 0.5, 0.333, 0.2$, or 0.067 .

It may be assumed that the attacker successfully implements a common-mode attack with a probability of $P_c = \omega$ on the DVR structure with a redundancy of $2n + 1$, where ω is any positive number > 0 , then the probability of a common-mode incident occurring after r attacks is $P_c = 1 - (1 - \omega)^r$. When r is large enough, $P_c \rightarrow 1$, and therefore, after q rounds of r attacks, $P_c^q = (1 - (1 - \omega)^r)^q \rightarrow 1^q$, where q is an integer > 0 . It can be seen that for a DVR structure with a total number of dynamic transformations of m (m is any integer > 1), a service redundancy of $2n + 1 < m$, and a common-mode disturbance probability of $P_c > 0$, q rounds of cumulative common-mode incident may occur in the whole time domain with a high probability.

When the attacker's ability is strong enough, the probability of complete common-mode errors can approach $\frac{q}{C_m^{2n+1}}$ through q rounds of accumulation. So, the condition of $q > 0$ makes $\frac{q}{C_m^{2n+1}} > \delta$, where δ is the acceptable probability of system errors. Therefore, the DVR structure cannot solve the problem of cumulative effects of common-mode attacks, and this solution is difficult to meet the high security requirements of the system.

To overcome the limitations of the DVR structure, by analyzing the dynamics, variety, redundancy, and ruling characteristics of the DVR structure, it is found that only feedback control based on strategic ruling (SR-FC) has the ability to perceive and locate known/unknown disturbance effects, thereby achieving immunity and protection against attacks, and this leads us to the following argument.

3.4.4.2 SR-FC Is the Necessary and Sufficient Condition and Proof for DVR Structure to Solve the Problem of Unknown Disturbances

Proposition: When and only when a DVR structure has the SR-FC function, it will have the ability to solve the problems of unknown differential-mode and common-mode attacks at the same time.

Sufficiency: When the DVR structure has the SR-FC function, it will have the ability to solve the problems of unknown differential-mode and common-mode attacks at the same time.

Demonstration: When the DVR structure has the SR-FC function, the accumulation of attack effect can be eliminated by feeding back the effect of unknown threats, thus ensuring that the system has a strong ability to resist unknown differential-mode or common-mode attacks. Relevant proofs have been given in Sect. 3.4.3. For instance, the simple differential-mode state can be eliminated through feedback control; the perceptible $n - 1$ common-mode state can be eliminated through strategic scheduling identification and feedback control; and the n -mode state can be forced to activate the system loop by external random disturbance instructions, and then it can be identified and eliminated by multiple rounds of feedback control (from n -mode to $n - 1$ mode and then to differential modes until the arbitrator returns to a stable state). Therefore, sufficiency is established.

Necessity: When the DVR structure has the ability to solve the problems of unknown differential-mode and common-mode attacks at the same time, the DVR structure has the SR-FC function. (inverse proposition: if the DVR structure does not have the SR-FC function, it will not have the ability to solve the problems of unknown differential-mode and common-mode attacks at the same time).

Argument: The necessity demonstration can qualitatively analyze and demonstrate the original proposition or quantitatively analyze and demonstrate the negative proposition.

Original proposition: When the DVR structure has the ability to solve the problems of differential-mode and common-mode attacks at the same time, it must perceive and handle these attack incidents. Obviously, perception cannot be realized in a single D domain or V domain, and perception in R domain can be divided into executor's own perception and ruling perception. The executor's own perception requires the introduction of additional protection functions such as firewalls and intrusion detection. These introduced functions rely heavily on prior knowledge, making it impossible to perceive and effectively defend against unknown threats. The executor's ruling perception uses multiple functionally equivalent heterogeneous executors to perceive the relative differences in the effects of unknown differential-mode or common-mode attacks. This function does not rely on prior knowledge, so strategic ruling perception based on the relativity algorithm is necessary. Once the DVR structure based on strategic ruling identifies an unknown differential-mode and perceivable common-mode attack, it will inevitably feed the arbitrator status information back to the heterogeneous redundant scheduling

module, thereby effectively shielding the attack and eliminating its cumulative impact of the attack. Therefore, the SR-FC function is necessary, that is, the necessity is established.

Inverse proposition: When the DVR structure does not have the SR-FC function, the mathematical analysis of the limitations of the structure in resisting unknown threats in the opening part of Sect. 3.4.2 has demonstrated that the DVR structure cannot solve the problem of cumulative differential-mode or common-mode attacks based on unknown vulnerabilities and backdoors by changing its own dynamic transformation space, isomorphic common-mode disturbance rate, and redundancy. Therefore, the necessity is established.

To sum up, when and only when a DVR structure has the SR-FC function, it will have the ability to solve the problems of unknown differential-mode and common-mode attacks.

3.5 DHR Architecture

3.5.1 Prerequisites of the DHR Architecture

Before elaborating on the principle of the DHR architecture, it is necessary to clarify its prerequisites: ① Because of the need to deal with nonartificial random disturbances, uncertain perturbations, and inhibit any unknown network attacks, one of the necessary conditions for the existence of ESS is that there is a feedback control mechanism that can realize the complete intersection of the three elements: D, V, and R. ② In theory, if it can be guaranteed that any potential attack resources of the input distribution components cannot be successfully exploited, then all the soft and hard components in the structure or operating scenarios are allowed to have differential-mode common ESS problems. ③ Network attacks include both external attacks based on internal vulnerabilities/backdoors and internal-external collaborative attacks based on dark functions such as dormant virus Trojans. ④ There are no interactive communication links and connection between reconfigurable operating scenarios (executors), and they are physically or logically independent from each other. ⑤ Unless the output state of the strategy arbitrator is expected, the feedback control loop always tries to restore the arbitrator state to the initial equilibrium state or within the designed fluctuation range by changing the current operating scenario, ruling algorithm, executor structure, or the way of resource allocation. ⑥ Under a given attack condition, as long as the arbitrator can perceive that there is a differential-mode expression between the output vectors or related states in the current operating scenario, a suitable substitute for the current problem scenario can always be found by iteratively replacing various operating scenarios. ⑦ The response time of feedback control or the time of iterative convergence of diversified defense scenarios is usually shorter than the interval of attacks. ⑧ The physical or logical boundary formed by the input agent and error-correction output is the attack surface of the DHR architecture.

For the sake of brevity, the “DHR architecture” wherever mentioned in the subsequent chapters of this book meets the above prerequisites by default, unless otherwise stated.

3.5.2 DHR Architecture and Its Working Principle

The abstract model of the DHR architecture is shown in Fig. 3.6. Let us assume that a DHR architecture has a distributed resource pool composed of M heterogeneous executors with an equivalent function P . The combination of heterogeneous executors with equivalent functions forms an operation scenario (essentially a physical or logical, physical or virtual, centralized or decentralized configuration corresponding to a reconfigurable or state-changeable system). If there are $k(k < V)$ heterogeneous executors in each distributed operating environment, then in the resource pool, there will be C_M^k operating scenarios for strategic scheduling. Any distributed operating environment that currently hosts or provides the service function P of the target object is called the current operating environment, in which the heterogeneous executor set S is composed of k heterogeneous executors whose equivalent function is P , and L is a component set that increases the diversity but does not change the function of S , and satisfies $P = f(S) = f(S_1 \cup S_2 \cdots \cup S_k) = f((S_1 \cup L_1) \cup (S_2 \cup L_2) \cdots \cup (S_k \cup L_k))$ and $S_1 \neq S_2 \cdots \neq S_k$, $L_1 \neq L_2 \cdots \neq L_k$, but L_i can be empty, and $P = P_i = f(S_i) = f(P_1 \cup P_2 \cdots \cup P_k)$, and $P_i \neq f(L_i) \neq f(L_1 \cup L_2 \cdots \cup L_k)$. In a mathematical sense, P is a function of the structure of the current executor set S , and all C_M^k sets of S have equivalent function P . It should be emphasized that the expression of function P will not be affected whether structure L is added to or deleted from structure S , but it will change the current operating environment or the apparent structure of S .

In the DHR architecture, the input agent or distribution link can be implemented by means of centralized control or distributed processing (hard-wired or offline programmable logic can also be used to reduce the exploitability of vulnerabilities), and the input sequence needs to be distributed to the reconfigurable executor set S with equivalent function P according to the instruction of the feedback controller. Each heterogeneous executor S_i with function P in this set should, with a high probability, be able to independently complete the technically designed function/performance of the cyber-physical system or digital product and can generate multimode output vectors or related states that meet the given semantics, grammar, and even pragmatics (context). The following points need to be emphasized: ① The multimode output vector or state mentioned here is in a broad sense, that is to say, in the current operating environment, any direct or indirect response information generated in the executor set S , or detection alarm information strongly related to the processing process and environmental state can be used as the output vector (this will not be reiterated in the subsequent chapters, unless necessary). ② Strategic ruling can adopt either centralized processing or distributed processing. It generates

the current ruling strategy according to the parameters and algorithm given by the system, judges the compliance of the output vector content or state of the multimode executor, and instructs the error-correction output component to form an output response sequence. Once an unexpected ruling state is found, the feedback control link will be activated. ③ Once the feedback control link is activated, it will decide whether to send an instruction to the input agent or the distribution link to replace and migrate the current “differential-mode state” executor or instruct the suspected problem executor to perform online/offline cleansing and recovery operations, including triggering related background processing functions, or perform diversified operations such as reorganization/reconfiguration/reset on the abnormal executor under the condition of functional equivalence. This feedback control process will not be suspended until the current noncompliance state of output vector disappears in the link of strategic ruling, or the frequency of such situations is lower than a given threshold. ④ The dynamic iterative convergence process also includes consciously or actively changing the strategic ruling algorithm, so as to multidimensionally confirm or verify the relative correctness of the ruling result affected by the blind zone of ruling due to unperceivable common-mode escape ($n: 0$). ⑤ In principle, the relativity ruling mechanism of the DHR architecture does not exclude the majority of executors with the same output vector. In a small or even tiny probability, there are still common-cause failures or common-mode escapes (i.e., what the arbitrator judges as differential-mode problems are actually common-mode ones). It is, therefore, necessary to adopt diversified means of strategic ruling based on an iterative mechanism rather than just a simple majority-based ruling algorithm. ⑥ The reasons why the “problem executors” must be strategically selected for offline cleansing and recovery, or diversified changes in the current operating environment should be realized by means such as reorganization, reconstruction, and iterative replacement are as follows. First, the attack chain in the current operating scenario should be forcibly destroyed or changed. The second is to transfer or transform the spatial-temporal dimension of security contradictions of the same nature in a diversified operating environment. In other words, regardless of whether the cause or mechanism of the differential-mode state can be discovered in real time, in theory, as long as “every time a differential-mode presentation is found, the current operating environment will be structurally transformed or encrypted under the condition of functional equivalence,” that is, under the conditions of $P = P_1 \cup P_2 \cdots \cup P_k$ and $S = S_1 \cup S_2 \cdots \cup S_k$, the scheduling (or reset, reconstruction, reconfiguration, etc.) of any S_i or the addition or deletion of structure L that has nothing to do with function P can be regarded as structural re-encryption of the current operating environment, although common ESS problems cannot completely disappear because of the scheduling of S_i . ⑦ To prevent perceivable or unperceivable common-mode escape from becoming a steady-state incident (although its probability is small or very small), it is necessary to introduce a backward verification mechanism and an appropriate relief mechanism, in an effort to obtain a cyber-resilient effect that “even if the attack is successful, it may only be once or the duration is within a tolerable range.”

In general, it can be considered that the stable working state of the DHR architecture (as shown in Fig. 3.11) has the same or similar dependability and

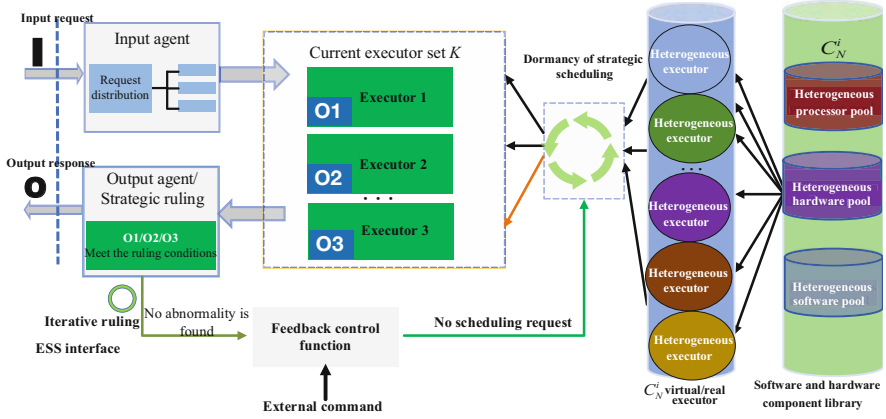


Fig. 3.11 Steady state of the DHR architecture

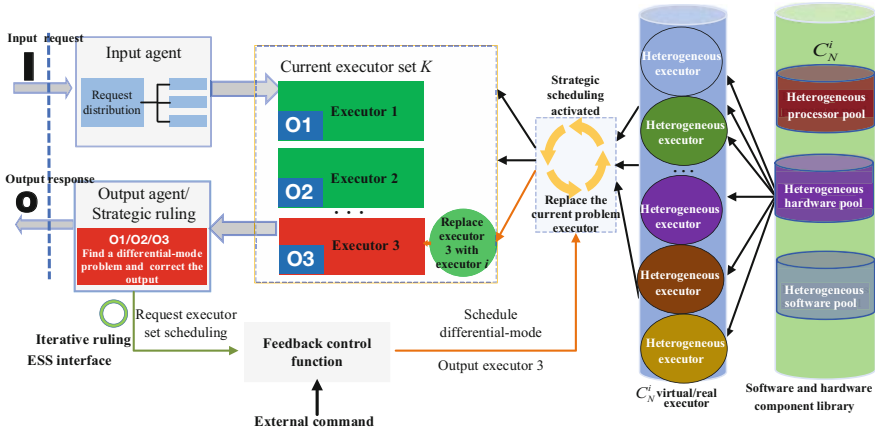


Fig. 3.12 Temporary steady state of the DHR architecture

anti-attack performance as the DRS operating environment. However, when a differential-mode state is perceived, the DHR architecture will automatically enter a temporary steady state (as shown in Fig. 3.12), the feedback control mechanism of dynamic iteration and convergence will be activated, and the current operating environment will be “structurally encrypted” until the abnormal state is no longer found in the link of strategic ruling and the system returns to the stable working state. In other words, once the DHR architecture perceives that there is, for whatever reason, a differential-mode (or perceivable common-mode) output state in the current operating environment, the feedback control mechanism will try to strategically schedule the elements in the current executor set or cleanse, restore, reset, and

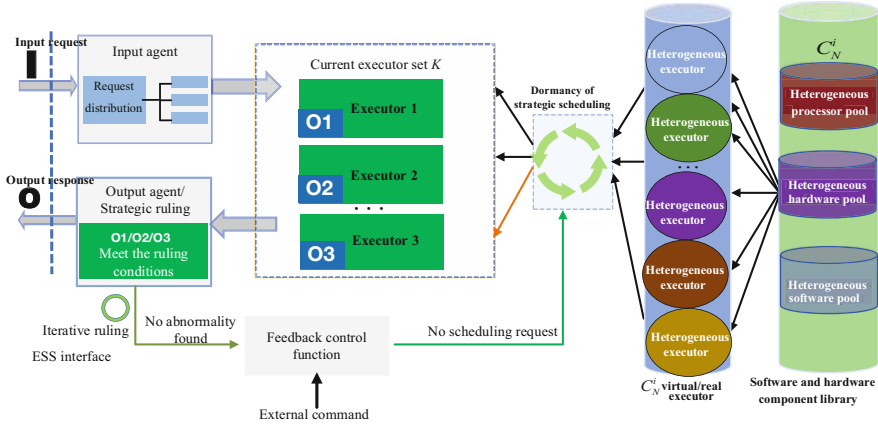


Fig. 3.13 The DHR architecture is forcibly disturbed by external commands

restart the problem executor itself to change the running status, or activate the multidimensional dynamic reconfiguration mechanism under the condition of functional equivalence, and automatically select the appropriate executor elements (or operating scenarios) to eliminate the differential-mode expression in the arbitrator. This process is equivalent to one or more “re-encryption” operations on the implementation structure in the current operating environment, and the resulting “double-blind effect” can maintain the entropy nondecreasing state in the structure. Therefore, the DHR architecture has the attribute of ESS in mechanism. Obviously, DHR, as a trinity of high dependability, high trustworthiness, and high usability not only significantly increases the uncertainty of the creation and use of attack chains, seriously undermines the effectiveness and reliability of attack operations, but also completely subverts the theoretical premise that any trial-and-error attack must keep the background unchanged and brings endogenous and universal functions of “mission assurance” and a “trusted execution environment” to any cyber-physical system or digital device.

It should be pointed out that DHR can also trigger the feedback control loop to generate the corresponding operation scenario scheduling operation according to the scheduling instruction formed by internal and external control parameters (as shown in Fig. 3.13). The purpose is to prevent attackers from using the blind zone of the “superposition state” of strategic ruling (pure relativity ruling cannot find common-mode escapes, e.g., the possibility of common-mode escape cannot be completely ruled out based on only the ruling result of $n: 0$ or $0: n$) and implementing hidden and maintainable standby collaborative escape through implanted virus Trojans, such as the problem of tunnel crossing [13]. After an imperative disturbance, the feedback control mechanism is forcibly activated, and the current operating environment is constructively reset or reorganized. If there is a hidden escape state, it will be found and resilience recovery operations will be performed. In this process, external or

internal control parameters can be derived from a random function generator, or from a hash value formed by the uncertain state information in the target system, such as the current number of active processes, CPU occupancy, memory allocation, or input/output traffic statistics. It is expected that the imperative disturbance is not so regular.

At this point, it is not difficult for readers to find that the DHR architecture not only inherits the original intrusion-tolerance and error-tolerance attributes of DRS but also effectively avoids the latter's genetic defects of lack of dynamics and strategic ruling. The DHR architecture can also differentially deploy existing or additional network security measures in redundant executors or operating scenarios to exponentially enhance its own cyber resilience and resistance to unknown network attacks [3]. This forces the attack complexity to rise from the threshold of "static heterogeneous multitarget coordinated attack under noncollaborative conditions" of DRS to the height of "dynamic heterogeneous multitarget coordinated attack under noncollaborative conditions." The exponential increase in the difficulty of "blind coordinated" attack makes it difficult for attackers or infiltrators to find a reliable common-mode escape path by advanced means such as intelligent trial and error, fuzzy exclusion, or tunnel crossing.

In particular, the uncertain structural effect from the attacker's perspective (similar to Heisenberg's uncertainty principle that it is impossible to determine the position and momentum of quantum at the same time with the same accuracy, and only one of them can be accurately determined) greatly reduces the success rate of trial-and-error attacks based on the common ESS problems of DHR. Unless such attacks can be "successful in one strike," once the link of strategic ruling perceives a differential-mode output state in the current operating environment, the feedback control mechanism will immediately change the current operating environment (including the intersection of dark functions), so that the scenario information or initial results obtained by the attack at moment t will no longer be usable or inheritable at moment $t + x$.

In other words, while shielding any differential-mode output state, DHR theoretically (not technically) changes the current operating (or service) environment in a dynamic iterative convergence manner according to a given strategy, creating a mimic disguise fog (MDF) [11] similar to that existing in nature (as shown in Fig. 3.14), so that any trial-and-error tactics can neither evaluate the attack effect at time t nor meet the precondition of environmental invariance. Because dynamics and randomness are most likely to destroy or interfere with attacks that require consistency or coordination (especially under noncollaborative conditions or lack of synchronization mechanism), the closed-loop feedback control mechanism of the DHR architecture can give full play to the interweaving and overlapping effects of basic security elements such as dynamics, randomness, and diversity and effectively avoid the blind zone of majority-based ruling of DRS, thus having incomparable advantages in dealing with trial-and-error, dormant-standby, or tunnel-crossing attacks.



Fig. 3.14 Striped octopus—mimic disguise fog

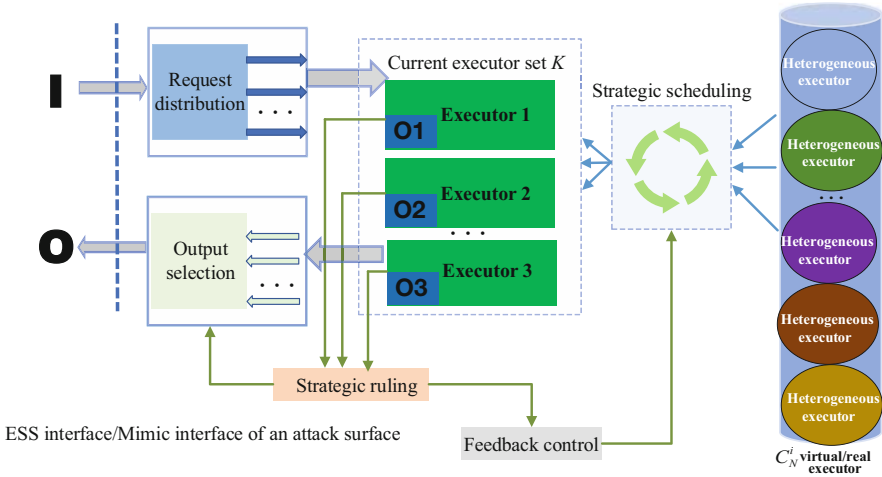


Fig. 3.15 DHR architecture and its attack surface

3.5.3 Attack Surface of the DHR Architecture

As shown in Fig. 3.15, the attack surface of the DHR architecture consists of input request distribution channel and error-correction output channel, while other components are inaccessible or indirectly accessible. Usually, it is difficult for an attacker (or even the product designer) to know which heterogeneous executors are working

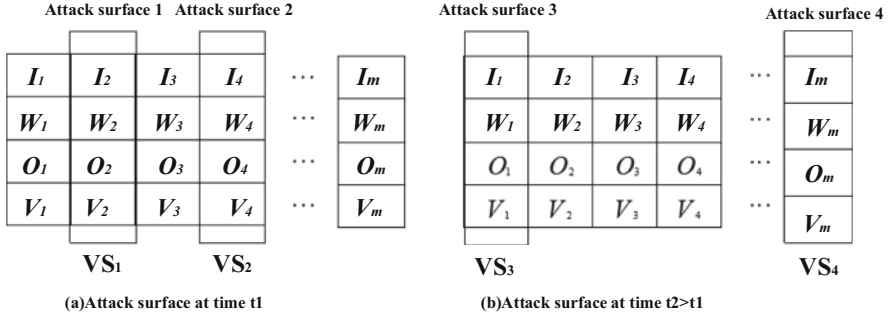


Fig. 3.16 An example of the DHR architecture's "mobile attack surface"

in parallel and the specific configuration of software and hardware resources in the current operating environment, let alone how many executors are available for scheduling or the reconfigurable operating scenarios in the background. The iterative mechanism of strategic ruling makes the attacker neither aware of the ruling algorithm nor the ruling parameters. The attacker cannot understand the feedback control algorithm, scheduling strategy, and the processing mechanism of differential-mode or common-mode state by scanning and sniffing. It is even difficult for attackers to figure out what kind of CPU, operating system, database, middleware, and other software/hardware resources are used in the current operating environment, and whether additional security software such as intrusion detection, firewall, honeypot, sandbox, antivirus software, and threat awareness means are configured. In a word, there is an unpredictable or uncertain apparent expression inside the attack surface. For example, multiple heterogeneous virtual machines with equivalent functions can be formed using the multivirtualization technology, or a number of functionally equivalent nonhomologous software and hardware elements to form a component pool, and select the executors or diversified operating scenarios needed to generate the current service set through random extraction, dynamic call, and random combination. Even if the attacker has some prior knowledge about the target object, the resources available to the attacker are uncertain and indistinguishable in the spatial-temporal dimension due to the diversity or variety of software and hardware resources in the current operating environment, which shows that the attack surface is moving irregularly based on the entangled state. Figure 3.16 shows that the configuration of soft and hard resources in the current operating environment will change dynamically with the frequency of differential-mode state perception of strategic ruling, which makes the apparent attack surface of the target system have uncertain changes in the nature of "structural coding or structural encryption."

This "structural coding or structural encryption" effect of the DHR architecture can be summarized as follows:

1. In the DHR architecture, the system as a whole presents an irregularly changing attack surface and unpredictable attack channels. At a certain moment or in a certain period of time, an attack surface or a possible attack channel that the

attacker “locks” based on some prior knowledge is “not online or unreachable,” or it may be that the request distribution link directs the probe/attack packet to the nontarget executor. Therefore, attackers must decrypt or deconstruct “structural encryption or structural coding” to lock the attack channel and target the exploitable resources. In this way, it is difficult to achieve an attack by relying on traditional cyberattack theories and means.

2. The DHR architecture limits the harm that may be caused by 0-day vulnerabilities or unknown backdoors. The iterative convergence mechanism of DHR makes it impossible to achieve differential-mode or perceivable common-mode escape in “internal-external collaborative” network attacks against an executor in the current operating environment, and it is also difficult to maintain common-mode escape using the blind zone of relativity ruling. This is because once the SR-FC mechanism finds a differential-mode state in the current running environment, the related execution scenario and even the whole operating environment may be cleansed, restored, reset, or replaced according to certain strategy. Even if there is no abnormal situation, the system will carry out periodic or nonperiodic dynamic cleansing, reorganization, reconstruction, and other operations in the current operating environment according to some preventive strategy to block the nonperceivable common-mode escape in extreme cases. In other words, even if an attacker can build an attack chain based on some 0-day or Nday vulnerabilities, backdoors, and other “spear-level” attack resources in the architecture, it is still difficult to break through the structural encoding or structural encryption mechanism to obtain the result of a “blind” collaborative attack across executors or service scenarios, thereby making these attack resources useless. This effect can be verified through “white box plug test” mentioned in Sect. 3.8.
3. Different from defense methods that only focus on the dynamics or randomness of a specific aspect within the system or components (such as instruction randomization and address/port dynamization), DHR enables diversified technical products or open COTS-level soft and hard components to achieve the generalized robust control effect of integrating dynamics, diversity, and randomness through the architectural effect using the iterative convergence control mechanism based on multidimensional dynamic reconfiguration and scheduling under the condition of strategic ruling. This, because of structural encryption, makes both the defects of software/hardware code design and deliberately implanted software/hardware attack codes in the architecture difficult to become resources that attackers can collaboratively exploit.
4. Because of the existence of the SR-FC mechanism, the DHR system will always make full use of strategic scheduling and “structural encryption” of multidimensional dynamic reconfiguration to give full play to the role of generalized dynamics, so as to keep the frequency of occurrence of differential-mode state in the current operating environment within a set threshold. Correspondingly, the attack surface of the DHR system will converge with the activated feedback control mechanism until it reaches the expected temporary steady state. The action results of any “fault perturbation, attack disturbance or external command” in the feedback loop will eventually tend to this state. Therefore, the

movement pattern of DHR's attack surface is also strongly related to the above disturbance factors and has both uncertain effects from the attacker's perspective and entangled properties that are difficult to separate.

In fact, multiple superimposed attack surfaces are constructed (just like overlapping images) in the attack surface of the DHR architecture. In theory, this seems to increase the overall attack surface of the target object and seems to be contrary to the attack surface theory that "reducing the attack surface is beneficial to security." However, its iterative convergence feedback control mechanism causes these attack surfaces to appear as parallel moving or dynamically confusing attack surfaces, and the error-correction output mechanism of request distribution and strategic ruling control makes multiple attack surfaces exhibit the properties of structural encryption in the spatial-temporal dimension, which is actually equivalent to "greatly reducing the attack surface." Therefore, the DHR architecture first breaks the assumption of the classic AS evaluation theory that "the attack surface remains unchanged" and that the attack surface is static, deterministic, and similar. Second, the attacker cannot correctly direct the attack data packet (upload virus Trojans or interactive information) to the target executor, that is, it is difficult to meet the assumption that "the target attack surface is always accessible to the attacker." Furthermore, the feedback convergence mechanism of DHR based on strategic ruling requires the attacker to have the blind collaboration ability to coordinate multiple heterogeneous mobile attack surfaces to achieve nonperceivable escape under noncollaborative conditions [1]. In other words, without the ability to crack the algorithm of "encrypting the operating environment every time a differential-mode state is perceived," all attempts of network attack seem to be vain in theory.

3.5.4 Filter Effects and Double-Blind Expression

The ESS of the DHR architecture can also be expressed as the transparency of intrinsic functions and the "black box" effect of undesired functions. The DHR architecture composed of an SR-FC loop and a reconfigurable redundant executor can theoretically be equivalent to the "band-pass filter" in a communication circuit, as shown in Fig. 3.17. By definition, a band-pass filter is a device that allows a specific frequency band to pass while shielding other frequency bands.

Intrinsic functions or services are equivalent to band-pass signals that can pass through the DHR filter, and undesired functions or services will be blocked by the DHR filter as out-of-band signals, which include both random error signals that may occur in the hardware of the reconfigurable executor and uncertain error or fault signals that may exist in the software of the executor, as well as network attack or disturbance signals implemented by attackers based on internal vulnerabilities, backdoors, and virus Trojans of the executor. From the filter's perspective, therefore, the defender only needs to know what frequency band the signal can pass, without the need to care about the origin of those extrinsic function signals. Whether the

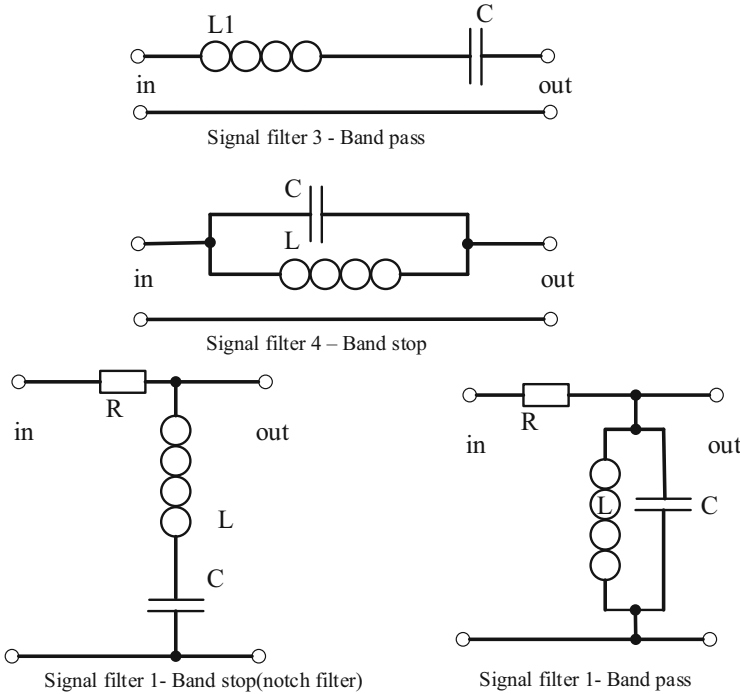


Fig. 3.17 Band-pass/band-stop filter

attacker in the position of “in” knows the specific structure or design parameters of the filter, he only knows that it is a “black box” through which no other signals can pass except intrinsic function signals. Obviously, this “double-blind” filtering effect has nothing to do with whether the attacker and defender have prior knowledge about the opposite party. This also means that even if the designer or manufacturer of a cyber-physical system with the DHR architecture intentionally or unintentionally reveals the sensitive information of the product, the attacker still cannot use this information to achieve the desired purpose of the attack. This is just like the case that buyers of commercial cipher machines cannot expect to use their products to crack other customers’ passwords.

In the next section, we will see that DHR, which does not rely on prior knowledge, has the property of structural encryption in the sense of cryptography (it is not difficult to prove that all security defense technologies that do not rely on prior knowledge have the property of encryption). This enables the endogenously safe and secure target system empowered by the DHR architecture to shield all “undesired functions” through structural encryption, except for “transparent intrinsic services.” Theoretically, when a differential-mode or perceivable common-mode state occurs in the DHR architecture, the current operating environment will undergo a “perfect safety” transformation similar to “one-time pad” or “reconfiguration carried out every time a differential-mode state occurs.” On the attack surface, all exploitable

software and hardware resources in the current operating environment, including vulnerabilities, backdoors, and virus Trojans unknown to the defender or known to the attacker, may be subjected to “structural transformation” or “environmental encryption” again under the condition of functional equivalence because of “a differential-mode disturbance” for unknown reasons, except that intrinsic services presented in plain texts remain unchanged. This not only makes it impossible for attackers to find a better cracking method than exhaustive trial and error but also makes it nearly impossible for exhaustive trial and error to achieve any desired convergence.

3.6 ESS Analysis Based on Cryptography

3.6.1 *The Revelation of Perfect Secrecy*

This section mainly studies how to resist network attacks based on the unknown ESS problems of the target object without relying on prior knowledge and then solve the multi-intertwined problems of functional safety, network security, and even information security in an integrated manner [14].

An obvious conclusion is that the apparently diversified expressions of the target object will cause great cognitive obstacles to the attacker [15], achieving discrete scene expression like “a blind man touching an elephant.” This shows that if the intrinsic functions of a cyber-physical system or digital product can be kept unchanged or “transparent” while the internal structure and operating environment are random or uncertain, then it is possible for the defender, even with a lack of prior knowledge, to reverse the one-way transparency advantage that attackers may have in the face of “unknown unknown” security threats or sabotage and cause the attacker to fall into the dilemma that he cannot create a targeted attack chain because of not recognizing the “structure of the target system” and therefore cannot launch any effective attacks or meet the goal attack. Obviously, if a target system can make the “historical attack results” irrelevant to the “next attack results” through its own endogenous structural attributes (i.e., ESS attributes), then no matter how much information about the “historical attack results” is obtained, it is not helpful for the attacker to understand the system structure and control the “next attack result.” This is the expected security effect of ESS, namely:

$$\Pr(\text{next attack result} | \text{historical attack result}) = \Pr(\text{next attack result}) \quad (3.7)$$

wherein $\Pr$ is probability. This equation means that ① “the next attack result” and “historical attack result” are statistically independent of each other, ② the mutual information between “the next attack result” and “historical attack result” is 0, and ③ it is impossible to deduce the next attack result by analyzing the historical attack result, that is, the posterior probability after knowing the historical attack result is equal to the prior probability when the historical attack result is unknown.

It is not difficult to find that the “unrecognizable system structure” expected by the ESS mechanism is very similar to the perfect secrecy defined in cryptography [8, 12], and both of them have the same connotation of “A and B being independent of each other” and “no need for the support of prior knowledge.” This shows that some characteristics of the ESS mechanism may be similar to those of classical cryptography, and it is possible to obtain security effects similar to those of classical cryptography under certain conditions. To further discuss this problem, we need to study the definition of perfect secrecy at first.

Definition 3.15. Perfect secrecy [8, 12]—for an encryption scheme with a plaintext space of M , when the plaintext is selected from M with any distribution, and any selected plaintext $m \in M$ and any obtained ciphertext $c \in C$ ($\Pr(C = c) > 0$) satisfy the following equation, the encryption scheme is a perfect secrecy scheme:

$$\Pr(M = m \mid C = c) = \Pr(M = m) \quad (3.8)$$

This definition is consistent with Equation (3.7) in form and concept. An obvious revelation is that constructing a “system whose next attack result has nothing to do with the historical attack result” is probably similar in principle to constructing an “encryption scheme whose plaintext has nothing to do with ciphertext.”

The definition of perfect secrecy tells us that when no information about the “plaintext” can be obtained from the “ciphertext,” there is no better way to decipher it than “blindly guessing the plaintext.” It should be noted that this does not mean that the decipherer guesses the plaintext 100% wrong, but it means that the decipherer cannot find a better method than exhaustion. In fact, this is the underlying logic (core meaning) of perfect secrecy: by perfectly shielding the plaintext information from the ciphertext, it is “useless” for the decipherer to analyze the ciphertext, and there is no better means than exhaustive attacks [12].

3.6.2 A Conjecture

The above analysis shows that perfect secrecy and ESS have the same premise assumptions, such as “not relying on prior knowledge” and “not considering the attacker’s ability and attack method,” and that they also have the same expected goals. For instance, Eqs. (3.7) and (3.8) both contain the security effect that “no information about the system structure (plaintext) can be obtained from the attack target (ciphertext),” making the attacker fall into a cognitive fog (mimic disguise fog). Inspired by this, we put forward a conjecture that there are some identical or similar properties between ESS that do not rely on prior knowledge and perfect encryption that does not rely on prior knowledge.

If the conjecture is true, it can provide a new perspective to interpret the ESS mechanism with the basic principles of cryptography. More importantly, when solving cybersecurity problems, we can draw on the concept of perfect secrecy of

information security and explore a “perfect” ESS system that can maximize the cognitive dilemma faced by the attacker (i.e., to maximize randomness) and minimize the information of a defect pattern when the system defects are transparent to the attacker in one direction, while the defender does not have any prior or posterior knowledge about the attacker’s ability, attack mode, defect pattern, and attack effect, so as to ensure the intrinsic function to the maximum extent. In the face of this “perfectly safe and secure system,” even if the attacker has all the information about the target system’s defect set, he can only conduct blind trial-and-error attacks indiscriminately because he does not know the current structural status inside the system.

This goal can be summarized by the following axiom: under the premise that the target system’s set of extrinsic functional defects is one-way transparent to the attacker, and the defender lacks the information of relevant defects but can perfectly shield the information of a defect pattern while realizing the intrinsic functions, then the attacker has no better way to trigger extrinsic functional defects than indiscriminate trial-and-error attacks.

To prove the above conjecture and axiom, with the help of the perfect secrecy principle of cryptography, we transform the GFS problem into the perfect secrecy problem in the cryptographic domain and propose a perfect intrinsic functional safety (PIFS). The core idea is to use system reconfiguration as an encryption method “under the condition of intrinsic function transparency.” By randomly reconstructing the system state or resource configuration, the defender can perfectly shield the internal defect pattern of the system without knowing any information about the system’s defects and about the attacker, so that there is no stable mapping relationship between input and output, and the attacker’s advantage of one-way transparency is transformed into the double blindness of both the offensive and defensive sides. This leads to the attacker’s inability to obtain useful information from the attack input and output (i.e., the entropy of the defect pattern is not reduced), making it impossible for him to find a better attack mode other than blind and exhaustive trial-and-error attacks. On this basis, it is proved that the DHR architecture that satisfies the requirement of “one reconstruction at a time” can realize PIFS, and the necessary and sufficient conditions for DHR to realize PIFS are given. The overall logic is shown in Fig. 3.18.

3.6.3 *Perfect Intrinsic Functional Safety (PIFS)*

Based on the above axiom and conjecture, this section gives the mathematical model, definition, and physical explanation of PIFS.

3.6.3.1 **Cryptographic Model of PIFS**

First, we introduce the following concepts.

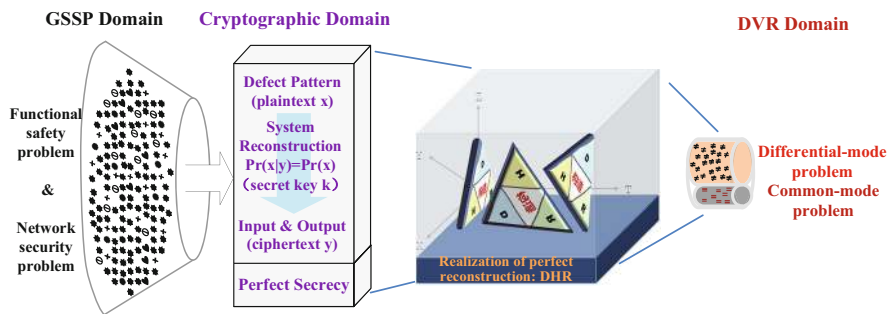


Fig. 3.18 The overall logic of PIFS

Defect pattern S_t^q : The structural spectrum of the distribution pattern of single or multiple defects in each component of the system when the system conducts the t th output is called the structural pattern of the defect (hereinafter referred to as the defect pattern).

Defect pattern space S_t^q : the value space of defect pattern S_t^q . For example, assuming that defect pattern S_t^q has three forms, a, b, and c, then $S_t^q = \{a, b, c\}$ and $|S_t^q| = 3$. Note that the system's defect pattern space can be restored, that is, S_t^q at the t th output can be restored to S_{t-i}^q at the $(t - 1)$ th output through reconstruction (including restart, reset, cleansing, reconfiguration, replacement, etc.), i.e., $S_{t-i}^q = S_t^q$ can be achieved through the reconfiguration of defect pattern.

Intrinsic function and extrinsic function: For a certain input, when the system outputs the function expected by the design, the output is called an intrinsic function, and the defect pattern at this time is $S_t^q = S_{\text{intrinsic function}}^q$; otherwise, it is an extrinsic function, and the defect pattern is $S_t^q = S_{\text{extrinsic function}}^q$.

For a given input, the current defect pattern of the system uniquely determines whether to output extrinsic functions. If the system state changes make the structural pattern of the defect uncertain, the key information about the currently existing defect will be shielded from the attacker.

On the above premise, the defect pattern S_t^q determines the output of the system when the input is determined. Since we cannot define unknown defect patterns, attackers, and attack methods, we might as well consider the worst possible case: the system's component or executor pool is fully public like the open-source community, and the defects of each executor are one-way transparent (known to the attacker, but unknown to the defender). Once the attacker knows S_t^q , the attack will be successful. Therefore, although each defect can exist independently, the combination of multiple defects, that is, the state in which the defects exist or are expressed, must be encrypted.

Correspondingly, the following inferences are made:

1. The plaintext the defender wants to protect is in defect pattern S_t^q .
2. The ciphertext known to the attacker is input x_{t-i} and output y_{t-i} at time $t - i$ ($i = 1, 2, \dots, t$) before the attacker launches the current attack.

3.6.3.2 Definition of Perfect Intrinsic Functional Safety

Definition 3.16. (Perfect intrinsic functional safety, PIFS):

If for any given ε ($0 < \varepsilon < 1$) and any input $x \in X$, the system's defect pattern at any time t satisfies:

$$\Pr[S_t^q = S_{\text{intrinsic function}}^q] = 1 - \Pr[S_t^q = S_{\text{intrinsic function}}^q] = 1 - \varepsilon_x \geq 1 - \varepsilon \quad (3.9)$$

wherein ε_x is the probability of extrinsic function when the input is x , satisfying $0 < \varepsilon_x \leq \varepsilon < 1$; and for defect pattern $S_t^q \in S_t^q$ at any time t and any input x_{t-i} and output y_{t-i} at time $t-i$ ($i = 1, 2, \dots, t$), the following condition is satisfied:

$$\Pr[S_t^q = s_t^q | x_{t-i}, y_{t-i}] = \Pr[S_t^q = s_t^q] \quad (3.10)$$

Then, the system is said to be a PIFS system with parameter ε . Assuming that the attacker has super ability and can completely and accurately obtain the system's defect pattern from input x_{t-i} and output y_{t-i} at any time in the past, the above equation can be transformed into:

$$\Pr[S_t^q = s_t^q | S_{t-i}^q = s_{t-i}^q] = \Pr[S_t^q = s_t^q] \quad (3.11)$$

wherein $S_{t-i}^q \in S_{t-i}^q$ is the defect pattern at any time $t-i$ ($i = 1, 2, \dots, t$).

According to the above definition, even if a super attacker can control x_{t-i} , observe y_{t-i} , and even deduce the system defect pattern S_{t-i}^q , it is of no practical significance for the prediction of S_t^q .

It is not difficult to find that the key to achieving the above definition is to make defect patterns S_t^q and S_{t-i}^q independent of each other, that is, the defect patterns are independently reconfigured. However, Eq. (3.10) is presented from the attacker's perspective, and the defect pattern is unknown to defender e . Therefore, it is necessary to find from the defender's perspective a "blind encryption" method that does not require prior knowledge about the defects: without changing the intrinsic functions, changes in system configuration status or software/hardware resources are used to drive changes in defect patterns and make defect patterns independent of each other at different times. This is because S_t^q is generally determined by the system structure, and system reconfiguration is autonomous and controllable for the defender and can be randomly reconfigured without affecting the "transparency" and dependability of intrinsic functions.

System state S_t and system state space $S_t: S$ are the operating environment and structure when the system makes the t th output, which determines the defect pattern S_t^q . S_t is the value space of S_t .

State control key K and system state space $\mathcal{K}: K$ determine the system state S_t at time t . $\mathcal{K}$ is the value space of K .

On this basis, if the system state and defect state change in conjunction and the following assumptions are satisfied:

Assumption 3.6. Changes in the executor are equivalent to changes in the defect state (the scheduling mechanism controls the degree of heterogeneity).

Assumption 3.7. Changes in the system state are equivalent to changes in the executor.

Then, changes in the system state or resource configuration are equivalent to changes in a defect pattern, that is, there is a “one-to-one mapping” relationship between the defect pattern and the system state. Therefore, the independence of the system state can be deduced from defect patterns S_t^q and S_{t-i}^q mentioned in Eq. (3.10). The significance for defenders is that reconstructing the system is equivalent to reconstructing the defect state and encrypting the defect pattern. Accordingly, the attacker’s difficulty increases from simply using the defect pattern to interpreting or identifying the defect pattern first and then exploiting the defect. As a result, the focus shifts from the defect pattern from the attacker’s perspective in Eq. (3.10) to the system state constraint that can be manipulated by the defender:

$$\Pr[S_t = s_t | S_{t-i} = s_{t-i}] = \Pr[S_t = s_t] \quad (3.12)$$

That is, system states S_t and S_{t-i} are independent of each other, which is also called independent reconfiguration of the system state.

Therefore, conditional expression (3.10) in Definition 3.16 can be equivalently replaced by Eq. (3.12) from the defender’s perspective.

3.6.3.3 Analysis and Inference

The above Definition 3.16 gives a perfectly safe and secure system that can “invalidate attack experience.” The key is to make the mapping from “historical attack results” to “future attack results” “perfectly safe” (or completely random), where:

1. Equation (3.9) is the intrinsic condition of the system, which means that the existence and dependability of the intrinsic function are the prerequisite. If we do not consider the intrinsic functions and only consider the perfect safety under the conditions of Eqs. (3.10), (3.11), and (3.12), then any randomly reconfigured system that can “blind” the attacker can meet the requirements, but the result is a cognitive fog for everyone, and the intrinsic function is lost. The intrinsic functions of the system we consider should be “transparent,” that is, the mapping from “historical intrinsic functions” to “future intrinsic functions” is “basically fixed” (i.e., with a high probability of being unchanged), and the mapping between any plaintext and any ciphertext is completely random, unlike the case of perfect secrecy. This is because the application scenario conditions and

original requirements of system security and information security cannot be exactly the same after all.

To make the system satisfy Eq. (3.9) at any time, system reconstruction (including restart, reset, cleansing, reconfiguration, or replacement) can be used to prevent attackers from increasing their knowledge of defect patterns, so that the pattern space before and after each reconfiguration remains unchanged (or the defect pattern space can be restored), thus keeping ε unchanged.

2. Equations (3.10), (3.11), and (3.12) are the security conditions of the system, consistent with the definition of perfect secrecy, which means that the defect pattern is perfectly encrypted by the state control key. Since the system structure determines the defect pattern, it can also be said that the system structure is perfectly encrypted, so that “the structures after each system reconfiguration are independent of each other” and can be regarded as “white systems with a 0/1 distribution of probability ε .” The so-called white comes from the classic “white noise signal with Gauss distribution,” and the noise values at different moments are independent of each other. Because bandwidth limitation is equivalent to the existence of inertia, white noise with an infinite bandwidth means that the noise signal has no inertia. Similarly, the white system, that is, the “completely non-inertial” system is perfect in the sense of safety because this will make all the attack experience worthless due to the non-inertia of the attack target. This means that the attack theory based on the premise of inertial systems has completely collapsed, and accordingly, all attack methods have become blind trial and error or indiscriminate attacks with a success probability of ε .
3. The parameter ε in Definition 3.16 means that the probability of the system generating extrinsic functions is no higher than ε , and ε can be made arbitrarily small through redundancy, heterogeneity, and other mechanisms (of course, this is only in a theoretical sense), as long as it is given in advance. In other words, under the condition of unknown vulnerabilities, attackers, and attack modes, the system can realize the intrinsic function through independent reconfiguration with a probability of $1 - \varepsilon$, and the attackers cannot obtain useful information from independent attack results to increase ε . Similarly, unintentional physical failures, software/hardware errors, or faults of the system in the past have no impact on the current operating state of the system. Under this definition, the dependability of the system can be characterized by the probability $1 - \varepsilon$ that the system outputs intrinsic function, where ε can be arbitrarily small, which means that the dependability can be arbitrarily close to 1, that is, the GFS problem of CPS can be solved with a “perfect ε ” in an integrated manner.
4. Inference 1: If there is a system that satisfies Definition 3.16, it proves “the existence of a perfect system with ε as the quantitative design indicator.” Moreover, ① ε can correspond to quantifiable design indicators and measurable test indicators in GFS systems. ② According to the definition of ESS [15], “a system or model is said to have ESS characteristics if there are interwoven properties of quantifiable design and verifiable evaluation of functional safety and network security determined by the endogenous structure or architecture of the system or

model.” If a PIFS system can be realized (such as proving that a DHR system is a PIFS system), it is equivalent to proving the existence of ESS.

5. Inference 2: PIFS has the entropy nonreducing effect of the system state and defect pattern. From Eq. (3.11), it can be deduced that:

$$H(S_t^q | S_{t-i}^q) = H(S_t^q) \quad (3.13)$$

That is, when S_t^q is known, the conditional entropy $H(S_t^q | S_{t-i}^q)$ of system defects is equal to the original entropy $H(S_t^q)$ [12], that is, the entropy of the defect pattern does not decrease. In the same way, according to Eq. (3.11), we get:

$$H(S_t | S_{t-i}) = H(S_t) \quad (3.14)$$

That is, the entropy of system state does not decrease.

3.6.3.4 The Correspondence Between PIFS and the Perfect Information Secrecy System

In summary, a PIFS system can be regarded as a system that “keeps the system’s structural information completely confidential” or a “perfect randomly reconfigured system whose intrinsic functions are transparent.”

1. **“System defect pattern” corresponds to “plaintext” and “ciphertext”:** For an unencrypted system, it is obvious that the attacker can infer the current defect pattern (plaintext) based on the defect pattern (ciphertext) obtained in the past. For a system whose structure is perfectly encrypted, however, the attacker only knows the probability distribution and defect pattern space of the current defect pattern (i.e., he knows the probability distribution and plaintext space) but does not know what the current specific defect pattern is (i.e., he does not know what the specific plaintext is).
2. **“System reconfiguration” corresponds to “encryption”:** System reconfiguration can be regarded as a kind of encryption of the system structure, that is, “dynamic structural encryption” of the software and hardware in the system, including GFS problems. “One-at-a-time reconfiguration” corresponds to “one-time pad,” which means that the system changes the state control key every time it is reconfigured, and the corresponding key encrypts or changes the key every time encryption is done.
3. **“System reconfiguration control method” corresponds to “the key”:** The system state at any time is determined by the state of the controllable structure after migration, and the key refers to the way of mapping from system structure S_{t-i}^q to system structure S_t^q .
4. **Encryption of system defect patterns is achieved through independent reconfiguration of the system.** Therefore, all structurally empowered systems

can be structurally encrypted, and the encryption effect can be guaranteed according to any given indicator ε .

5. **“Exhaustive attack” corresponds to “exhaustive cracking”**: The cracking method of “using output/input to reversely infer the system” corresponds to “using the ciphertext to reversely infer the plaintext” in cryptography. For a perfect system, however, “its input and output” have nothing to do with the “current system state,” so there is no better attack method than “exhaustive input,” and there is no better attack method than “exhaustive cracking” for the perfect secrecy scheme of cryptography.
6. **PIFS transforms the attacker’s one-way transparency advantage into a double-blind problem for both the attacker and the defender**: Without relying on any prior knowledge about the attacker, the defender can achieve “blind shielding” any attacks by structurally encrypting the GFS problems within the structure. If the attacker wants to exploit the defects to launch a robust attack under the condition of knowing the set of defects and the carriers in the system, he must be able to decipher the system structure at first.
7. **For intrinsic functions, it is logically transparent and fog-free, which is equivalent to no encryption**, while for extrinsic functions, it is perfectly scrambled cognitive (mimic) fog.
8. **A perfect intrinsic function system only needs encryption and does not need decryption**. The “plaintext” protected by PIFS is a system defect pattern. This information does not need to be transmitted to the outside world, so there is no need for decryption and key distribution, and there is no need to consider the security of key distribution.
9. **The “one-at-a-time reconfiguration” of the perfect intrinsic function system not only reconfigures the defect pattern of the system but also keeps the defect pattern space unchanged before and after reconfiguration**, that is, the defect pattern space at any two moments being equal (e.g., $S_{t-i}^q = S_t^q$) can be achieved by defect pattern reconfiguration. The former is to shield the defect patterns through “one-at-a-time encryption,” and the latter is to restore the defect pattern space of the system (i.e., to prevent attackers from increasing the defects) through system reconstruction (including restart, reset, cleansing, reconfiguration, or replacement).
10. **The perfect intrinsic function system is characterized by “one-at-a-time reconfiguration”** for any two inputs or any two outputs, so it can cope with some injection vulnerability attacks that only have inputs but have no outputs or some backdoor defects that only have outputs but have no inputs. From the perspective of PIFS definition, they can be regarded as a special case of Eq. (3.11) in Definition 3.16.

3.6.3.5 Key Elements to Realize PIFS

Based on the above definition, how to achieve PIFS is what we will discuss next. It is not difficult to see from the previous analysis that the realization of PIFS requires

three core technical elements, namely, dynamics/randomness, variety/heterogeneity, and redundancy [15]. Among them, randomness can be regarded as a special case of dynamics. Likewise, heterogeneity can be considered a special case of variety. Furthermore, we list several possibilities to realize PIFS.

DV intersection: A typical technology is moving target defense (MTD) [16], which is characterized by randomly selecting target objects to provide service functions. Even if it is indeed “one-at-a-time reconfiguration” and satisfies Eqs. (3.11), (3.12), and (3.13) of Definition 3.16, due to the lack of redundancy and the corresponding redundancy-based ruling mechanism, the intrinsic function is a 0/1 probability distribution where ε cannot be given arbitrarily, that is, it does not satisfy the conditional formula (3.10).

DR intersection: The typical technology is dynamic isomorphic redundancy [17]. However, because it is isomorphic, even “one-at-a-time reconfiguration” cannot change the defect pattern, resulting in conditional formulas (3.11), (3.12), and (3.13) not being satisfied.

VR intersection: The typical technology is dissimilar redundancy structure (DRS) [17], which refers to multiple redundant bodies working in parallel with a majority-based ruling mechanism. However, the reconfiguration method does not meet the randomness requirements of perfect safety, causing the system entropy to decrease as the number of attacks increases, that is, it is a “dissipative system,” so it does not satisfy conditional formulas (3.11), (3.12), and (3.13).

In summary, although D, V, and R are all key elements of perfect intrinsic safety, no combination of any two of them can meet all the requirements of Definition 3.16. Therefore, next we will discuss how to use the complete intersection of the three core elements to achieve perfect safety.

3.6.4 *The DHR System Is the Condition and Proof of the PIFS System*

3.6.4.1 General Properties of the DHR System

The DHR system architecture [1] transforms the complex and changeable GFS problems in the structure into a simple differential-mode or common-mode problem in the DVR domain and can be used as a way of realization of the PIFS system, as shown in Fig. 3.6 (see Sect. 3.3.5). Its main features are as follows: ① By introducing dynamics, heterogeneity, and redundancy based on feedback control into the cyber-physical system or digital facilities, and using heterogeneous executors with the same intrinsic function to obtain the ESS capability of reconstructing the system, it transforms the GFS problems including random or uncertain disturbances within the system (such as random errors, failures, faults, and other reliability problems) and the unknown man-made network attacks aimed at the “dark functions” such as software/hardware vulnerabilities and backdoors of the target system into a simple differential-mode or common-mode problem in DVR domain, which can be

expressed by probability tools. ② Based on the strategic ruling mechanism of “relatively true axiom” [18], it automatically perceives and shields the system errors caused by “known” and “unknown” attack threats by arbitrating the state vectors of heterogeneous executors with the same intrinsic functions and ensures the correct provision of intrinsic functions even if there are “poisonous bacteria” in the system. ③ Without relying on any prior knowledge about GFS problems and uncertain disturbances, it can resist any form of trial-and-error attacks or blind attacks, so that the attacker’s attack experience is not inheritable.

The DHR architecture can realize the expected intrinsic functions of the system with high probability under the influence of traditional random or uncertain disturbances (such as random errors, failures, faults, and other reliability problems) and unknown manmade network attacks aiming at the “dark functions” such as software/hardware vulnerabilities and backdoors of the target system. This is because the executors in the system are heterogeneous executors with the same or equivalent intrinsic functions, and under the “relatively correct” strategic ruling mechanism, intrinsic functions can be provided with a high probability even if individual executors are abnormal [15].

Therefore, the properties of DHR can be briefly summarized as follows (which can be considered to have been proven) [15]:

DHR Property 1: Existence of intrinsic functions (Seiendheit): Since all the executors have the same intrinsic function, when the extrinsic function of some executors is triggered, the system can produce the intrinsic function output with a given probability $(1 - \epsilon)$ because of the relatively correct strategic ruling and feedback cleansing mechanism.

DHR Property 2: Determinability of differential-mode disturbances: When the extrinsic functions of some executors are triggered but presented in a differential mode, they can be detected and converted into a certain incident because of the redundancy/heterogeneity of the executor set.

DHR Property 3: Escapability of common-mode disturbances: When the extrinsic functions of some executors are triggered simultaneously and pass the relatively correct strategic ruling mechanism, there will be common-mode escape and undesired output.

DHR Property 4: Reconfigurability of the structural state: There are mechanisms of feedback control and timed cleansing of executors for differential-mode and common-mode disturbances, making the system operating state or structure dynamically reconfigurable.

In principle, the DHR architecture’s strategic ruling mechanism does not exclude the majority of executors with the same output vector, and there may still be common-cause defects or common-mode escapes with a small probability. Only by making common-mode escape a “minimum-probability incident” as far as possible and making the attacker unable to obtain useful information from “the incident” can we prevent perceivable or nonperceivable common-mode escape from becoming the norm to the greatest extent [15]. Therefore, whether the DHR architecture can be realized and how to achieve perfect intrinsic functions is the focus of attention below.

3.6.4.2 Theorem of DHR's Perfect Intrinsic Functional Safety

It can be seen from the above that the DHR architecture meets the system's intrinsic conditions required by Eq. (3.9) in Definition 3.16, so let us consider how to meet the system's safety conditions in Definition 3.16.

To this end, we derive DHR's perfect intrinsic functional safety (PIFS) theorem by drawing on the idea of Shannon's perfect secrecy theorem [12].

Theorem 3.3. (Shannon's perfect secrecy theorem): The necessary and sufficient conditions for an encryption scheme (Gen, Enc, Dec) whose plaintext space is $\mathcal{M}$ and which satisfies $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$ to be a perfect secrecy scheme are as follows:

1. The key generator (Gen) selects any key from $\mathcal{K}$ with uniform distribution (equally possible), that is, $\Pr(K=k) = 1/|\mathcal{K}|$.
2. For each plaintext and each ciphertext, there is only one key to map the plaintext to the ciphertext.

To correspond to Shannon's perfect secrecy theorem, the DHR system is analyzed as follows:

1. For a DHR system, the encryption algorithm consists of an executor resource pool, the number of online executors, the ruling strategy, the feedback mechanism [15], etc. Like cryptographic encryption schemes, the encryption algorithm can be made public, but which executors are selected each time is determined by a private state control key.
2. For heterogeneous executors [17] with the same intrinsic functions in the DHR architecture, their heterogeneous nature determines the diversity and differentiation of defect patterns of different executors. Therefore, the system structure of DHR is strongly related to system defects, that is, there is one-to-one mapping between defect patterns and system states.
3. It can be seen from the above DHR properties 1–4 that the dynamics, heterogeneity, redundancy, and ruling mechanisms of the DHR architecture make ε in Definition 3.16 arbitrarily small. Among them, the ruling mechanism can optimize and reduce ε by automatically perceiving and shielding “known” and “unknown” defects under the premise of “the relatively true axiom.”
4. Theorem 3.1 requires that the plaintext space remains unchanged every time it is encrypted. For a DHR system, the system state space S_t before and after reconfiguration can be kept unchanged through methods such as restart, reset, cleansing, or replacement each time it is reconfigured (e.g., when a differential mode occurs). On the basis of (2), when the input and ruling mechanism are fixed, the invariability of S_t means that ε remains unchanged, thus satisfying Eq. (3.9) in Definition 3.16.

The logic of Shannon's perfect secrecy theorem is it only applies to $|\mathcal{K}| = |\mathcal{M}| = |\mathcal{C}|$ encryption schemes, that is, encryption schemes in which the ciphertext space, plaintext space, and key space are equal, and the encryption scheme must

satisfy two necessary and sufficient conditions. Corresponding to the DHR system, it satisfies $|S_t| = |S_{t-i}|$ by itself because the system state space has nothing to do with time [15]. It is only necessary to standardize the key space $|\mathcal{K}|$ for controlling the system state, so that for any t and i , each system state S_{t-i} at time $t-i$, and each system state S_t at time t , $|S_t| = |S_{t-i}| = |\mathcal{K}|$ there is only one state control key k to convert S_{t-i} into S_t , and there is also only one state control key k to convert S_t into S_{t-i} .

At this point, according to the logic and conditions of Theorem 3.1, the realization of PIFS based on general DHR only requires the following additional constraint:

DHR Property 5: Independentivity constraint: For any two inputs, the system state is reconfigured independently at each input, and the probability of the system being in any state is equal. Equivalently, for any two outputs, the system state is also reconfigured independently at each output, and the probability that the system is in any state is $1/|\mathcal{K}|$.

Based on the above analysis, according to the logical system of Shannon's law of perfect secrecy, the PIFS theorem of DHR is given.

Theorem 3.4. (The PIFS theorem of DHR): The necessary and sufficient conditions for a DHR system that satisfies Properties 1–4 to be a PIFS system are as follows:

Condition 1: The system changes its state every time it runs and every time it selects any state control key from K with uniform distribution (equally possible), that is, $\Pr(K=k) = 1/|\mathcal{K}|$.

Condition 2: For any t and i , for every system state s_{t-i} at time $t-i$ and every system state s_t at time t , there is only one state control key k to convert s_{t-i} into s_t , and there is also only one state control key k to convert s_t into s_{t-i} .

The proof of the theorem is as follows:

Referring to the proof of Shannon's perfect secrecy theorem, we give the following proof of Theorem 3.4. First, for simplicity and intuition, we let $S_t = M$ and $S_{t-i} = C$.

Sufficiency: First, according to Condition 1 of Theorem 3.4 and Condition 2 of 3.2, we have:

$$\Pr[C=c|M=m] = \Pr[K=k] = 1/|\mathcal{K}| \quad (3.15)$$

wherein (a) is established because of Condition 2 of Theorem 3.4 and (b) is established because of Condition 1 of Theorem 3.4. Then, for any two system states m_1 and m_2 at time t , and system state c at any time $t-i$, we have:

$$\Pr[C=c|M=m_1] = \Pr[C=c|M=m_2] = 1/|\mathcal{K}| \quad (3.16)$$

According to the following Lemma 3.3 of the literature [19] (i.e., the equivalent definition of perfect secrecy: perfect indistinguishability):

Lemma 3.3. If and only if any plaintext $m_1, m_2 \in \mathcal{M}$ and any ciphertext $c \in C$ of an encryption scheme both satisfy $\Pr[C = c | M = m_1] = \Pr[C = c | M = m_2]$, the encryption scheme is a perfect secrecy scheme.

From this we can get:

$$\Pr[M = m | C = c] = \Pr[M = m] \quad (3.17)$$

Necessity: First, we prove that Condition 2 of Theorem 3.4 is necessary.

For any one $c \in C$, system state c at time $t - I$ may come from any $m_i \in \mathcal{M}$ at time t . It may be assumed that if there are n state control keys that can convert m_i into system state c at time $t - i$, and these state control keys are called $k_1, k_2, \dots, k_n$, then $k_1, k_2, \dots, k_n \in \mathcal{K}_i$ and $\mathcal{K}_i \in \mathcal{K}$. For another system state $m_j \in \mathcal{M}$ at time t , we also assume that the state control key space that can convert it into state c is $\mathcal{K}_j \in \mathcal{K}$. Based on the above assumptions, we have the following inferences:

1. For any i and j , there is no intersection between $\mathcal{K}_i$ and $\mathcal{K}_j$, that is, $\mathcal{K}_i \cap \mathcal{K}_j = \emptyset$. Assuming that $\mathcal{K}_i$ and $\mathcal{K}_j$ intersect, then there exist $k \in \mathcal{K}_i$ and $k \in \mathcal{K}_j$, and k satisfies the condition that “ m_i is converted into c according to k ” and “ m_j is converted into c according to k ,” and simultaneous establishment is achieved. This is contradictory to Condition 2 of Theorem 3.4, so $\mathcal{K}_i \cap \mathcal{K}_j = \emptyset$.
2. For any i , there is only one state control key in $\mathcal{K}_i$, that is, $|\mathcal{K}_i| = 1$. Since (1) is established, we can deduce that for each system state $m_j \in \mathcal{M}$ at time t , there is a $\mathcal{K}_i \in \mathcal{K}$ to make state control key $k \in \mathcal{K}_i$ in $\mathcal{K}_i$ satisfy $\text{Enc}_k(m_i) = c$, and there is no intersection between any two $\mathcal{K}_i$. So, there is $\sum \mathcal{K}_i = \mathcal{K}$.

Assuming that there exists $m_i \in \mathcal{M}_i$ for each $m_i \in \mathcal{M}$, and there is no intersection between any two $\mathcal{M}_i$ (because any two states in the system state space at time t are different), then it can be deduced that $|\mathcal{M}_i| = 1$ and $\sum \mathcal{M}_i = \mathcal{M}$. And because of $|\mathcal{K}| = |\mathcal{M}|$, $|\mathcal{K}_i| = 1$.

From (1) and (2), it can be deduced that Condition 2 of Theorem 3.4 is necessary. Then, we prove that Condition 1 of Theorem 3.4 is necessary.

As can be known from Condition 2 of Theorem 3.4, for any system state $c \in C$ at any time $t - i$, if and only if the unique state control key k_i is selected, the system state $m_i \in \mathcal{M}$ at time t is converted into c , so there is $\Pr[C = c | M = m_i] = \Pr[K = k_i]$. Similarly, if and only if the unique state control key k_j is selected, the system state $m_j \in \mathcal{M}$ at another time t is converted into c , that is, there is $\Pr[C = c | M = m_j] = \Pr[K = k_j]$.

According to Lemma 3.3 [19], we get $\Pr[C = c | M = m_j] = \Pr[C = c | M = m_i]$, that is, the probability that system state $c \in C$ at any time $t - I$ comes from any two system states $m_j \in \mathcal{M}$ and $m_i \in \mathcal{M}$ at any time t is the same. Therefore, for every I and j , we have $\Pr[K = k_j] = \Pr[K = k_i]$. This means that every key is selected with the same probability, i.e., $\Pr[K = k] = 1 / |\mathcal{K}|$.

3.6.4.3 Analysis and Inference

From the theorem in the previous section, it can be known that by adding the above constraints (Condition 1 and Condition 2) to the general DHR system, it can be ensured that the constrained enhanced DHR (eDHR) system is a PIFS system with an ESS intensity of ε . The ESS intensity ε is related to the number of executors deployed, the degree of redundancy/heterogeneity between executors, and the ruling/cleansing strategy but has nothing to do with the nature and behavior of defects. This shows that the DHR architecture has cyber resilience.

The significance of “one-at-a-time reconfiguration” of the DHR system lies not only in randomly reselecting the executors but also in ensuring that the system state space remains unchanged through reconstruction (on the premise that there is a one-to-one mapping between defect patterns and system states, this means that system reconstruction can keep the defect pattern space unchanged at any two moments). The former enables the DHR system to directly encrypt the system state through reconfiguration, and the latter enables the DHR system to restore its defect pattern space through reconstruction (including restart, reset, cleansing, reconfiguration, or replacement of the current executor, operating environment, etc., to prevent attackers from injecting defects).

According to DHR Property 5, for any two inputs or any two outputs, the DHR system always has the characteristics of “one-at-a-time reconfiguration,” so it can deal with some injected-vulnerability attacks without outputs, or some backdoor-type defects that can trigger extrinsic functions and produce outputs without inputs, or information security threats that do not affect intrinsic functions but can affect the privacy, usability, and integrity of data.

3.6.5 Information Theory Basis of DHR’s Relatively Correct Strategic Ruling Mechanism

It is not difficult to find that DHR’s relatively correct strategic ruling makes the “ciphertext” information obtained by the attacker incomplete, which is equivalent to filtering the output of each executor again, that is, reconducting a many-to-one, high-dimensional to low-dimensional “entropy nonreduction mapping” [8] and changing the multidimensional output of multiple executors into one-dimensional output through the ruling mechanism.

This is equivalent to adding a multi-input, single-output MISO channel. The channel response is majority-based diversity with sign erasure, and its output has a reduced amount of information compared with the input. In the sense of information theory, additional equivocation is inevitably introduced, which means that even if the channel response is known, it is impossible to recover the complete information of the channel input from the channel output.

The strategic ruling mechanism based on the relatively true axiom is asymmetric to the defender and the attacker, positive to the defender but negative to the attacker, which will cause the attacker to observe entropy loss. This is equivalent to the defender actively completing one-way encryption of the structural information, which makes the attacker dubious about the system. Even if he can master the input sequence and know the output response, it is difficult for him to deduce the system structure. Therefore, it is another important means to reduce or even eliminate the attacker's "one-way transparency" advantage.

In particular, the size of entropy loss or dubiousness is related to the number of inputs (heterogeneity or redundancy [14]), that is, the number M of heterogeneous executors participating in voting. The greater the M in the " $M:I$ mapping", the greater the degree of freedom in the original high-dimensional space, the greater the loss ratio of degrees of freedom after falling to one dimension, and the greater the degree of dubiousness. When M approaches infinity (e.g., executor heterogeneity and redundancy approach infinity, and defect intersection approaches 0), the degree of dubiousness is infinite. When the degree of dubiousness is high enough, theoretically the attacker will be unable to identify the specific structure of the system, that is, the structure's defect pattern is perfectly shielded.

Therefore, DHR first completes the first encryption through online executor reconfiguration and then increases the attacker's dubiousness again through strategic ruling with an iterative mechanism, which is equivalent to encrypting the structure in the current operating environment for the second time or more before the system outputs the response, thereby further improving the safety of the system.

3.6.6 *Proof and Analysis of the Conjecture*

We have preliminarily proved the existence of the proposed conjecture, that is, there is a similarity between perfect encryption that does not rely on prior knowledge and ESS that does not rely on prior knowledge.

The logic and process of the proof are as follows: First, a cryptographic model of intrinsic functional safety is established, which transforms safety problems in the network domain into encryption problems in the cryptographic domain, allowing us to inherit and apply existing ideas and methods of perfect secrecy and then give the concept, definition, and inference of perfect intrinsic functional safety (PIFS); second, the DHR system is taken as a specific embodiment of PIFS, and the perfect safety theorem of DHR is proposed corresponding to Shannon's perfect secrecy theorem to prove that the constrained enhanced DHR structure (eDHR) satisfying "one-at-a-time reconfiguration" is the necessary and sufficient condition to realize PIFS, thus proving the existence of PIFS systems (hereinafter referred to as "perfect-safety" or "perfectly safe" systems in this section).

The derivative value of this proposition is that it not only provides a new perspective and method for interpreting and evaluating ESS mechanisms using

cryptographic principles, but more importantly, it can provide some inspiration for the further improvement and development of ESS.

To sum up, the establishment of the proposition of similarity between the two means that complex CESS problems can be explained in a normalized manner using relatively simple and popular knowledge of cryptography.

1. From a cryptographic perspective, ESS may essentially create a physical or logical encryption method for the system structure to achieve encryption of extrinsic functions other than intrinsic functions. If an attacker wants to exploit the structural defects within the system to carry out an attack, he must first decipher the structure or its password. In this sense, system or network security changes into password security.
2. All structurally empowered systems can be encrypted by the structure itself. We have demonstrated that the encryption strength is strongly related to the set of three elements of D, V, and R, and the coexistence of the three elements is the key to realizing a perfect system.
3. Since attackers are required to have a new ability to decipher structural passwords, the threshold of cybersecurity is raised to at least the same height as password cracking, and ubiquitous network threats and sabotage can be greatly reduced.
4. Just like “one-time pad” is a well-known cryptographic scheme with perfect secrecy, DHR with “one-at-a-time reconfiguration” is a classic embodiment of perfect safety and has the achievability of intrinsic functions and perfect shielding of defect information, thus proving the existence of a perfect system with ε as the quantitative indicator and also proving the existence of ESS from a cryptographic perspective.
5. The strategic ruling mechanism based on the relatively true axiom means that the DHR system’s dynamics, redundancy, heterogeneity, and multimode ruling work simultaneously, which enables the DHR system to reduce the probability of occurrence of extrinsic functions by automatically perceiving and shielding “known” and “unknown” defects, so it can be regarded as the guarantee of the existence of ε . From an engineering point of view, the greater the redundancy and heterogeneity of relatively correct ruling and the greater the abundance of a multimode ruling mechanism, the greater the freedom of ε value space, so it is an optimal means to achieve perfect safety.
6. The DHR ruling mechanism corresponds to Eq. (3.9) in the definition of perfect safety. The dynamics, heterogeneity, and redundancy of the DHR system and mechanisms such as cleansing, recovery, reset, and replacement correspond to the reconfiguration in the definition, that is, Eqs. (3.10), (3.11), and (3.12). At this point, the DHR system corresponds perfectly to the definition of PIFS, and the connotation of the ruling mechanism is also included in the definition of PIFS.
7. Perfect safety should be GFS including functional safety and cybersecurity. The existence of a perfect system with ε as the quantitative indicator is equivalent to proving that GFS can be “quantifiably designed and verifiably tested.”

8. It can be strictly proved that perfect safety has the property that “the entropy of a system defect pattern does not decrease,” which theoretically proves the existence of the “MDF” effect of ESS structures.
9. Under open-loop conditions, that is, under the assumption that there is no prior knowledge and no posterior knowledge (such as attack perception and effect feedback), perfect safety and perfect secrecy are cryptographically equivalent in theory. Only on this premise can the two be compared. Because the encryptor cannot perceive the attacker’s attack mode and its attack effect, perfect secrecy can only imagine a more severe open-loop scenario.
10. Even under open-loop conditions, perfect safety is “encryption-only” safety, that is, there is no legal decryptor, which avoids the biggest problem in perfect safety application—the security problem of key distribution. In this sense, therefore, perfect safety and perfect secrecy are symmetrical. The former has relative innate advantages and safety gains and should be more operable and achievable.
11. Under closed-loop conditions, that is, on the premise that there is no prior knowledge but the posterior knowledge of the attack can be obtained and the security strategy is adjusted through feedback control, perfect safety and perfect secrecy are theoretically not equivalent. For example, DHR’s “differential-mode disturbance decidability” allows the extrinsic functions of the executor to be triggered and detected when they are presented in differential modes, which means that the defender can use DHR to obtain posterior knowledge about system defects. This advantage may break through some limitations of cryptography, allowing the defender to use the feedback information provided by DHR to change the original “blind encryption” strategy, thereby disrupting the attack chain with higher efficiency, lower engineering complexity, and more pertinence.

For example, “one-at-a-time reconfiguration,” like “one-time pad,” stems from the open-loop assumption only. In closed-loop situations, it may be a sufficient condition, but not a necessary condition (it may be too strong and should be relaxed). At least, what is “one time” and how to optimize its interval can be studied. For example, with the help of the feedback information provided by DHR, the defense can adaptively and pertinently optimize the reconfiguration interval of the system. As long as the reconfiguration interval is shorter than the time needed for establishing the attack chain, there is no need to blindly pursue shortening the reconfiguration interval. In this way, the defender can disrupt the establishment and stability of the attack chain at a lower cost.

12. Therefore, the attack-defense problem of the perfect DHR system under closed-loop conditions can be transformed into a game between “establishing the attack chain” and “disrupting the attack chain.” This is similar to the computational power game between the decipherer and the algorithm based on computational complexity in traditional cryptography and cannot escape the contradiction between ideal vision and realistic feasibility. It is worthwhile to sacrifice some quantifiable and controllable safety losses for engineering feasibility and substantial efficiency improvement. Because the DHR strategic ruling feedback

control mechanism provides the system with additional degrees of freedom, it is possible to optimize safety under closed-loop conditions in the direction of “maximizing efficiency while satisfying safety constraints.” This can be one of the key points of follow-up research.

13. Perfect safety under open-loop conditions makes network offense-defense games no longer rely on whether the attacker and defender have prior knowledge about the opposite party. Both parties return to the “Stone Age” scenario without prior knowledge. Furthermore, the network attack and defense under the closed-loop conditions are more likely to return to the normal state of “easy to defend but difficult to attack.” The defender can not only encrypt himself but also rely on “differential-mode presentation” to perceive the attack effect and change the current defect pattern in real time, thus changing the overall situation of the defender from simple passive defense to autonomous and controllable active defense.
14. The process of proving the existence of perfect safety of DHR also answers the difficult question of “how much randomness the DHR reconfiguration must have to realize perfect safety without any prior knowledge,” that is, how “perfectly random DHR” should be designed. The answer can be used as design criteria for the randomness and dynamics of DHR, so as to reexamine possible problems in DHR design from the perspective of perfect secrecy and carry out follow-up research on related problems.
15. It is possible to create a type of structurally empowered resilient system and even the idea, criterion, and realization method of endogenously safe and secure cyber resilience. This depends on more in-depth and detailed follow-up research.
16. As the safety goal is raised to the level of “perfection,” generally speaking, the requirements imposed on the defender will be more stringent (such as one-at-a-time reconfiguration), and the cost will increase accordingly (for DHR, the cost may not be as high as expected). If a perfect effect is what you pursue, this chapter proves that the cost is worth it and necessary. If you are pursuing a compromise between theoretical perfection and engineering implementation, you can define “subperfection,” asymptotic perfection, average perfection, and short-term perfection based on this chapter.
17. Shannon invented the concept of the information entropy in his “A Mathematical Theory of Communication” [8] and pointed out that the communication system should be optimized in the direction that “the entropy of the information source is not reduced,” where the channel coding theorem is the origin of the concept of ESS structural coding of DHR. In the following year, in his another work “Communication Theory of Secrecy Systems” [12], Shannon pointed out how to use the information theory to guide the design of secrecy systems in the direction of “undiminished plaintext entropy,” in which the perfect secrecy theorem is the source of perfect safety mentioned in this chapter. Looking back on both of them, it is not difficult to find that they have the same first principle, that is, “the entropy of the information source does not decrease” and “the entropy of the plaintext does not decrease.” This is not only the law that

Shannon discovered when creating the two fields of reliable communication and secret communication but also the script preset for the integration of ESS based on the theorem of endogenously safe and secure structural coding and perfect safety based on perfect secrecy. The related problems in this field are worth more in-depth studies in the future.

3.7 Engineering and Technical Effects of the DHR Architecture

3.7.1 Dimensionality Reduction and Transformation of GFS Problems

Various CESS problems are emerging endlessly, and attack methods are changing with each passing day. The knowledge base of related vulnerabilities, backdoors, virus Trojans, attack features, black and white lists, etc. is rapidly expanding. Network security, functional safety, and even information/data security problems are intertwined, bringing unprecedented challenges to GFS. The key reason why the DHR architecture can achieve GFS without relying on prior knowledge and additional defense measures, and without caring about the causes/mechanisms of common ESS problems and specific details such as faults, failures, and network attacks, is that it can transform these various and seemingly disorderly security problems into differential-mode or common-mode generalized reliability incidents that can be expressed by probability tools in the DVR domain through the DHR architecture. This provides a basic support for solving or avoiding these common engineering problems using mature fault-tolerant, error-correcting, and automatic control theories and technologies.

Theoretically, the network security defense effect of the DHR architecture is weakly unrelated or even unrelated to whether there is generalized uncertain damage or knowing the origin of generalized uncertain disturbance. However, this does not affect the appropriate introduction of background analysis and processing functions such as artificial intelligence and big data to achieve the transition from “knowing what it is” to “knowing not only what it is but also why it is so.” With the use of such information as running logs, field snapshots, and abnormal state retention and the help of increasingly mature intelligent analysis tools, we can not only find or locate software/hardware faults but also trace unknown vulnerabilities, backdoors, virus Trojans, and related attack resources and means, including capturing the extremely harmful “0-day” unknown security problems or network attacks.

3.7.2 *Local DHR*

As we know, the DHR architecture belongs to the complete intersection of DVR (whether it is unique or not remains to be proved) and usually consists of M functionally equivalent heterogeneous executors or related software and hardware, which can form a dynamically callable resource pool with at most C_M^K combinations (K usually takes 3). The current operating environment of the DHR architecture can be formed by arbitrarily selecting a feedback control loop based on policy decisions. The current operating environment of the DHR architecture can be formed by selecting any combination of executors based on the feedback control loop of strategic ruling. Theoretically, if any heterogeneous executor or operating environment is “absolutely heterogeneous,” then whether it is random failure, uncertain error, human disturbance, or nonhuman perturbation, it can only generate an expression of differential-mode output in the current operating environment. In actual product design, however, there is only a limited degree of heterogeneity between functionally equivalent executors or operating scenarios. Therefore, various abnormal perturbations or disturbances within the architecture will not only produce differential-mode expressions but also perceivable or nonperceivable common-mode expressions (the probabilities of their presentations are definitely different). In engineering, we always hope that the probability of the former is higher and that of the latter is as small as possible. The larger the ratio between the two is, the higher the GFS or dependability will be. Therefore, how to obtain sufficiently high GFS under the condition of limited global static heterogeneity is a challenge in engineering implementation. Fortunately, the iterative feedback mechanism of the DHR architecture based on strategic ruling can largely reduce the very demanding global static heterogeneous redundancy requirements such as DRS or heterogeneous redundancy.

In fact, it is enough to ensure that the current operating environment has local DHR characteristics. The prerequisites are as follows: ① The attacker cannot control the common ESS problems in the current operating environment in real time; ② the attacker cannot sniff or perceive the resource configuration of the current operating environment by scanning or the trial-and-error method; ③ the attacker cannot control the scheduling strategy and iterative replacement law of the executors or scenarios in the current operating environment; ④ even if entering the perceivable or nonperceivable common-mode escape state, the multidimensional backward verification mechanism, the forced disturbances based on external commands, and attempted cleansing or replacement of relevant software and hardware resources in the operating environment may disrupt the stable escape state. In other words, the current operating environment of the DHR architecture is usually in a stable working state, but perturbations or disturbances by internal and external factors may be excited to a temporarily stable state of perceivable differential-mode processing. Once the differential-mode output shielding mechanism finds a differential-mode expression in the current running environment, it will change the working state of the current operating environment or the executor scene or even reset the resource

deployment in real time under the condition of ensuring that the service function P remains unchanged and the target object and attacker are nonperceivable, until the ruling process no longer perceives differential-mode expressions and the current operating environment returns to a stable working state.

It is worth noting that this process does not mean that the factors causing differential-mode perturbation or disturbance have been eliminated nor does it mean that there are no common-ESS problems in stable working conditions. It only means that the current attack has lost its pertinence and the effect of “fighting the enemy with force, blocking the water with earth” has been achieved. It should be emphasized that this local DHR mechanism requires the introduction of other dimensional relativity parameters for deeply correlated strategic ruling. The purpose is to obtain the identification between differential-mode expression and perceivable common-mode escape on the whole through the iterative replacement of operating scenarios [11] and reduce the probability of “perceivable common-mode escape expressed in a differential mode” in the blind zone of relativity ruling. For instance, relevant parameters such as the historical performance weight of each executor or scenario in the operating environment, the executor’s software/hardware safety design level, the maturity of the running version, and whether there are various warning messages are included in the scope of strategic ruling. In addition, if the strategic ruling determines that there is a perceivable common-mode escape, it is necessary to try to change the executor or scenario encoding in the current operating environment, including those objects that do not have differential-mode output. This is because when common-mode escape occurs, the problem scenario is often not those executors that have differential-mode performance.

In addition, with the development of the industrial chain featuring “open-source technologies and specialized division of labor” industrial models, it is no longer alarming to worry about homologous vulnerabilities, backdoors, trapdoors, and the spread of malicious codes. A large number of facts show that the ecology of cyber-physical systems or digital products based on open-source codes is facing more and more serious cybersecurity threats. What makes the industry extremely frustrated is that we can neither ignore the new trend of open-source community technology development nor can we thoroughly investigate the vulnerabilities, backdoors, and even virus Trojans in the open-source codes. Worse still, there is no solution with the expected safety effect so far. Fortunately, the uncertain scheduling strategy, cleansing and reconfiguration strategy, and iterative ruling strategy of heterogeneous executors or scenarios in the DHR architecture will significantly increase the difficulty of collaborative and consistent exploitation of environment-dependent homologous vulnerabilities and backdoors. From this it is not difficult to infer [16] that the DHR architecture and its generalized robust control mechanism have much looser requirements for dissimilar design in engineering implementation than the DRS architecture with dissimilar redundancy.

To sum up, DHR can obtain stronger cyber resilience at a lower cost of heterogeneity.

3.7.3 *Strategic Ruling Mechanism*

The DHR architecture does not rely on yet does not reject prior knowledge in its working mechanism. Therefore, the ruling mechanism of multi-mode output vectors or states can only be based on relativity algorithms. However, simple output state majority-based ruling inevitably has a blind zone in local DHR scenarios. For instance, if there is a 2:1 or 1:2 differential-mode state when the number of redundancies in the operating scenario is 3 and the “absolute heterogeneity” cannot be guaranteed in engineering, it cannot be simply considered that the current differential-mode problem must be present (although it can be considered in a high probability), but it is still impossible to “selectively ignore” the problem of perceivable common-mode escape in a small probability. For this reason, it must be identified from multiple perspectives through iterative ruling based on multiple parameters. For instance, according to the result of majority-based ruling, the historical performance weight of each executor in the current operation scenario, the designed safety level and maturity of the related software/hardware versions, and whether there is alarm or prompt information of the operating system, security software, middleware, and related physical layer hardware, multidimensional correlation strategic ruling can be conducted to reduce the probability of perceivable common-mode escape in the differential mode as much as possible. In practice, whenever a differential-mode state occurs in the ruling process and there is a certain alarm in the architectural environment, it can be directly concluded that the result of ruling is “a true differential-mode” rather than “perceivable common-mode escape,” and there is no need to restart the backward review and verification mechanism. If the system output response has delay constraints, the system output can be selected based on the result of majority-based ruling, and then the backward review and verification mechanism can be used for follow-up or retrospective processing.

It needs to be particularly emphasized that the strategic ruling mechanism makes it difficult to achieve blind collaborative attacks using trial-and-error methods. This is because the strategic ruling process forces any noncollaborative attack that attempts to “tamper with” the current operating environment to mechanically fail and it is difficult to obtain the expected results of common-mode escape in dynamic multitarget collaborative attacks without using trial-and-error methods. In addition, no matter whether it is an agreed collaborative attack or a blind attack of “hitting the big luck,” any “collaboration mistake” in the process will cause all previous efforts to be wasted.

In addition, the selection of the strategic ruling algorithm is strongly related to whether the target system adopts the centralized processing or distributed processing architecture and is inseparable from the specific details in the application scenario. For instance, in a geographically or spatially distributed application scenario, when the operating environment includes all participants in the blockchain, or when software and hardware resources such as cross-platform heterogeneous clouds and data centers are used, the strategic ruling algorithm needs to choose a distributed consensus algorithm such as “Byzantine generals problem.” In short, it is crucial for

the strategic ruling mechanism to “adapt measures to specific circumstances and apply what is learnt flexibly”.

3.7.4 *Entropy Nonreduction and Uncertainty*

There are functionally equivalent heterogeneous redundant executors (or operating scenarios) in the I[P]O model of DHR. This means that the structural expression of service function P can be realized with a single executor or scenario, or it can be presented in the form of “structural coding or mimic disguise” of multiple executors or scenarios. In other words, the DHR architecture allows executors or operating scenarios to be implemented through definable software and hardware methods such as self-reorganization, reconfiguration, and virtualization while ensuring service function P ; it can also be expressed by pooled executors. For example, when there are M elements ($M \geq 3$) in the resource pool with equivalent function P , there are C_M^3 combinations to choose from when creating a triple modular redundancy. Elements A, B, and C are functionally equivalent under given constraints, that is, $P = P_A = P_B = P_C$. Therefore, there are four combinations in the current operating environment: $P = P_A \cup P_B$, $P = P_A \cup P_C$, $P = P_B \cup P_C$, and $P = P_A \cup P_B \cup P_C$, not to mention that there are still $C_M^3 - 1$ combinations in the resource pool that are ready to be called at any time. That is to say, as long as the normal expression of the service function P (including related performance) can be guaranteed, the actual mapping (structural representation) between the specific implementation structure (or algorithm) of the operating environment and the service function P is no longer definite or rigid in a theoretical sense. This uncertain, one-to-many structural coding representation coupled with the differential-mode state-driven strategy feedback mechanism can not only achieve the effect of “entropy nonreduction” in the system empowered by the DHR architecture but also exponentially increase the difficulty of attackers’ awareness of the target system’s environment or vulnerability sniffing, vulnerability/backdoor discovery or exploitation, uploading virus Trojans, and covert internal-external communication. It not only causes difficulties in the attacker’s knowledge or perception of the operating environment or defense scenarios within the DHR architecture but also brings great inconvenience to the accurate delivery of attack packages (data or executable code) targeting known vulnerabilities, backdoors, etc. For any differential-mode expression of ruling that leads to inconsistent output state or abnormal operating environments, the DHR strategy scheduling and multidimensional dynamic reconfiguration feedback mechanism will reimplement “structural encryption” or change “structural coding” for the current operating environment until the abnormal situation disappears or the frequency of occurrence is limited to a given threshold. Theoretically, neither “differential-mode perturbations,” “differential-mode attacks,” nor blind collaborative attacks are mechanically effective against the DHR architecture, which is also true of common-mode escape that “cannot be stably maintained even if the attack is

successful.” This is both theoretically and technically subversive to all plannable attack operations and predictable attack results.

3.7.5 *Mimic Disguise Fog (MDF)*

Under the condition of not changing the equivalent service function P of the executor or the operating environment, the empowerment of the DHR architecture and its operating mechanism can make the potential common ESS problems in the target environment present or show great uncertainty. Vulnerabilities, backdoors, virus Trojans, and other dark functions that are parasitic or associated with the executors in the current operating environment are structural products. On the one hand, their location or exploitability may change with the reconfiguration, reorganization, cleansing, recovery, and other operations of the host executors or operating scenarios (theoretically, semantic-level vulnerabilities and backdoors may not be affected). On the other hand, the current operating environment will undergo strategic iterative scheduling due to some differential-mode disturbances or external-command-forced perturbations, which will lead to some changes in software/hardware resource configuration or the state of the operating environment. The superposition of the two factors causes the DHR architecture to exhibit the structural encryption attribute of “one differential-mode disturbance, one change in the operating environment” from a macroscopic perspective. For internal infiltrators or external attackers, this attribute presents a “double-blind” mimic disguise fog and uncertain effect, which makes it extremely difficult to achieve the desired attack effect based on prior knowledge, information obtained through on-site sniffing, preplanned attack strategy, and technical means. To achieve the desired effect, the attacker must first crack the structural code or environmental password, which is no easier than cracking the information encryption algorithm. Obviously, once the dark function intersection (common-cause/common-mode problems) hidden in the current operating environment is structurally coded or environmentally encrypted, the practicability and effectiveness of the current mainstream network attack theories and methods will be greatly reduced, and even the rules of network attack-defense games will be fundamentally changed.

What is challenging is that if we want to use the intersection of uncertain dark functions in the DHR architecture to achieve collaborative and stable common-mode escape of multiple heterogeneous dynamic targets under noncollaborative conditions between executors or scenarios in the current operating environment, even under the condition of white box injection tests [11], it is by no means easy to achieve effective differential-mode disturbance through the attack surface. In addition, the strategic ruling and feedback scheduling of the DHR architecture, as well as the multidimensional dynamic reconfiguration mechanism, make the available attack resources in the current operating environment highly uncertain, so it is difficult for the same attack sequence or combination of operations to get exactly the same output response in the temporal dimension. For instance, the target system’s vulnerabilities

and their characteristics that the attacker can see under the static heterogeneous redundant architecture will be uncertain or even unable to identify and locate in the DHR environment. This brings great challenges to the establishment of a stable attack chain; even if it can be barely established, the dependability and stability of attack-chain operations such as vulnerability detection, code injection, back-connection, and interaction cannot be guaranteed. To put it bluntly, even if an attack is successful, its experience will be difficult to reproduce in the spatial-temporal dimension. Facing a target environment where “structural encryption is made every time a differential-mode problem is perceived,” an attacker without the ability to decipher and deconstruct can only feel helpless, even if there are “0-day” attack resources. Therefore, implementing environmental encryption and structural coding is the core of creating mimic disguise fog and uncertainty.

It should be noted that the mimic disguise fog and uncertainty of the DHR architecture are almost the same for defenders and attackers. Not only is it difficult for attackers to know where and what kind of dark function intersections exist in the current operating environment, whether they are exploitable or not, but also product designers or users do not know in real time what changes will happen to the working state of related software/hardware, resource configuration, and algorithm implementation in the current operating environment of the application system. This uncertain “privacy” has a “double-blind effect” on both offensive and defensive sides. This is just like the case of commercial cipher machines that anyone can buy, but the use of the same model of machine will never compromise the information security between users. Therefore, the “double-blind effect” also has a natural mechanical advantage in resisting social engineering-based attacks. However, the defender is not completely “blind”; he just does not know the cause of the differential-mode state in real time. On the contrary, it is not difficult to find “0-day” vulnerabilities, backdoors, virus Trojans, and software/hardware reliability problems using all kinds of on-site reserved information and knowledge of the defender’s own software/hardware, with the help of knowledge base technologies and various analysis tools. In short, the mimic disguise fog and uncertainty of the DHR architecture make the attacker “really blind” but only causes the defender pseudo-blindness that can be recovered through post hoc analysis.

3.7.6 Theoretical Prerequisites of Subverting Trial-and-Error Attacks

Continuous sniffing or penetration technologies such as black box scanning, blind or trial-and-error attacks, and APTs are one of the necessary means of network attacks. In theory, the effectiveness of trial-and-error methods requires two necessary premises. First, the background conditions of the trial-and-error object must remain unchanged during the whole trial-and-error process, or the system transfer function is always determined; otherwise, the prerequisite for the convergence of the

trial-and-error algorithm cannot be established. Second, the output state can always correctly respond to the input state, that is, the input sequence can be arbitrarily arranged by the trial-and-error attacker, and the corresponding information can always be obtained at the output end regardless of whether the trial-and-error attack is successful or not. The two prerequisites are indispensable. The reason why the DHR architecture can effectively inhibit the use of trial-and-error methods is mainly reflected in three aspects. One is the compliance check of the input sequence. That is, all excitation signal sequences applied at the input end must pass the compliance check of the consistency of each executor in the current operating environment. Whenever an executor generates a corresponding noncompliance alarm (due to the lack of uniform compliance standards for most software and hardware components), the current operating environment theoretically needs to make changes to the structure and operating state without changing function P (depending on the design strategy of differential-mode processing), which leads to a change of the structural code of the operating environment under the condition of functional equivalence. In addition, the trial-and-error attack sequence can only be “traversed” or orchestrated to a limited extent under specified conditions, which greatly reduces the scope of trial and error. Second, even if a certain trial-and-error input causes an individual executor to produce a differential-mode output response, the strategic ruling process will consider this to be a “differential-mode incident” and prohibit or block its output, which will lead to the attacker’s failure to successfully obtain a trial-and-error response. Third, once the strategic ruling mechanism discovers a “differential-mode incident,” it will activate the feedback control loop, which may instruct the executor to perform some cleansing and recovery operations or even change the current operating environment. It may even directly issue an order to replace the current “problem executor.” As can be seen, the prerequisites that “the background of the trial-and-error process remains unchanged” and that “the input state can always be mapped to the output end” are difficult to establish in the DHR architecture. It needs to be emphasized that even if the attacker has certain prior knowledge, how to “bypass” the DHR architecture’s mechanism of denial of trial-and-error attacks and achieve a coordinated “non-perceivable common-mode escape” is an extremely difficult challenge.

3.7.7 New Challenges of Code Injection

Exploitable software/hardware vulnerabilities are inseparable from “suitable” environmental conditions and “tailor-made” attack processes and specialized tools. The exploitation of vulnerabilities, backdoors, and other attack resources also requires the introduction of input stimulus sequences (referred to as attack sequences) that comply with grammar, semantic rules, and even pragmatic environments through the attack surface and accessible channels. The reason for emphasizing this premise is that the input/output channel of the DHR architecture is not only the channel through which the current operating environment receives service requests and service

responses but also the attack surface that enters the relevant executor or operating scenario. More importantly, ESS problems ultimately belong to the endogenous contradictions of the target object. Whether it is random/uncertain perturbation or network attack disturbance, it is the result of interaction between external factors and internal factors. Therefore, the attack surface and channels, as well as methods and environment, all have unavoidable “pass/strategic position” problems for both attackers and defenders.

Any attack sequence created at the **[1]**end of the DHR structure should not only satisfy the service request compliance check (otherwise it will be discarded or prohibited by the input rules) but also ensure that the injected attack code will not be regarded as illegal code and cause false alarm because of the unique “environmental encryption” effect of the “structural coding” of parallel excitation. For instance, it is assumed that there are many types of CPUs in the current running environment, and the binary attack code that can be accepted by the x86 system will have a high probability of generating illegal instruction alarms for ARM systems, and vice versa. Since the vulnerabilities of each executor or operating scenario in the DHR architecture are different, and the exploitable attack resources, the attack chain created, and internal-external communication mechanisms are also different, it is by no means an easy task for any attacker to inject the pre-orchestrated attack sequence into the attack surface and shared channel with compliance checks in the spatial-temporal dimension and realize accurate “one-to-many” delivery without exposing the attacker’s attempt prematurely because of the mistakes in accessibility, feasibility, or collaboration. A more general expression is that under noncollaborative conditions and in a DHR environment, it is extremely difficult to achieve collaborative injection across heterogeneous executors or operating scenarios through a single input channel and the same attack surface.

3.7.8 *Natural Integration of Add-on Security Technologies*

Although the effectiveness of the DHR security mechanism does not rely on (but does not reject) any prior knowledge or add-on, built-in, embedded, and symbiotic security measures or technical means, exponential security gains can be obtained by reasonably orchestrating or configuring COTS-level traditional network security products with different security mechanisms within the architecture. This is because the inherent fusion ability of the DHR architecture enables other security technologies to be naturally “integrated with” rather than “attached to” the system. Just like the relationship between the “steel skeleton,” “concrete material” and the strength and toughness of the building, the steel bar configuration, material properties, concrete material formula, and mixing technology determine the desired strength of the building. The DHR architecture plays the role of “steel skeleton configuration” here. If traditional security technologies similar to steel bar materials, concrete materials, and mixing technology are selected as required, it is not difficult to obtain, through quantitative combination of relevant parameters, the stability robustness and

quality robustness that can be quantifiably designed and verifiably measured to meet the needs of digital products for GFS, based on which the functional safety and network security of the products should be able to withstand the most stringent “white box injection” golden test in the resilience verification theory [3, 11].

It should be pointed out that traditional security technology plays a role in enhancing the heterogeneity of the current operating environment. It can be seen from theoretical analysis that the ESS of the DHR architecture is exponentially related to the number of redundant resources and the heterogeneity between executors, and it is not based on the breadth and depth of knowledge base accumulation such as prior knowledge or behavior characteristics of the attacker nor on the effectiveness, dependability, and trustworthiness of any add-on security means. However, in terms of mechanism, the unique fusion attribute of ESS is not exclusive to other security technologies, and by naturally accepting traditional or COTS-level add-on security technologies, it can not only significantly improve the technical economy of heterogeneity in the DHR architecture in engineering implementation but also greatly increase the difficulty of creating an attack chain in the environment of heterogeneous executors. In other words, strategically, differentially, and dynamically configuring some add-on or embedded security protection measures that are insensitive to application service P in the executor or scenario of the operating environment can equivalently improve the heterogeneity of the current operating environment, so exponential security gains can be obtained through the DHR architecture [11]. Generally speaking, under the condition of function P equivalence, any means that can enhance the dissimilarity or diversity among executors can nonlinearly improve the defense effectiveness and reliability of the target object. For instance, under certain constraints, scientifically deploying or configuring various dynamic, diversified, and randomized methods and means in the current operating environment, including accepting traditional security measures such as intrusion detection, intrusion prevention, intrusion isolation, encryption, authentication, antivirus and Trojan killing, backdoor sealing, and trusted computing, which are transparent to service function P and have different operating mechanisms, can effectively improve the cyber resilience of DHR architecture-empowered target objects or systems. More importantly, there is no need to worry about the safety and efficacy of these traditional “plug-in” measures (such as missed detection, false alarm rate, dependability, etc.), and there is no need to make sure that there are no ESS problems, a promise that is difficult to verify and impossible to honor. In other words, there is no need to “prove self-innocence.”

3.7.9 Safety Quality That Can Be Designed and Measured

The DHR architecture’s feedback control mechanism not only makes the dynamic, random, and diverse defense elements present an integrated effect but also makes the empowered system always behave as a “TEE” that can effectively cope with or avoid the impact of uncertain network attacks or random failures, making the “cyber

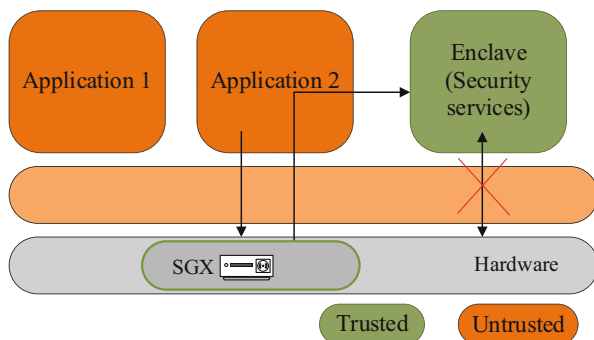
resilience” of functional safety, network security, and even related information security “quantitatively designed and verifiably measured,” a goal that can be realized in engineering. This is determined by the inherent properties of the DHR architecture and has nothing to do with the premise of human or nonhuman disturbance. Therefore, any digital product empowered by the DHR architecture can “directly inject” the corresponding differential-mode or common-mode test cases into the “open-state” executor or operating scenario under the condition of “white box plug” or even “destructive test” with the help of the resilience function verification theory and method, design the matching input test sequence, transform the input sequence and test cases on the attack surface, observe the state of the strategy arbitrator and the service response at the output end, and calculate the frequency of abnormal rulings in a given test period to indirectly measure the safety level of the target object or analyze whether there is escape information expected by the “injected differential-mode test case” at the output end. Different from most existing security defense technologies, the GFS of the DHR architecture can be designed, calibrated, tested, and measured; similar to the traditional reliability indicator, it can be measured by a probability value [17]. It needs to be emphasized that although DHR can mechanically perceive and shield 100% threats and sabotage caused by noncollaborative attacks or differential-mode faults, the recovery or avoidance of these abnormal operating environments will be at the expense of increasing the cost of resilience recovery and reducing the cost performance of the executors or operating scenarios, and service performance may be degraded, or even service provision may be interrupted for a short period of time under extreme conditions. Just as the evolution, transformation, and reconciliation of contradictions must pay an extra price or cost, the DHR architecture will inevitably pay technical and economic costs while gaining cyber resilience functions such as redundancy, diversity, and dynamics.

With the same technical architecture, DHR can also provide cyber-physical systems or digital products with high dependability, high trustworthiness, and high usability in an integrated way. Therefore, both traditional and nontraditional security functions can achieve network resilience based on structural gains through integrated system design based on this architecture, especially providing a brand-new solution for solving the multiple interweaving problems of functional safety, network security, and information/data security. In addition, the security of the target system empowered by the DHR architecture is weakly related to the technical ability and past experience of the operation and maintenance personnel, which can significantly reduce the system’s maintenance threshold and full-lifecycle operating costs.

3.7.10 Endogenously Safe and Secure TEE

The trusted execution environment (TEE) is a secure area (secure enclave) built in the CPU through software/hardware methods, which ensures the confidentiality and integrity of the programs and data loaded in it. TEE is an isolated execution

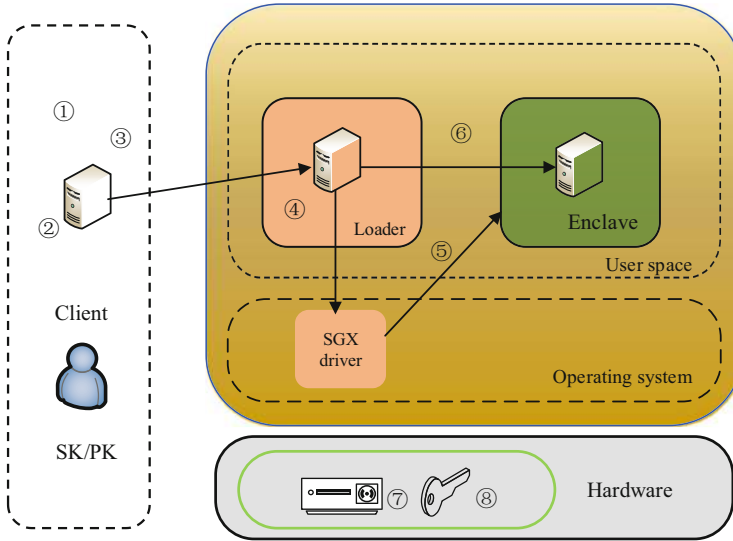
Fig. 3.19 Schematic diagram of the SGX architecture



environment that provides a higher level of security than a rich operation system (ROS) and more functions [19] than a secure element (SE) for running trusted applications on the device. A typical embodiment of TEE is the concept and principle of the SGX technology put forward by Intel in the Workshop HASP at the ISCA conference in 2013 [19]. In October 2015, the first-generation CPU supporting SGX technology came out; its architecture is shown in Fig. 3.19. SGX provides a trusted space on the computing platform to ensure the confidentiality and integrity of users' key codes and data, reduce the trusted computing base (TCB) that security applications rely on to the extent that it only contains the CPU and security applications themselves, and support new software architectures that exclude untrusted complex Oss and virtual machine monitors (VMMs) from the security boundary. The creation process of its "secure enclave" is shown in Fig. 3.20. The SGX architecture does not identify or isolate all malware on the platform but uses the idea of trusted computing to encapsulate the safe operations of legitimate software in an independent storage space to protect it from malware attacks. No other software can access this independent storage space regardless of whether it has permission. In other words, once the software and data are located in the independent storage space, even the operating system or virtual machine monitor (VMM) cannot affect the code and data in this space.

However, SGX still has security risks. The first is how to ensure the trustworthiness of the CPU as one of the security boundaries, especially how to prove that the hardware functions that control the independent storage space are not vulnerable, so there is still no answer to the question "whether the trusted root itself is trustworthy." Second, the TEE mechanism similar to SGX has inherent hardware design defects, such as the negative impact of specter vulnerabilities on SGX. Third, if the trusted program encapsulated in an independent storage space itself has design defects or malicious codes, then under the protection of the "enclave fortress," it can still avoid any security monitoring and problem removal operations. Fourth, once the malware successfully enters the independent storage space, the entire SGX function may be exploited by malware developers.

Public information shows that SGX itself has a variety of security problems, such as hardware design defects, transient execution vulnerabilities, side-channel security vulnerabilities, memory security vulnerabilities, and state consistency



① Creating application; ② Creating an application certificate (HASH and client PK); ③ Loading the application to the loader; ④ Creating enclave; ⑤ Assigning the enclave page; ; ⑥ Loading the test application; ⑦ Verifying the integrity of the certificate and enclave; ⑧ Generating the enclave key

Fig. 3.20 Enclave creation process

vulnerabilities. A team led by Professor Zhang Yiqian of South University of Science and Technology of China implemented SgxPectre attacks on SGX for the first time based on Intel processor’s Specter vulnerability exposed in 2018. They successfully stole Intel’s key information embedded in SGX through side channels [20]. In addition, the team also found that AMD SEV, the trusted execution environment of the chip manufacturer AMD, had similar hardware design defects, such as untrusted I/O, ASID abuse, ciphertext side-channel vulnerabilities, etc. In 2020, a joint team from the University of Birmingham in the UK and the University of Vienna in Austria exploited the “Plundervolt” design of Intel hardware processors (i.e., implanting malware into the target computer to temporarily reduce the voltage flowing into the Intel chip) and succeeded in bitwise inversion of the key stored in SGX’s secure enclave [21].

A team from Graz University of Technology in Germany used Intel SGX to develop a piece of “enclave” malware—SGX-ROP attack software and deployed it in the SGX Enclave, successfully bypassing the attack detection of ASLR, stack canary, and address cleaner. The attack software can be used to steal or encrypt documents for ransom, send phishing emails, or launch denial-of-service attacks. Their research confirmed that SGX does not protect users from harm but constitutes a security threat, facilitating the so-called super malware launching vulnerability

attacks on the system at any time, for which traditional means such as antivirus software are ineffective [14].

The DHR architecture naturally has the attributes of TEE in principle, and there is no “soul torture” about whether the “trusted root” is trustworthy, and there are no complicated problems in encryption algorithm and key engineering management. In theory, it can be independent of (but can be integrated with) any security defense knowledge base. Its secure enclave function ensures that no matter the data stored in the architecture or the processing of sensitive data, it will not be threatened and damaged from inside and outside due to the ESS problems of software and hardware in the TEE platform. That is, DHR ensures that no differential-mode security defect in the architectural environment will become a security threat to private or trusted computing applications at any time. In other words, DHR has not only the mathematical encryption properties of structural coding but also the physical encryption properties of the operating environment and even the mathematical-physical encryption properties formed by the interweaving and superposition of the two. Compared with the simple information or data encryption mechanism, the DHR architecture’s iterative algorithms such as strategic ruling and feedback scheduling are naturally random, and its encryption characteristics of “one differential-mode perception, one environmental reconstruction” make APT-like human trial-and-error attacks unable to achieve convergence. More importantly, even if the defensive side has a “mole,” it is difficult to grasp the specific situation in the current operating environment in real time, thus greatly reducing the complexity of the “internal defense” work in the operation and management process, which is one of the desired goals of the “double-blind effect” of the DHR architecture. Theoretically, if there are diversified trusted software versions that are functionally equivalent, the DHR architecture can reduce the potential security threats that have not been detected in these software versions to a considerable extent. In short, the information processing or control system empowered by the DHR architecture is able to provide a confidential computing or trusted computing environment that integrates high trustworthiness, high usability, and high reliability with endogenous security and safety.

It should be emphasized that, unlike the technologies that cannot give quantifiable security indicators such as secure processor or secure operating system or secure database that cannot avoid soul torture, cyber resilience or toughness of the DHR architecture can have a variety of dimensions, which can be based on the reliability analysis model (e.g., GSPN) to give a probabilistic expression of security indicators, based on the theory of cryptographic strength to give the threshold of resistance to brute-force cracking, based on the game theory to give a quantitative expression of the equilibrium of the noncooperative game, and even based on the theory of complexity science to give the construction of resilience indicators and other multidimensional, multiperspective metrics and assessment methods. The ideal goal of cyber resilience is to design and manufacture “safer digital products,” including “out-of-the-box mission assurance” under the premise of “trusted service and default security.” However, current approaches to cyber resilience engineering have not yet moved beyond the traditional best-effort approach to security nor have they been able to provide trusted execution environments for trusted services with

security quality assurance. In the following sections, we will focus on how the DHR architecture enables the practice of cyber resilience engineering for trusted execution environments.

3.8 Restricted Applications

As an empowering technology to solve the multi-intertwined problems of functional safety, network security, and even information/data security, the DHR architecture is of universal application significance. In particular, it can provide an exponential security gain for sensitive applications that require high reliability, high usability, and high credibility, and it can also play an indispensable supporting role in some confidential computing or trusted computing application scenarios where trustworthiness cannot be guaranteed. Nevertheless, DHR is not a “one-size-fits-all technology.”

1. High Precision-Synchronization/Delay-Sensitive Application Environment.

Different from the implementation difficulty of isomorphic redundancy engineering, the DHR architecture has different requirements on hardware and software implementation structures or algorithms and pushes up the algorithm complexity of strategic ruling. Therefore, it is more difficult to ensure the accurate synchronization of heterogeneous executors in output response time (such as microseconds or even nanoseconds) in engineering implementation, and it is also impossible to avoid the delay influence and system overhead caused by the introduction of the ruling algorithm itself. This is indeed a big challenge for application scenarios with strict delay and microsynchronization requirements. In fact, even classical isomorphic redundant systems cannot avoid the problems of insertion delay and high-precision synchronization caused by algorithms such as majority-based voting or the consensus mechanism or strategic ruling.

2. Scenarios That Have Time-Delay Constraints and Do Not Allow Postcorrection.

For a certain input incentive, it is difficult for the DHR architecture to fulfill the strategic ruling of multiple heterogeneous executor output vectors or states in real time under demanding response conditions. Therefore, it is usually designed to take the output of the first executor that meets the requirements as the default response of the system according to some weighting policy or historical performance or “first in, first out” convention, so as to meet the real-time response requirements of application scenarios to the greatest extent. Thereafter, a strategic ruling is made on the default result and the output vector or state that arrives later in a given timeframe, so as to decide whether to add modification operation to the default output or send an instruction to the downstream link to stop the subsequent operation. Therefore, in any application scenario that has strict delay requirements and does not allow “first in, first out, and postcorrection,” the design

of DHR architecture's ruling algorithm needs to make a compromise between security quality and service performance.

3. **No Normalizable Input/Output Interface.**

The DHR architecture requires the input and output interfaces of reconfigurable executors with functional equivalence to be normalizable or standardizable. On the interface, under the excitation of a given input sequence, the output response or related states of functionally equivalent reconfigurable executors or scenarios are identical in a high probability. In other words, based on the interface, the equivalence between executors can be judged by means of consistency testing of a given function or performance. For complex cyber-physical systems or digital products, engineering technologies often cannot give a complete test set, especially the exception handling algorithm beyond the protocol or standard. The designs of equipment manufacturers are often so diverse, especially the alarms or alerts related to the security posture of the operating environment, that the equivalence of the functional performance of the executors is only valid to the extent that a given test set can be covered. Therefore, an important prerequisite for the engineering application of the DHR architecture is that there are input/output interfaces or related state information that can be normalized (not necessarily standardized) and can be tested for equivalence.

4. **Lack of Diversified Versions of Application Software.**

Heterogeneous redundant configuration of software and hardware processing resources is a prerequisite for the effectiveness of the DHR architecture, which generally requires the executor to be independent in physical and logical space. In the long run, COTS-level products such as basic and supporting software, hardware, and middleware, embedded systems, and IP cores with broad market applications will be increasingly diversified. In particular, the development of open-source community ecology and business format has greatly lowered the threshold of diversified product development with functional equivalence. However, personalized third-party applications or service software often lacks diversified versions in the market (app software with a large audience can often provide diversified versions for mainstream host systems or platforms, such as software versions based on processors such as x86, ARM, and RISC-V, as well as versions for different operating systems such as Windows, Linux, IOS, and Android, while highly professional application software products often have stronger directionality or dependence on the operating environment). If the source codes are available, special diversified compilation tools can be used to generate the running versions required for heterogeneous executors. If there are only binary executable files, although we can use some limited differential processing methods such as binary decompilation and diversified compilation, it will still have a significant impact on the expansion of the DHR architecture application field. The diversity of application software based on cross-platform scripting languages (such as Java) requires designers to be able to insert personalized "fingerprints" into the DHR running environment. Engineering practice shows that deploying different, lightweight, and transparent-to-intrinsic-function

commercial security plug-ins or components for cross-platform app versions in multiple execution environments can often cause considerable trouble to attackers. In a word, the diversity and heterogeneity of application software have much to do with the probability of common-mode escape of the DHR architecture.

5. **“Black Barriers” to Software Update**

Generally speaking, the software version upgrade of each functionally equivalent executor in the DHR architecture is likely to be out of sync (which is usually unavoidable when using COTS-level products), and there may be two situations. First, version updates that do not involve the content of the normalized interface of input or output (such as patching, improving performance, optimizing algorithm, etc.) can be considered a “transparent upgrade” for the DHR architecture system if the output vector or state used for strategic ruling is not changed. The second is the situation where a version upgrade affects the normalized interface or changes the vector of strategic ruling. In this case, the function of setting the output weights of the relevant executor according to the software update can be introduced into the DHR system. When the versions are different, we can specify that the functions expanded by the higher version have higher output weights (in fact, most functions should be the same in view of the compatibility requirements of the old and new versions). When inconsistencies are found in the ruling process, weight values can be introduced to make a further strategic ruling. Of course, this “nontransparent” processing method will more or less reduce the target system’s ability to deal with security risks during version upgrades.

6. **Cost-Sensitive Areas.**

It should be emphasized that in the era of digitalization today, the design and manufacturing complexity of software and hardware products is no longer the main factor determining the market price. Virtualization, heterogeneous multicore processors, user-customizable computing, software-definable functions, CPU + FPGA, software-defined hardware (SDH), RISC-V processors, 2.5/3D packaging, wafer/sub-wafer-level heterogeneous core-particle integration (SoW/CoWoS), and software-defined system-on-wafer SDSoW products that change the physical realization form of digital system technologies will make the application of DHR no longer dragged down by factors such as cost or volume power consumption. For instance, the world’s first DHR MCU chip, which is expected to be put into production on a large scale in 2Q in 2024, has a cost increase of <16% compared with the benchmark product with traditional architecture, and its manufacturing cost is less than one-third of the traditional benchmark products with high-reliability architecture, showing obvious cost-performance advantages.

7. **Problems of the DHR Architecture as a Highly Robust Software Architecture.**

Theoretically, the DHR architecture also has important application significance in enhancing the dependability and trustworthiness of software products. Software vulnerabilities are really difficult to eliminate under today’s technical conditions, the crowd-creation model of the open-source community may lead to

problems such as trapdoors (malicious codes brought in unintentionally), and state explosion problems make formal proof not always feasible in engineering, but DHR is not worried about these seemingly intractable problems in principle. The biggest challenge at present is the software design and development cost of cyber resilience empowered by the DHR architecture (some data show that it needs to be increased by about 19%), followed by the problem of software operation efficiency. This is because an interface operation may need to strategically call multiple functionally equivalent heterogeneous modules, and it is also necessary to make strategic ruling about multiple output results, which may be financially unbearable in a single processing space and an environment of shared computing resources. Fortunately, with the popularization of new computing architectures such as multicore, many-core, user-definable computing, mimic computing, and on-chip systems, we can use the parallel processing technology to solve the problem of simultaneous computing of multiple heterogeneous software modules, and the cost of strategic ruling can be dealt with by preprocessing the output results or states in functionally equivalent heterogeneous modules (such as performing some coding calculation or only paying attention to the compliance of state transition, etc.) so as to reduce the processing complexity in the stage of strategic ruling. The rest is the inherent problem of the heterogeneous redundant system itself, that is, the increase in design and maintenance complexity and cost (in fact, apart from the problems of version development, upgrade, and maintenance, the DHR architecture software can greatly reduce the cost of security maintenance for manufacturers and users in the whole life cycle of the product, and because there is no need to take into account 0-day or n-day vulnerabilities, there is no urgent after-sales service demand for patching and upgrading). The author believes that these will not be a problem if DHR can be applied on a large scale. In today's increasingly severe network security threats in particular, we need to update the traditional product design mode that only pursues functional performance without considering the safety of digital products, return to the basic law of commodity economy that "product design defects (including safety defects) are the responsibility of product providers or manufacturers," establish a new concept that "safety is not only product quality but also a service," and support the development trend of ubiquitous application of cyber resilience.

8. Ruling Problems.

Due to the heterogeneity of a DHR execution environment, the output response generated by the executable code of the same source program version will be different. Even in a homogeneous redundant environment, there are cases where the output response time or the relative input sequence is out of order. For instance, the starting sequence number of TCP in the IP protocol stack of an operating system is often random; the contents of optional or undefined extensions in IP packets are uncertain when using different protocol stacks; the process scheduling strategy and timing of the operating system are different; the encryption algorithm used in the output data packet may be strongly related to the host environment parameters; and some access authentication modules often need

randomized graphics or verification codes. Theoretically, between the parameters containing these uncertain contents, it is impossible to make a transparent ruling that has nothing to do with syntax and semantics. At this point, some creative solutions need to be introduced. For instance, if conditions permit, we can adjust the functional arrangement between the output agent and the executor, adopt a unified protocol stack version, share the random number generator, use the mask shielding technology, make rulings about the detection vector generated by the symbiont, compare environment-related parameters or states, and use the output results of intrusion detection, threat perception, and related security protection mechanisms. It should be noted that too much ruling comparison will cause considerable insertion delay, so additional functions such as finding output vector checksum and hash value need to be added to the executor to reduce the time cost of ruling comparison. For instance, when making a ruling for a 1 kb data packet, the ruling complexity can be greatly reduced by just adjudicating the hash value or check sum of the data packet. Similarly, when the executors use the same encryption and decryption algorithm but different keys, it is definitely not feasible to directly adjudicate and compare the encrypted data packet. If a hash value is formed and placed in a certain domain of the encrypted data packet before the data packet is encrypted, the ruling comparison of the ciphertext data packet can be changed into the comparison of only the domain where the hash value is located. Obviously, ruling is a complex issue that requires special attention in the implementation of DHR engineering, especially when using “black box” software because “being difficult to modify” is often more challenging.

Fortunately, in the past 10 years, cyber-resilient system engineering has become the development trend of the new generation of information technologies and digital products around the world. Advanced Western countries are introducing relevant technical standards and market access regulations one after another, mandatorily requiring digital products on the manufacturing side to have “mission assurance ability to deal with any potential threats and sabotage.” In this context, cyber resilience will be an indispensable function of digital products in the future. DHR, because of its high trust worthiness, high usability, high dependability, and integration, will help to make up for the major defects of cyber resilience engineering at present. With this process, the above limitations are believed to harvest more innovative solutions as cyber resilience engineering evolves. From the next chapter, we will discuss why cyber resilience engineering needs the empowerment of ESS.

3.9 Backdoors in Application Software

Generally, backdoors/trapdoors in the source codes of application software belong to “intentionally implanted” or “unintentionally imported” codes, and usually, the “semantics” of intrinsic service functions will not change due to the diversified compilation of versions. In other words, if there are semantic-level backdoor/

trapdoor functions in the source codes, including “environment-independent” backdoor functions such as obtaining private information, deliberately modifying data, remote monitoring/maintenance, and even maliciously stopping service, it is difficult to eliminate the influence of such problems in the DHR operating environment even if a variety of object codes are formed by diversified compilation methods. This is because the endogenously safe and secure “structural coding/environmental encryption” mechanism is transparent to the intrinsic function of the target system or is regarded as a “plaintext without the need for encryption and decryption.” Therefore, the backdoor problem in the “semantics” of intrinsic functional design or the trapdoor problem introduced by the technology of “reusable software and hardware” cannot be solved only by the DHR architectural effect in theory. However, the differential-mode backdoors or virus Trojans that the attacker may create in the DHR environment are not included in the insoluble range.

In particular, the popularity of open-source community model or the formation of ecological environment will easily lead to the same-origin security defects and cross-platform common-mode (common-cause) attacks, which, in mechanism, bring a challenge to the effectiveness of endogenously safe and secure cyber resilience and affect the stability of ESS of the complete DVR intersection. Therefore, when constructing the DHR environment, on the one hand, it is necessary to identify and avoid the configuration of homologous software and hardware at the engineering implementation level; on the other hand, it is necessary to automatically insert or deploy some lightweight security functions that are transparent to intrinsic services or functions, have different security defense mechanisms, have threat perception or behavior monitoring or data (library) protection functions, and can be called in the form of middleware and function (libraries). In a word, it is necessary to impose “transparent” security measures on diversified versions of application software running in the DHR environment and try to include “operating environment-independent” or “semantic-level” backdoor, trap door, virus Trojan, and other attacks into the defense scope of ESS mechanism.

It should be emphasized that, in theory, the expected effect can only be achieved if the empowered object meets the necessary preconditions for the application of the DHR architecture. Therefore, the security problem of application software itself cannot be completely solved by “environmental security.” The following situations are of particular concern:

1. If the application software belongs to “trusted software” and has a cross-platform running software version, the engineering design of the DHR architecture platform “only needs to ensure that its own security problems will not become a security threat to the trusted application software,” but it is still necessary to normalize the random output results that may be generated by the same application software in a distributed execution environment (e.g., if there is encryption authentication or the use of local random parameters).
2. If the application software does not belong to “trusted software” and the user has high security requirements, diversified running versions that are adapted to the

DHR architecture must be redeveloped with the participation of the source code provider.

3. Automated multiversion compiler tools need to be able to insert “barricade” functions or middleware codes that are strongly related to the environment and transparent to the intrinsic functions into the target codes, so as to enhance the dissimilarity of the environments between different executors, making it difficult for attackers to achieve the purpose of collaborative attacks using the same attack method.
4. Automated fingerprint comparison tools for vulnerabilities, backdoors, and virus Trojans are needed to find common-mode security problems that may exist in “diversified versions.” In the process of product design or use, it is also necessary to avoid software versions with high similarity running in the same environment through differentiated deployment.
5. Generally speaking, as long as the exploitation of vulnerabilities in application software is strongly related to the operating environment, the DHR architecture will be able to effectively defend against it. If there are “semantic-level” security threats such as backdoors, trapdoors, and virus Trojans that have nothing to do with the environment, they may lead to information security or even functional safety accidents, including nonperceivable common-mode escapes.

3.10 A Game-Rule Changer in Cybersecurity

“Vulnerabilities are inevitable and backdoors cannot be eliminated. Today’s scientific and technological capabilities are still unable to thoroughly investigate common ESS problems such as vulnerabilities and backdoors. This is the (not a) core problem of today’s cyberspace security defense.” However, the invention of the DHR architecture can undoubtedly change the rules of network attack and defense games based on the bottom drive of the digital ecosystem and reverse the current pattern of “easy to attack but difficult to defend” in cyberspace.

1. Theoretically, the DHR architecture with the property of complete DVR intersection has no use value under the same attack surface even if there are security problems such as vulnerabilities and backdoors—whether they are “known unknown” or “unknown unknown,” as long as they exist or are expressed in a differential mode. Similarly, even if there exist the same attack resources within the DHR environment such as vulnerabilities known to the attacker or backdoors and malicious code created later, they cannot constitute a stable common-mode escape if environmental decryption cannot be completed.
2. In practice, even if there are security problems such as diversified differential-mode vulnerabilities and backdoors in the DHR operating environment, it is difficult for network attacks across executors or in noncollaborative scenarios to achieve collaborative exploitation under the conditions of strategic ruling, feedback iteration, structural coding, or environmental encryption. Similarly, even if there are the same attack resources such as vulnerabilities, backdoors, and virus

Trojans known to the attacker in the DHR environment, they will not constitute a stable common-mode escape if environmental decryption cannot be completed.

3. Based on the dynamic iterative convergence mechanism of the executors and the effects of cleansing, recovery, and reconfiguration, the DHR architecture can effectively block or disrupt collaborative trial-and-error or blind attacks (such as APT attacks). “Even if a common-mode escape occurs, its stability is difficult to maintain,” which makes the attack experience and attack results nonreproducible and noninheritable.
4. The effectiveness of the ESS function of the DHR architecture is weakly related or even unrelated to how well the attacker has understood the ESS principle and method, whether he has known the backdoors, trapdoors, and other design defects of software/hardware components and where he has operational advantages. It is weakly related to the timeliness of users’ security maintenance of cyber-physical systems or digital products and operators’ past experience or technical ability.
5. Within the DHR architecture, strategically configuring some add-on security means or ruling parameters and feedback scheduling algorithms that are transparent to architectural function P will not only improve the system’s security performance exponentially but also enable users on the user side besides the manufacturing side to freely collocate according to the “selection list” given by the manufacturer and even add some private security measures that do not affect the intrinsic service function of the architecture. These measures can add much more uncertainty to the attacker’s efforts to reverse analyze the same technical products and achieve “environmental decryption.”
6. The GFS characteristics of the DHR architecture not only make the empowered cyber-physical systems or digital products naturally immune to the intertwined problems of functional safety and cybersecurity (even information or data security) but also provide a practical specification of cyber resilience engineering with quantifiable design and verifiable measurement.
7. The data storage or information processing capabilities provided in the DHR architecture naturally have the attribute of TEE, which can provide a “safe enclave” with high trustworthiness, high usability, and high dependability for trusted computing or confidential computing even if there are various differential-mode security problems.

In short, the empowerment of the DHR architecture will undoubtedly enable the new generation of information technology or cyber-physical systems or digital products to have “structure-determined” GFS or endogenously safe and secure cyber resilience. It not only challenges the current gray or black industrial chain and trading market based on vulnerabilities, backdoors, and virus Trojans to change the rules of the game but also opens up a new direction for the manufacturing side and the user side to achieve cogovernance of cyberspace, completely subverts the current mainstream attack theories and methods against the fragility of software and hardware design, and provides new theoretical and engineering methods for the security, openness, advancement, and trustworthiness of digital products to move from the current “either-or” state to a new stage of “unity of opposites.”

After answering the questions of what the ESS mechanism is, how to realize it, how effective it is, and what defects exist, we need to solve the problem of how to test or determine whether a given cyber-physical system or digital product has ESS functions and performance.

3.11 White Box Testing of ESS Functions

First, it is assumed that all the intrinsic functions and performances of the tested system have met the design requirements, and then the security defense capability of the tested system is evaluated. Usually, the latter mostly adopts vulnerability scanning, penetration testing, code auditing, risk assessment, security assessment, and other means, of which the latter three items are mostly offline or online. Vulnerability scanning and penetration testing are commonly used online testing methods, in which:

- Vulnerability (including virus Trojans, etc.) scanning refers to checking the security vulnerabilities of designated (attack surface) remote or local systems through scanning based on the prior knowledge base or known feature information, to find low-, medium-, and high-risk security vulnerabilities.
- Penetration testing is a simulated hacker attack under “authorization” based on the attacker’s thinking within the system being tested, including internal scanning based on vulnerability database, virus Trojan database, abnormal behavior feature library, etc., aiming at evaluating the security of the target object and finding potential security risks before the attacker exploits them. In other words, penetration testing means that the infiltrator tests the components in the “key or sensitive” parts of the tested system by possible access means (such as internal network interfaces, connecting terminals and standard physical ports, etc.), to find or dig out the known backdoors or virus Trojans that may hide in them, including weak passwords, unclosed idle ports, encryption algorithms with substandard strength, etc.

White box testing of ESS functions [3] is essentially a kind of “sensitive” functional segment of running codes in the execution environment. By manually inserting “hooks” or “loading test modules,” the unknown and nondestructive functional codes of “vulnerabilities, backdoors and virus Trojans” of the tested object are “implanted” into the corresponding executors of the current running environment, so that testers can inject or modify “functional codes needed for white box test cases” through channels connected with the outside of the system. Then, the activation sequence of “vulnerability, backdoor, and virus Trojan” test items that can “bypass” the channel compliance check is sent through the input channel or attack surface of the system to observe whether the output response expected by the test case design appears at the output end. What needs to be emphasized is that all the implanted test function codes are “unknown” to the

manufacturer and user of the device being tested. Therefore, the “white box injection/white box plug” test is the golden test of ESS functions.

1. Test Object.

Taking the triple modular redundancy (TMR) model of the DHR architecture as an example, A, B, and C, respectively, represent the three heterogeneous executor environments of the system being tested ($N = 3$ in this case), and the three have the equivalent service function intersection P .

2. Test Purpose.

Under the condition of injecting differential-mode and common-mode test cases in advance, we observe whether there is an abnormal response output (escape) at the output end of the tested system caused by the injected differential-mode or common-mode test cases and the input sequence and check the ESS function or performance of the tested system that can be quantifiably designed and verifiably measured. It should be emphasized that all dark-function test cases injected through the white box method are prepared in advance by the testers, but for the system under test, they should all be regarded as “unknown vulnerabilities, backdoors, or virus Trojans.”

3. Differential-Mode Injection Test 1.

Test steps: Inject differential-mode test cases (dark functions unknown to tested object) into executor environments A, B, and C, respectively, and then inject the test case activation sequence from the input end to observe whether there is a response sequence at the output end.

Expected effect: The tested object should not be able to obtain the response sequence of the tested sequence and the embedded differential-mode dark function codes from the output terminal. A 100% differential-mode inhibition performance is verified, as shown in Fig. 3.21.

4. Differential-Mode Injection Test 2.

Test steps: Pack the differential-mode dark function codes x , y , and z , and inject them into executor environments A, B, or C in sequence, then input the attack test sequence from the attack surface, and observe whether the attack response sequence appears at the output end.

Expected effect: No matter how many kinds of differential-mode dark function codes exist in an executor environment and how they are distributed, they should be 100% invalid for the ESS architecture system, and theoretically all injected differential-mode dark function codes should be able to be automatically removed or made unexploitable to the current attack during the reset, reconfiguration, or replacement process of the host operating environment, as shown in Fig. 3.22.

5. $N-1$ -mode Injection Test.

Test steps: Implant the same dark function codes into the AB, BC, and AC heterogeneous executor environments respectively to form a common-mode distribution of $N-1$, then input the related test case activation sequence from the attack surface, observe whether the response sequence appears at the output end,

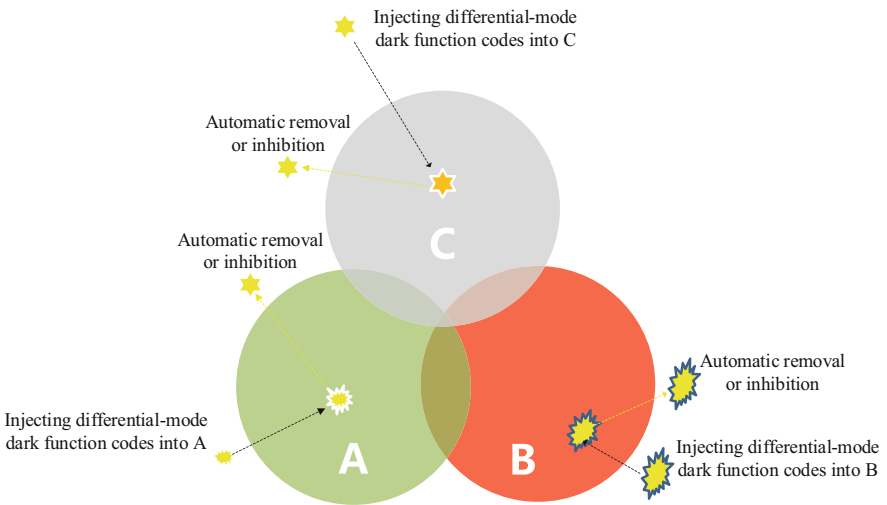


Fig. 3.21 Injection of triple-modular-redundancy differential-mode test cases into the DHR architecture

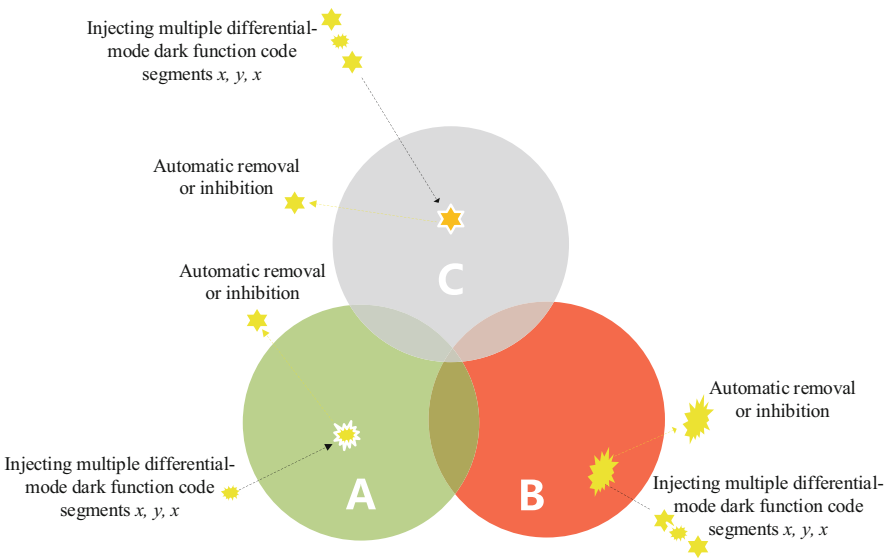


Fig. 3.22 Injecting multiple differential-mode dark function codes in sequence

and verify the probability perceivable common-mode escape and the performance of feedback convergence, as shown in Fig. 3.23.

Expected effect: The strategic ruling and backward iterative verification mechanism of the DHR architecture should be able to ensure that even if there are $N-1$ common-mode test cases in the system, once a perceivable

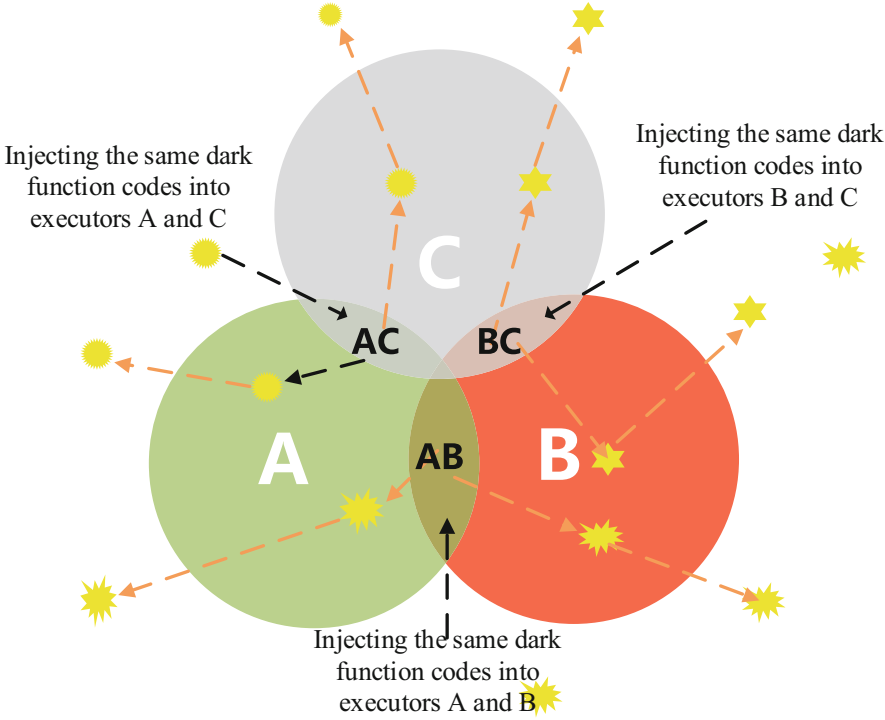


Fig. 3.23 Test of $N-1$ perceivable common-mode escape

common-mode escape occurs, the system will automatically converge from the perceivable common-mode escape state to the differential-mode transient steady state until it returns to the stable operating state, and the maintenance time of the perceivable common-mode escape state should be less than the threshold allowed by design.

6. *N*-mode Injection Test.

Test steps: Implant the same dark function codes into three heterogeneous executor environments of A, B, and C at the same time to form an N -mode common-mode distribution, then input the test sequence from the attack surface, observe whether there is an test case response sequence at the output end, and verify the function of command disturbance to exit the N -mode escape state and the feedback convergence time, as shown in Fig. 3.24.

Expected effect: The system’s external strategy perturbation mechanism will force the executor injected with the N -mode dark function codes to automatically degrade to the perceivable common-mode ($N-1$) state, and then degrade to the differential-mode working state. “Even if a non-perceivable common-mode escape occurs, the stable maintenance time of the attack chain will not exceed the time allowed by the strategy disturbance algorithm.”

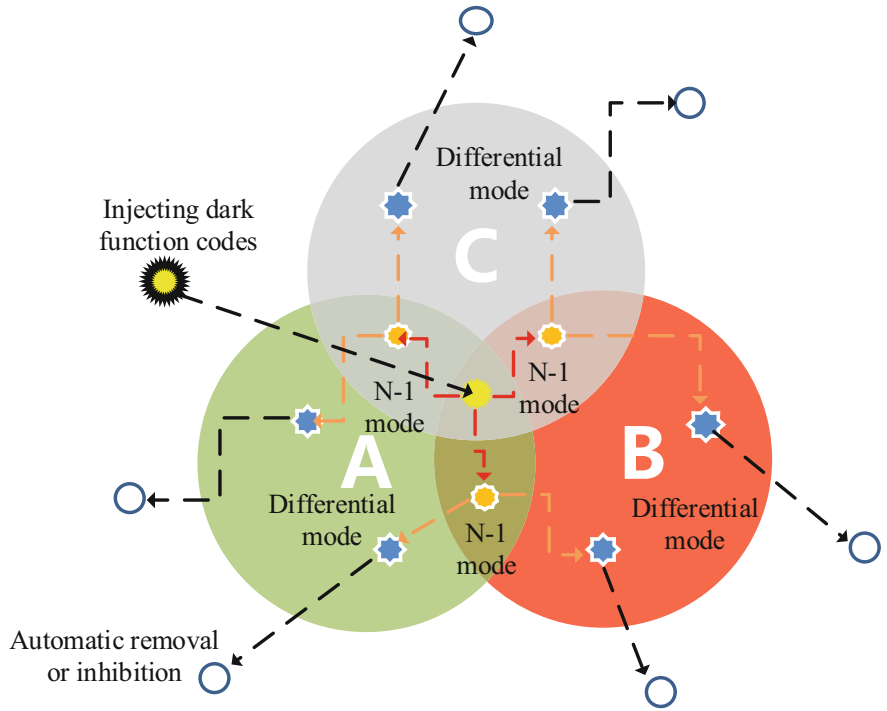


Fig. 3.24 The external disturbance mechanism releases the common-mode escape state

7. Control Loop Injection Test.

Test steps: The control loop is composed of input agent, strategic ruling, feedback control, and error-correction output components. Except for the components realized by hardwired logic (because test cases cannot be embedded), as long as the components are controlled by stored programs and allow on-site modification of program codes, they should be able to embed the dark function of test cases in general, and then input the attack sequence from the attack surface of the system.

Expected effect: Observe at the output end whether the test response sequence generated by the synergy between the attack sequence and the preimplanted dark function can be obtained.

It should be pointed out that the error-correction output, strategical ruling, and feedback control components have a reconfigurable operation scenario in the middle relative to the input agent, which is not directly accessible from the attack surface. So, the probability of a successful attack should be very low unless the attacker can not only grasp the vulnerabilities and backdoors of a certain component in the control loop but also grasp the software and hardware vulnerabilities and backdoors in individual executors in the current operating environment in real time and can accurately inject attack codes through the input end. What is

more, control components in real applications are usually designed according to hardwired logic and are quite private. Without prior knowledge of related software and hardware codes, attackers have insurmountable challenges in trying to obtain unknown vulnerability information and create the attack chain simply by scanning and sniffing. In addition, the strategic ruling and feedback control loop ontology is part of the DHR architecture, so the inherent ESS gains of the architecture can be obtained. Its concise logic and specialized functions are transparent to and independent of the reconfigurable executors in the DHR architecture. The refined design ensures that even if there are unknown vulnerabilities, they will not be exploitable. The functional safety of its monotonous and simple design can usually use formal proof tools to ensure that there are no backdoors, trapdoors, or as few vulnerabilities as possible. Trusted computing, dynamic defense, or even DHR iterative technologies can be used to enhance the robustness of control loop components themselves and can be used in a standardized embedded module manner. Just like the commercial cipher machine, if the feedback control, strategic ruling, or trusted storage areas are given special functions such as customizable private algorithms, user-definable data, or sensitive area access control authorization, their trustworthiness can be guaranteed in application.

8. Test Analysis.

For systems with ESS functions, the white box injection test is expected to achieve the following goals:

- (a) Observe whether the tested system can transform various test disturbances into simple differential-mode or common-mode problems in the DVR domain without prior knowledge.
- (b) Theoretically, it should be able to suppress or shield 100% differential-mode test disturbances within the structure, and usually the artificially injected software test cases can be automatically cleared or bypassed.
- (c) Observe that when a perceivable common-mode escape problem occurs in the operating environment of the tested object, it can recover resiliently.
- (d) Observe that the operating environment of the tested object can still be automatically released within the designed time after it enters the nonperceivable common-mode escape state.
- (e) Even if there are vulnerabilities and backdoors in the control loop, it will be difficult to exploit them through the attack surface.
- (f) Give quantitative metrics for relevant tests and assess how well the security design of the tested system meets the standards.

3.12 Promotion of Diversified Ecological Development

The rapid development of information technologies has provided a solid engineering foundation for the diversity of software and hardware, but the rich diversity has also led to a wide variety and huge number of security problems such as vulnerabilities and backdoors, which seems to be an “eternal” disaster with no end and hope for the precise defense that relies on the acquisition and perception of attack features. However, it is a rare development opportunity for the DHR architecture based on the theory of ESS because the diversity of software and hardware components provides a good heterogeneous foundation for the commercial application of the DHR technology. As far as the web is concerned, 1550 heterogeneous web service executors can be easily realized through the heterogeneity of the five-layer software stack [22]. Furthermore, the hardware platform hosting the software can also be made heterogeneous, and more types of web executors can be obtained using different processor platforms. The author borrows Dickens’s famous saying from *Oliver Twist*: “It was the best of times; it was the worst of times.”

1. “Isomorphic, Heterogeneous, and Diverse” Ecology

As the “steel skeleton” of cyber-physical systems or digital products with “reinforced concrete texture,” DHR is often inseparable from standardized and diversified components with equivalent functions or the support of industrial ecology like the “concrete supply chain.” The traditional network security industry, open-source community, cross-platform computing, customizable computing, heterogeneous computing, confidential computing, trusted computing, software/hardware definability, functional virtualization, RISC-V, and other models and technologies can naturally support such development requirements. The DHR architecture empowers digital products to create new demands for “supply-side” cyber resilience/toughness technology. The market of homogeneous and heterogeneous products is no longer just an exclusive zero-sum game, and diversified add-on security products can bring exponential security gains to the DHR architecture in the form of “concrete.” The empowering effect of DHR has boosted the market in turn, bringing stronger openness (preventing the trend of generalizing the concept of national security and deliberately creating barriers to trade liberalization), complementarity, and diversification and providing valuable market space for the development of similar nonhomologous and innovative products.

2. New Ways to Accelerate Product Maturity.

DHR architecture products can accelerate the improvement of functional performance of immature heterogeneous components or components. Generally speaking, there are differences in technical maturity between functionally equivalent components or components, especially for comers to the market or the initial application stage of products. By adding on-site recording and log analysis functions to the DHR architecture, the design defects and performance weaknesses of new products can be found as early as possible by online, quasi-online, or offline methods through the identification of and comparison between the

operating conditions of executors. The DHR's differential-mode perception and error-correction mechanism are expected to gain four benefits:

- (a) As long as there are no common-mode ESS problems in the diversified elements or components used in the DHR architecture, there is no need to worry about the dependability of products whose technical maturity is yet to be improved.
- (b) Because of the "homogeneity and heterogeneity" demand of DHR, late-comers to the market are no longer blocked by the exclusive barriers created by the first comers who "ride on a horse to demarcate the land," which has opened up a new blue ocean for homogeneous and heterogeneous products.
- (c) Cyber-physical systems or digital products with dynamics, heterogeneity, and redundancy can mechanically tolerate the "differential-mode" design defects in the redundant body, thereby significantly reducing the verification workload in the design stage, shortening the trial time and cost of products, and speeding up the process of new products entering the market (the opposite is true for cyber resilience design).
- (d) Because the DHR architecture does not require strict "self-proof of innocence" for the "concrete materials" used, there is no need for various existing network security products to guarantee that there are no ESS problems. In other words, only the differences between concrete components are required, and the components themselves are not required to be perfect.

3. Complementary Forms of Autonomous Control.

The DHR architecture requires the task processing environment to be constructed with a variety or diversity of hardware and software entities or virtual bodies, but it does not require the components themselves to have no common ESS problems such as vulnerabilities, backdoors, or potential virus Trojans. This makes it possible to use some COTS-level products whose supply-chain trustworthiness cannot be guaranteed but whose functional performance and maturity are good in a globalized or multipolar industrial ecology in combination with trusted products, which have a high degree of autonomous control but which are not yet advanced and mature enough. To do so, we can adopt a mixed configuration or monitoring work mode, introduce weight parameters into the process of strategic ruling, and add refined and intelligent ruling strategies such as performance priority, confidence priority, new function priority, and historical performance priority. In this way, we can, at the technical levels of architecture and components, give full play to the complementary advantages of the products that are autonomously controllable but lack maturity and those that are advanced and reliable but cannot ensure security. By doing so, we not only circumvent the security defects of the products but also ensure the dependability and usability of the system, so that the contradiction between advancement and trustworthiness and that between openness and security can be unprecedentedly unified on the basis of the DHR architecture, thus greatly reducing the initial investment of digital products and their whole-lifecycle costs of use and maintenance.

4. Creating an Integrated Cyber Resilience Environment.

Cyber-physical systems not only have to control traditional functional safety issues everywhere but also have to deal with nontraditional security threats all the time, especially in sensitive industries concerning life and property safety, such as next-generation digital infrastructures, cyber-physical systems, industrial controls, embedded applications, special application fields, and even smart networked vehicles.

The GFS feature of DHR can tolerate design defects to a great extent in mechanism (theoretically, the reliability of DHR is improved by an order of magnitude compared with the classical DRS architecture under the same redundancy [1]), as long as these defects cannot be simultaneously and successfully exploited to cause common-cause or common-mode faults among heterogeneous executors. In other words, whether it is a random physical failure, uncertain fault caused by software design, or a malicious function introduced by human factors in engineering implementation or operation, it will be found in the ruling detection link or “dynamically cleansed or reconstructed and reset” by the strategic mechanism unless it can produce a completely consistent common-mode output or state in the DHR architecture environment at the same time and ensure that “common-mode escape at all operation steps” can be realized. Therefore, the DHR architecture can deal with the threats and damages caused by multiple intertwined problems of functional safety, network security, and even information/data security and create, by empowering generalized functional safety, a new generation of highly dependable, highly trustworthy and highly usable cyber-physical systems and digital infrastructure that integrate cyber resilience and trustworthiness, or various digital products that have high requirements for privacy, integrity, and usability.

3.13 Summary of This Chapter

Using the innovative impossible triangle principle of defending against unknown threats—the IIP principle—we can not only qualitatively analyze whether a given network security technology has ESS but also roughly understand whether the target object can cope with generalized uncertain disturbances including unknown network attacks without relying on (but not rejecting) prior knowledge. According to the conjecture that there is ESS in the complete DVR intersection, the unknown security threats and damages to a cyber-physical system, whether due to random/uncertain perturbations, or due to human/nonhuman disturbances, can be expressed as differential-mode or common-mode security incidents that can be expressed by probability in the DVR domain. The theory of structural coding is established, and the ESS-existence theorem (ESS-ET) is proved, enabling us to propose an architecture of “perfect safety” with “structural coding or environmental encryption,” which can deal with generalized uncertain disturbances in an integrated manner in the DVR domain. By introducing a strategic ruling-based dynamic feedback control mechanism on the basis of the dissimilar redundancy structure (DRS), a dynamic

heterogeneous redundancy (DHR) architecture is invented, which not only transforms complex ESS problems into generalized reliability problems with differential-mode or common-mode properties in the DVR domain but also effectively controls the probability of “known unknown” or “unknown unknown” ESS problems being excited into ESS incidents. More importantly, the DHR architecture, in an engineering sense and in an integrated manner, can integrally endow cyber-physical systems or digital products with endogenously safe and secure cyber resilience or toughness that can be quantifiably designed and verifiably measured, thereby achieving a trinity of dependability, trustworthiness, and usability.

The core mechanism of the DHR architecture can be summarized as follows: ① The multiform, varied, and ever-developing influences of multiple intertwined problems of functional safety and network security (even information/data security) are mapped into differential-mode or common-mode local disturbance incidents expressed in a way of “knowing what it is but not knowing why it is so” in a DHR environment. ② The global static heterogeneous redundancy with high engineering complexity is transformed into local dynamic heterogeneous redundancy with technical and economic advantages. ③ The majority-based voting mechanism, which cannot distinguish between differential mode and perceivable common mode, is replaced by an iterative ruling based on strategy and multiple parameters. ④ The “before, during, and after event,” the bodyguard-style security protection mechanism that relies solely on “exhausting problems” or “patching vulnerabilities,” is transformed into the mode of endogenous security protection, such as shielding and correcting the output state, migrating the problem scenario, and tempospatially circumventing local security incidents. As an underlying driving technology and new paradigm for digital ecosystems “transparent to intrinsic functions” but structurally coded or environmentally encrypted for unintended dark functions, the DHR architecture can be a “method of methods” for cyber resilience engineering.

Thanks to the unique ESS features and structural coding of the DHR architecture environment, whether it is a known or “unknown unknown” attack, a random perturbation or uncertain disturbance, and caused by nonhuman factors or deliberate human behavior, it can be effectively suppressed by a unified mechanism as long as it is perceived as a differential-mode disturbance in a DHR architecture environment, and the generalized dependability or functional safety can be quantifiably designed and verifiably measured. There is no doubt that ESS plays an indispensable role in solving the three major hardcore problems of current cyber resilience engineering.

The inherent fusion characteristics of the DHR architecture can naturally accept all kinds of traditional network security technologies, so it can obtain “reinforced concrete-like” security gains, which is beyond the reach of various current security technologies piled up or added on outside or inside digital systems. More importantly, the endogenously safe and secure architecture can be used not only as the underlying technology supporting cyber resilience on the design/manufacturing side of digital products but also as a new paradigm of network security. It can not only reverse the serious imbalance in network security risks and responsibilities between the manufacturing side and the user side but also truly integrate network security into the design and manufacture of digital products.” It is not that two engineering teams

design two subsystems and use one to protect the other but that a single system is designed to integrate security capabilities as required. In addition, it has practical and long-term significance for opening up and developing new network security markets and technologies.

The ESS theory and methods have opened up a new paradigm for the development of network security defense, created a new perspective for defending against GFS problems, invented an empowerment method based on the DHR architecture, and developed practical engineering specifications of ESS that can be quantifiably designed and verifiably measured. To realize the transformation of the underlying driving paradigm of digital ecology, the development of cyber resilience engineering with integrated structural gains has opened up a new technical direction, which not only provides a surging development momentum for a new generation of digital products with “out-of-the-box default security” but also opens up a vast blue ocean of markets with “the same caliber” as the digital industry for the add-on network security industry. It can fundamentally resolve the sharp contradictions between development and security such as “the contradictions between openness and security, between advancement and reliability, between autonomous controllability and security trustworthiness, between functional safety and network security (even information or data security),” etc., and is of great significance for reversing the current strategic predicament that cyberspace is easy to attack but difficult to defend. In the following chapters, readers will see that the design and practice of ESS-empowered cyber resilience engineering is at least one of the ideal ways to break the vicious circle or dilemma of “patch after patch” in the current cybersecurity field.

References

1. Jiangxing W. Cyberspace endogenous security and safety—cyberspace mimic defense and generalized robust control [M]. Beijing: Science Press; 2020.
2. Malik F. Management: the essence of craftsmanship. Beijing: China Machine Press; 2013. p. 29–31.
3. Jiangxing W. An introduction to cyberspace mimic defense [M]. Beijing: China Science Press; 2017.
4. Jiangxing W. Development paradigms of cyberspace endogenous safety and security [J]. *Scientia Sinica (Informationis)*. 2022;52(2):189–204.
5. Yeh YC. Triple-triple redundant 777 primary flight computer [C]. Paper presented at: Proceedings of IEEE Aerospace Applications Conference, Daytona Beach; 1996. p. 293–307.
6. Voas J, Ghosh A, Charron F, Kassab L. Reducing uncertainty about common-mode failures [C]. Paper presented at: Proceedings of the Eighth International Symposium on Software Dependability Engineering. Albuquerque; 1997. p. 308–319.
7. Liu F, Yarom Y, Ge Q, et al. Last-level cache side-channel attacks are practical [C]. Paper presented at: Proceedings of IEEE Symposium on Security and Privacy; 2015. p. 605–622.
8. Shannon CE. A mathematical theory of communication [J]. *Bell Syst Tech J*. 1948;27(3): 379–423.
9. Haiquan L, Jian L. Computer network security and encryption technology [M]. Beijing: Science Press; 2001.

10. Ashby WR. Design for a brain. London: Chapman & Hall; 1960.
11. Xuesen Q. On systems engineering [M]. Shanghai: Shanghai Jiaotong University Press; 2007.
12. Shannon CE. Communication theory of secrecy systems [J]. Bell Syst Tech J. 1949;28(4): 656–715.
13. Raman D, Sutter B D, Coppens B, et al. DNS tunneling for network penetration [C]. Paper presented at: International Conference on Information Security and Cryptology, Heidelberg; 2012, p. 65–77.
14. Jin L, Hu XY, Wu JX. From perfect secrecy to perfect safety & security: cryptography-based analysis of endogenous security [J]. Secur Saf. 2023;2:2023004. <https://doi.org/10.1051/sands/2023004>.
15. Wu JX. Cyberspace endogenous safety and security [J]. Engineering. 2021;15(8):179–85.
16. Jiexin Z, Jianmin P, Zheng Z. Quantification method for heterogeneity on web server with mimic construction [J]. J Softw. 2020;31(2):564–77.
17. Jiangxing W. Research on cyber mimic defense [J]. J Cyber Secur. 2016;1(4):1–10.
18. Jajodia S, Ghosh AK, Swarup V, et al. Moving target defense: creating asymmetric uncertainty for cyber threats [M]. New York: Springer; 2011.
19. Katz J, Lindell Y. Introduction to modern cryptography [M]. New York: CRC Press; 2020.
20. Shih MW, Kumar M, Kim T, et al. S-NFV: securing NFV states by using SGX [C]. Paper presented at: Proceedings of ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, New Orleans; 2016. p. 45–8.
21. Kit M, David O, Flavio DG, et al. Plundervolt: how a little bit of undervolting can create a lot of trouble [J]. IEEE Secur Priv Spec Issue Hardw-Assist Secur. 2020;18(5):28–37.
22. Jiangxing W. Cyberspace mimic defense—generalized robust control and endogenous security [M]. Beijing: Science Press; 2018.

Chapter 4

Introduction to Functional Safety and Cyber Resilience



4.1 Review and Development of Functional Safety

Human beings' own safety is the first and foremost thing no matter how technological revolution and industrial changes affect people's production and life. As more and more electrical/electronic/programmable electronic (E/E/PE) systems are used in such fields as industrial control, power system, fuel supply, aerospace, automotive industry, robots, medical equipment, information weaponry, and combat systems for safety purposes, controlling the overall risk of these safety-related systems within the required range through appropriate technical and management measures to protect personal safety and the environment has become a global core concern and action goal—functional safety. Today, all industries are empowered or transformed by digital, network, and AI technologies, and functional safety is welcoming re-interpretation, re-adjustment, re-optimization, re-upgrading, and re-balancing.

While promoting the progress of the human society and profoundly changing human production and lifestyle, scientific and technological developments and the industrial revolution may also bring new security risks. For a long time, major industrial accidents and traffic accidents have occurred frequently, causing a large number of casualties. This has driven the emergence and continuous development of functional safety. In order to improve safety, since the 1960s, E/E/PE systems have been increasingly used in industrial production, energy, transportation, medical care, and other fields to undertake safety-related functions to protect workers and the public from injury or death and the environment from being polluted. However, the failure and malfunction of these safety-related systems themselves may also cause catastrophic accidents such as fire, explosion, radiation, nuclear leakage, etc. Therefore, reducing potential safety hazards to an acceptable range through safety technologies and safety management has become a major practical demand in many industries.

Safety-related systems used in the industrial field in the early days were mostly composed of relays, solid-state circuits, or hard-wired logic, whose faults were

mainly random hardware failures. The system behavior under fault conditions could be completely determined and predicted based on statistical theories and methods. However, troubleshooting was difficult in most cases. When one or several relays in a large relay group fail, or the contacts have poor contact, it may take several hours or even longer to find and eliminate the fault.

Around the 1970s, programmable electronic systems began to be gradually used to realize safety-related functions. Programmable logic controllers (PLCs) provided a flexible way to realize control actions, allowing a single device to replace the hard-wired relay groups the industry had been accustomed to. Early PLCs were mainly used for sequential control and could only implement less complex logical operations. With the development of microprocessor and microcomputer technologies, some manufacturers in the United States and Japan introduced microprocessors into PLCs in the mid-to-late 1970s, thus breaking the traditional PLC system technology architecture based on relay control.

In the 1980s, the introduction of microprocessor technologies made software-controlled systems more cost-effective. However, as the software in the systems became increasingly complex, it was more and more difficult to completely avoid the defects in software design and to judge every potential fault and failure in real time or on site. Based on this understanding, many researchers began to find new methods to avoid and control uncertain failures of safety-related software systems. By the mid-to-late 1980s, controller systems, PLCs, and solutions dedicated to safety-related systems developed rapidly, and heterogeneous redundant voting functions and diagnostic functions were gradually introduced into product design.

Since 1990s, with the continuous expansion of the scale of automated production, computer and network technologies have been widely introduced into safety protection systems. Compared with traditional hard-wired logic and PLC technologies which lack intelligence and flexibility, these technologies have significantly enhanced the safety control capability of factories and industrial processes. In this way, safety-related systems with a combination of software and hardware gradually occupy a dominant position, but at the same time, the complexity and uncertainty of these systems also increase rapidly, which is a new challenge. First of all, since software development is affected by the stages of technological development and the cognitive limitations of code writers, it becomes an extremely challenging task to verify the correctness of software functions, let alone completely eliminate design vulnerabilities, potential hidden functions, or unknown vulnerabilities. In addition, as hardware is getting larger in scale and more complex in structure, it is more and more difficult to verify the correctness of design functions due to “state explosion.” In the meantime, with safety-related systems getting larger, more systematic, and more complex, their functional safety, reliability, and continuous effectiveness have attracted great attention from the industry and academia.

In response to increasingly serious industrial accidents, Germany issued the DIN V 19250 standard “Control Technology: Fundamental Safety Aspects to Be Considered for Measurement and Control Equipment” [1] in May 1994 and released the world’s first classification standard for safety equipment of process industries AK1-8. Furthermore, Germany formulated the V VDE 0801 standard “Principles

for Computers in Safety-related Systems,” which provided a method for programmable electronic systems to meet DIN V 19250 level in the process of design, coding, program execution, and confirmation, marking the birth of the concept of functional safety. In February 1996, the Instrument Society of America (ISA) released the ISA S-84.01 standard “Application of Safety Instrumented Systems for the Process Industries” [2], which put forward the concepts of the safety integrity level and safety life cycle. The standard soon became a national standard and started to be legally enforced by the US Occupational Safety and Health Administration’s (OSHA) Process Safety Management (PSM) and the US Environmental Protection Agency’s Risk Management Program (RMP).

In May 2000, the International Electrotechnical Commission (IEC) officially released IEC 61508 “Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems” [3]. In fact, as early as 1988, IEC 61508 issued a draft standard for safety software and safety hardware. Later, the more complex question of how to regulate systemic failures was raised. With the Commission’s continuous efforts, the seven-part draft IEC 61508 standard was released in 1995. In 1998, Parts 1, 3, 4, and 5 of the draft were approved as international standards; in 1999, the remaining parts were approved as international standards. In 2000, the complete standard was released. IEC 61508 is a collection of technical concepts, scientific evaluation systems, and optimized management methods. It is expected to provide standardized safety guidance for systems containing electronic/electrical equipment and computer software/hardware when used in fields related to personal and property safety. Integrating the core contents of relevant standards of the United States and Germany, IEC 61508 was released as a basic international standard for functional safety. This marks that functional safety, as a professional discipline and engineering science, has begun to form its own system.

In September 2022, the US Cybersecurity and Infrastructure Security Agency (CISA) released the “2023–2025 Strategic Plan” [4]. The plan identifies four cyber security objectives, namely, network defense, risk reduction and resilience enhancement, service coordination, and unified organization, with a total of 19 sub-objectives for risk reduction and resilience enhancement, providing guidance for the work of network and infrastructure security in the United States. The core of the plan is to treat cyber security defense and infrastructure resilience and recovery capabilities in a unified manner, with the enhancement of resilience and the reduction of risks as an important part of the sub-objectives.

In March 2023, the White House released the “National Cybersecurity Strategy” and proposed that the cyber security strategy should shift from merely emphasizing application services to placing equal emphasis on manufacturing supply and the application services. Based on the concept of “network of networks,” the Strategy aims at building a trusted execution environment to ensure the security of key applications and data; it emphasizes strengthening collaboration and consistency among alliances to form greater defense capabilities against collaborative attacks.

4.2 Basic Concepts of Functional Safety

4.2.1 *Definition of Functional Safety*

According to the national standard GB8223-8 of the People's Republic of China, function refers to “an attribute of an object that can meet a certain need” [5]. A function can only be realized by relying on its carrier. When establishing the theory of value engineering, L.D. Miles put forward that when a customer buys something, what he needs is its function, not the thing itself, and the commodity is only the carrier of its function [6]. On the one hand, functions and their carriers must be distinguished conceptually; on the other hand, functions must be combined with their carriers because the former cannot be realized without the latter. Of course, the same function can be realized through different carriers, or in other words, the carrier of a certain function is not unique in most cases.

In English there are two related concepts: safety and security. Many literatures have given different interpretations of these two words from different perspectives. Literature [7] regards safety as avoiding unacceptable damage to people, property or the environment, and security as avoiding illegal or unnecessary infiltration into and interference with normal operation or improper access to confidential information. Literature [8] points out that, according to different situations in specific fields, safety refers to preventing incidents caused by carelessness, system failures, and natural phenomena, while security refers to eliminating all kinds of intentional threats and attacks. Therefore, the intention of the actor (usually a human) can be used to distinguish safety problems from security problems in most cases.

IEC61508 defines functional safety as an integral part related to equipment under control (EUC) and EUC control system, which depends on the correct functioning of E/E/PE safety-related systems, other safety-related technical systems, and external risk reduction facilities [3].

As the main body of functional safety, a safety-related system can take appropriate actions to prevent EUC from entering a dangerous state when it detects a situation that may lead to dangerous events. Obviously, the faults and failures of the safety-related system itself should also be classified as the events that can lead to danger. Safety-related systems are generally divided into safety-related control systems and safety-related protection systems. In different application fields, there are different kinds of safety-related systems, such as safety instrumented systems (SIS), key control systems, railway signal systems, fail-safe systems, interlocking protection systems, advanced driver assistance systems, automatic parking systems, etc. These systems are either used to reduce the probability of dangerous events or to reduce the impact of dangerous events and finally achieve the required or expected safety goals.

The boiler control system is a typical SIS. Its function is to detect the boiler pressure and turn off the furnace when the pressure reaches a dangerous value. If this function fails, the boiler will be at risk of explosion and put operators in danger. In this case, safety relies on the boiler control system performing its function correctly.

This situation where safety depends on system functions is called “functional safety.”

A safety-related system can be an integral part of the EUC control system, such as the automatic driving system of an autonomous vehicle, which is both a control system and a safety-related system of the vehicle; it can also be an additional, separate, or specialized system used to implement safety functions. Systems such as boiler control systems utilize independent pressure sensors and pressure relief devices to reduce safety risks.

4.2.2 Attributes and Differences

The definition of functional safety at least clarifies two basic attributes: objective and carrier. First, from the perspective of target attributes, safety does not refer to security. Therefore, the main objective of functional safety is to propose risk control measures to avoid unacceptable damage to personal safety and the environment. Second, in terms of carrier attributes, the carrier of safety function includes all the hardware, operating system, application software, communication, protocols, algorithms, etc., so the random hardware failure and system failure of these components and the whole system become the core concern of functional safety.

In order to understand functional safety more accurately, two main differences need to be clarified, namely, the difference between functional safety and intrinsic safety and the difference between functional safety and reliability.

Intrinsic safety is a method of ensuring safety by eliminating the causes for danger. Its prerequisite is to know the causes for danger and then completely avoid them in product or system design. Correspondingly, functional safety is a method of reducing risks to an acceptable level by designing functions to ensure safety. Its prerequisite is also to know the cause for the risk and adopt technical measures that can reduce the risk to a given threshold. A classic example of the difference between the two is safety measures when roads and railways intersect. As shown in Fig. 4.1, intrinsic safety can eliminate the risk of vehicle collisions at intersections by building an overpass to prevent railway trains from colliding with road vehicles. In contrast, functional safety considers a lower-cost approach to safety



Fig. 4.1 A comparison between intrinsic security and functional safety in principle

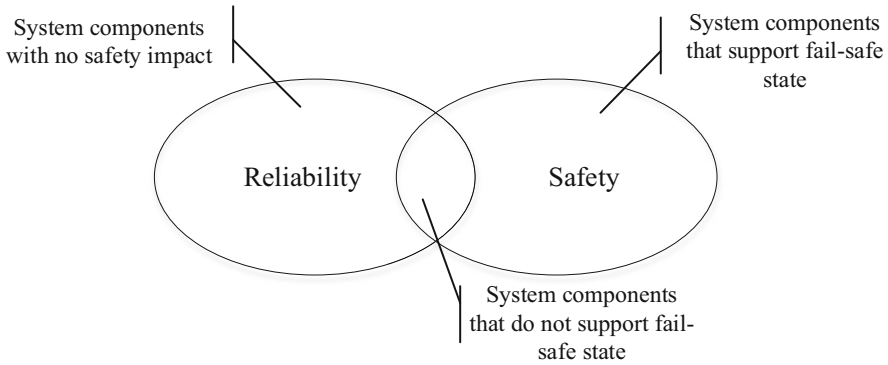


Fig. 4.2 Relationship between reliability and safety

by installing barriers, sensors, and alarms at the railway-highway intersection. When receiving the signal that the train is approaching, the system gives an alarm and puts down the barrier to stop road vehicles from passing and prevent them from colliding with the passing train. This approach does not intrinsically address the risk of collision, but it can reduce it to an acceptable level. Of course, the method of intrinsic safety is the most ideal, but in many cases, because of its high cost, or because the system itself cannot eliminate the sources of hazards, especially for a highly complex system like the on-board electronic control system, intrinsic safety can hardly be realized, so functional safety is widely accepted.

Systems are generally expected to be both reliable and safe. However, reliability and safety requirements are different. China's national standard GB/T 2900.99—2016 defines reliability as “the ability to perform requirements without failure under given conditions and within a given time interval” [9]. It can be seen that what is reliable is not necessarily safe. A reliable system may perform an action perfectly without failure every time, but under the disturbance of external factors, this reliable action may not be able to avoid or alleviate safety accidents. For example, a reliable random access memory only needs to correctly perform reading or writing operations and keep the stored information unchanged for a given time, but it doesn't care whether the operator has the right to read and write information. Likewise, what is safe is not necessarily reliable. For example, a smoke alarm may occasionally give false alarms and therefore can be considered unreliable. However, as long as it can always give an accurate warning when smoke really appears, it can be considered safe. Figure 4.2 shows the relationship between system reliability and safety. From the perspective of reliability, a system can be divided into the reliability of safety-related parts and the reliability of non-safety-related parts. From the perspective of safety, a system can be divided into components that support a fail-safe state and components that do not. The intersection area between reliability and safety is the system components that rely on reliability to provide safety.

4.2.3 Evolution of Functional Safety

Safety is the most important and basic need of human beings. With the continuous penetration and accelerated application of E/E/PE technologies in human production and life, from the early industrial process control and nuclear power control to the present railway, automobile, robot, medical care, household appliances, and other fields, people’s constant concern for safety requires that these E/E/PE technologies be standardized and restrained, which has become an inevitable requirement for the development of functional safety. Unfortunately, the IT field has long pursued the richness of functions and the extreme performance and paid little or no attention to functional safety. This may be due to the lack of hardware resources, or the lack of knowledge and understanding of the common ESS problems in software/hardware products, or the lack of correct theoretical guidance and effective response strategies.

With IEC 61508 as the basic standard for functional safety, various safety-related industries have launched their own industry-specific functional safety standards, including IEC 61511 “Functional Safety—Safety Instrumented Systems for the Process Industry Sector”; IEC 61513 “Nuclear Power Plants—Instrumentation and Control for Systems Important to Safety—General Requirements for Systems”; EN50128 “Railway Applications—Communication, Signaling and Processing Systems—Software for Railway Control and Protection Systems”; ISO26262 “Road Vehicles—Functional Safety”; IEC 62304 “Medical Device Software—Software Life Cycle Processes”; ISO10218-1 “Robots for Industrial Environments—Safety Requirements—Part 1: Robot”; ISO22201 “Programmable Electronic Systems in Safety-related Applications for Lifts”; etc. As shown in Fig. 4.3, the overall development trend of various safety-related industries makes functional safety a

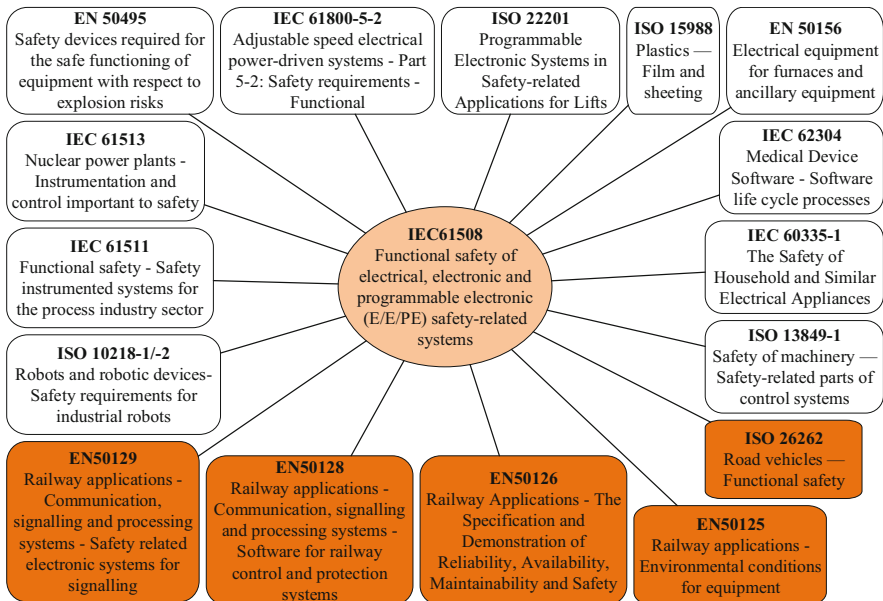


Fig. 4.3 Functional safety standards in different industries

common requirement, and developing industry-specific standards has become a trend.

After a series of functional safety standards were issued, they have been gradually brought into the category of safety laws and regulations by the relevant authorities in many countries and have greatly promoted the compliance of functional safety standards in the whole process of design, installation, operation, maintenance, and decommissioning of safety-related systems in various industries, as well as the development of certification, service, training, and engineering services of relevant functional safety standards. Currently, international certification agencies such as Germany's TUV organization, the United States' FM Global, and the United Kingdom's Sira Certification Services provide functional safety certification services, including product certification, factory certification, equipment installation certification, qualification certification, etc.

China also attaches great importance to the formulation and promotion of functional safety standards. It transformed IEC61508 into Chinese national standard GB/T20438 in 2006 and IEC61511 into Chinese national standard GB/T21109 in 2007. Various industry-specific functional safety standards have been published in China, such as GB/T16855 "Safety Related Parts of Machinery Safety Control System," GB/T34590 "Road Vehicles—Functional Safety," GB/T35850 "Programmable Electronic Systems in Safety-related Applications for Lifts (Elevators), Escalators and Moving Walks," etc., forming the system architecture of functional safety standards in China.

4.3 Basic Connotation of Functional Safety

Functional safety is a systematic, complete, and complex system. It ensures the effective performance of safety functions through comprehensive application of advanced technologies and management measures such as risk assessment, determination of safety integrity level, design, development, and functional safety management in the whole life cycle, so as to control risks within an acceptable range. A classic interpretation of functional safety is "Functional safety is a part of overall safety, which depends on whether the system or equipment can be operated correctly according to its input. When each specific safety function is realized and the necessary performance level of each safety function is met, the safety goal is achieved" [10]. In other words, the functional safety of a device or system should be correctly and properly guaranteed in case of random failure, system failure, or common cause failure.

4.3.1 Risk-Based Safety

In theory, there is no absolute safety. Therefore, functional safety advocates a new safety concept—an approach based on risk management. The existence of risks is

believed to be absolute, and as long as a certain risk is within the tolerable range in the current stage of social development, it can be considered safe. If the risk is intolerable, it is unsafe, and corresponding measures must be taken to reduce or eliminate it. Therefore, in a certain sense, the essence of functional safety is to control risks.

The general concept of risk reduction in IEC 61508, the basic standard for functional safety, can be illustrated in Fig. 4.4. When using a safety-related system to achieve the safety goal, the first step is to determine the risk of the object of protection, namely, the equipment under control (EUC) mentioned in the standard, find out the possible dangerous points in the interaction between the internal and external environment of EUC, and calculate or evaluate the risk for each dangerous point. The second step is to determine the allowable risks based on laws, regulations, contracts, or user requirements. The third step is, according to the difference between the two, to achieve the necessary risk reduction, that is, the target value of risk reduction.

According to functional safety requirements, if the EUC risk is greater than the allowable risk, measures such as E/E/PE safety-related systems, other technical safety-related systems, external risk reduction facilities, etc. must be used to reduce the risk below the allowable risk. These measures must comply with the safety requirement specifications during implementation. The specifications include all the requirements that safety-related systems perform safety functions. One is the specification of safety function requirements, which is about the function needed to achieve safety, and the other is the specification of safety integrity requirements, which is about the probability that this function can be realized correctly. In this way, the assessment of safety issues is simplified to the assessment of risks.

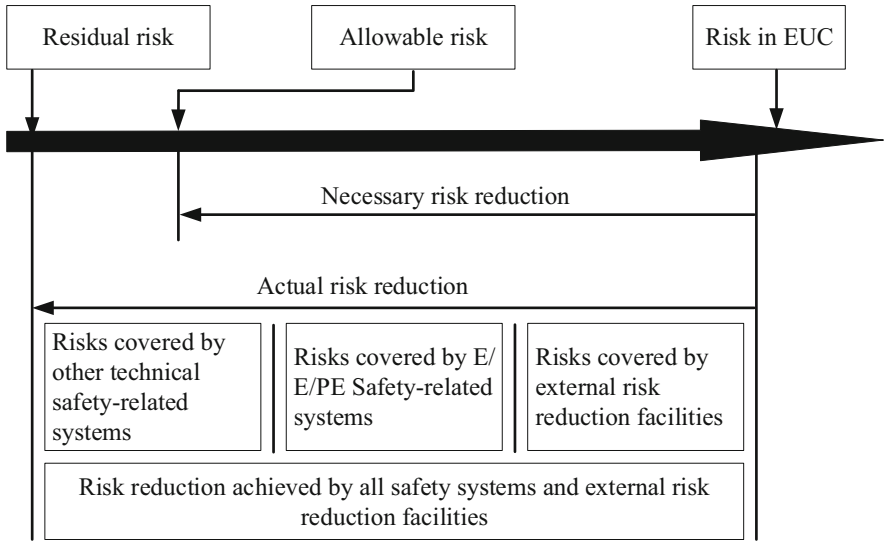


Fig. 4.4 Reduction of functional safety risks

4.3.2 Levels of Safety Integrity

IEC 61508 defines safety integrity as the possibility that safety-related systems can successfully complete the required safety functions under specified conditions and within specified time. The safety integrity levels (SIL) are a classification of safety integrity. IEC 61508 specifies four safety integrity levels, in which SIL 1 is the lowest and SIL 4 is the highest.

When determining safety integrity, all the causes for failures leading to unsafe states should be considered. Among the causes for failures, there are both random hardware failures that can be quantified and system failures that, in most cases, cannot be accurately quantified, such as uncertain software failures. Precisely because of the impossibility of accurate quantification in many cases, IEC61508 uses a discrete hierarchical method to describe safety integrity.

IEC 61508 divides the operation of safety-related systems into two modes and specifies the corresponding target failure amount of SIL for each of the two modes. As shown in Table 4.1, one is a low-demand operation mode, in which the safety-related system should not be activated more than once a year. In this case, SIL is expressed by the probability of failure on demand (PFD) whenever the safety-related system is activated. For example, if the airbag’s control system adopts the low-requirement operation mode shown in Table 4.1, the airbag’s safety-related system is required to function at most once a year, and the average failure probability each time the safety-related system functions does not exceed 10^{-4} , which is equivalent to the possibility of airbag failure at most once every 10,000 years. The other is a high-demand operation mode or a continuous operation mode, which requires safety-related systems to function more frequently than once a year. In this case, SIL is expressed by the probability of failure per hour (PFH). As shown in Table 4.1, if a communication system claims to be as safe as SIL 3, then the probability that it cannot communicate correctly and cause danger is $[10^{-8}, 10^{-7}]$, which is equivalent to an accident at most once every thousand years.

Unlike hardware, software does not have an aging process or a random failure pattern, but software failures may cause system failure. Therefore, unlike the random failure of hardware that can be characterized by probability, the SIL of software cannot be quantitatively characterized by PFD or PFH. However, it is also necessary to clarify the safety function and SIL requirements of software in design, integration, use, and maintenance. IEC 61508 has formulated a complete set of development

Table 4.1 Safety integrity levels (SIL)

Safety integrity levels	High-demand operation modes (probability of dangerous failure per hour)	Low-demand operation modes (probability of failure when required)
4	$[10^{-9}, 10^{-8})$	$[10^{-5}, 10^{-4})$
3	$[10^{-8}, 10^{-7})$	$[10^{-4}, 10^{-3})$
2	$[10^{-7}, 10^{-6})$	$[10^{-3}, 10^{-2})$
1	$[10^{-6}, 10^{-5})$	$[10^{-2}, 10^{-1})$

procedures, including software safety requirement specification, software safety confirmation plan, software design and development, software integration, software operation and modification procedures, software safety confirmation, etc. IEC 61508 has put forward a series of technologies and measures, including the selection of development tools and programming languages, fault detection and diagnosis, software self-monitoring, testing, moderate degradation, formal methods, modular methods, trusted/verified component libraries, computer-aided specification tools, dynamic analysis and testing, etc., in an attempt to achieve fault avoidance, fault detection, fault removal, and fault tolerance through strict quality management and full-life program control and ensure the safety and integrity of software.

4.3.3 Safety Throughout the Life Cycle

A study [11] conducted by the British Health and Safety Executive (HSE) shows that failures of a control system may occur at every stage of its life cycle. As shown in Fig. 4.5, a large proportion of faults are “embedded” in the system before it is put into use. Although the proportion of failures at each stage varies with the complexity of the industry and system, it goes without saying that if functional safety is to be achieved, whole-life-cycle safety must be supported.

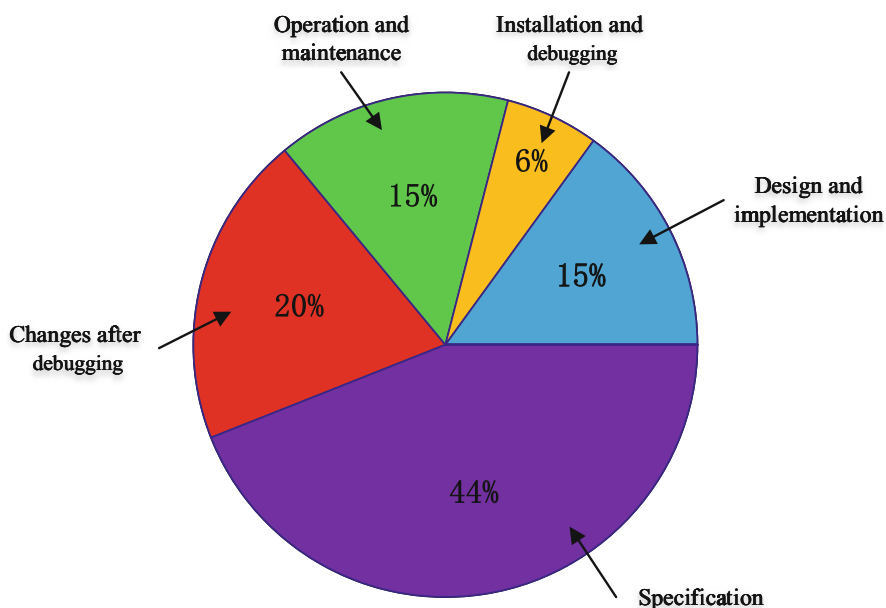


Fig. 4.5 Distribution of main causes of control system failures at each stage of life cycle

The life cycle of functional safety provides a closed-loop model that covers all safety activities from the initial conceptual design, hazard analysis, risk assessment, safety requirements, specifications, design, implementation, operation, maintenance, modification of requirements to decommissioning. It not only focuses on the reliability of the safety-related system itself but also regards the entire life cycle of the safety-related system as a whole, selects optimization plans based on overall needs, and comprehensively evaluates system effects.

The whole life cycle of functional safety specified in IEC 61508 includes the conceptual design in the first stage and the decommissioning or disposal in the 16th stage. Each stage has its own scope, purpose, required inputs, and qualified outputs, and each stage needs to be verified and evaluated to ensure that the activities in that stage are performed correctly so that the outputs meet the requirements of the next stage. As shown in Fig. 4.6, the output of overall safety requirements in the fourth stage is used in fifth stage to determine the allocation of safety requirements. It is precisely because of this structured full-life-cycle analysis and methodological guidance that the safety function requirement specification and safety integrity requirement specification are implemented step by step and maintain good consistency throughout the safety life cycle.

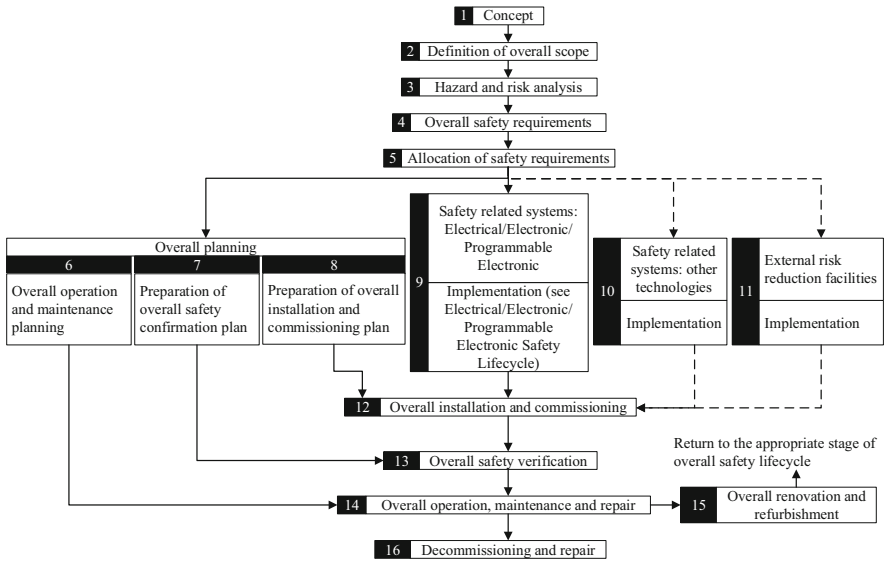


Fig. 4.6 Life cycle model of IEC 61508

4.3.4 Safety Under Systematic Management

In addition to technical factors, many system failures are also caused by latent or active (but not deliberate) human factors. Traditional risk analysis assumes that accidents are caused by natural failure of components and oversimplifies the influence of human factors. In fact, it is wise to consider human factors when identifying and analyzing hazards. Functional safety is closely related to people's safety knowledge, safety skills, safety awareness, and safety attitudes at all stages of the whole life cycle. Errors or functional failures in any link may cause accidents.

Functional safety management is an important factor in achieving functional safety. In the whole life cycle of functional safety, it is necessary to bring all individuals and organizations involved in analysis, evaluation, design, verification, confirmation, audit, configuration, operation, maintenance, and other activities into the scope of systematic management; clarify their respective responsibilities, requirements, and procedures; and establish a sound functional safety management system. Therefore, compared with the 2000 version, the 2010 version of IEC 61508 adjusted the functional safety management requirements. The adjustment requirements are that all personnel involved in safety-related systems should be identified and should be competent for their respective jobs, and their responsibilities for activities at all stages of the life cycle should be clearly defined.

Functional safety needs systematic management in the whole life cycle of products. The whole life cycle includes planning, concept, design, verification, production, operation, and scrapping. It is process-driven and includes requirement determination, design, implementation, integration, test verification, confirmation, and other links, with corresponding technical requirements in each link. It covers all electronic and electrical aspects, including CPU, FPGA, ASIC, and other hardware, basic software such as various operating systems, a dazzling array of application software, and related supporting software tools. In these life cycle stages, its systematic management process includes project management, safety management, demand management, configuration management, change management, production management, and after-sales service management. Moreover, most of the time, adjusting any part of it may have an overall impact, and any local changes must be considered from a holistic perspective, which often requires lengthy regression testing and verification.

4.4 Overview of Functional Safety Technologies

Functional safety technologies generally include the discipline-specific functional safety technology (including hazard analysis and risk assessment technology, etc.), functional safety development technology, and functional safety testing technology. The core goal of these technologies is to reduce the risks caused by random hardware failures and system failures below the tolerable level.

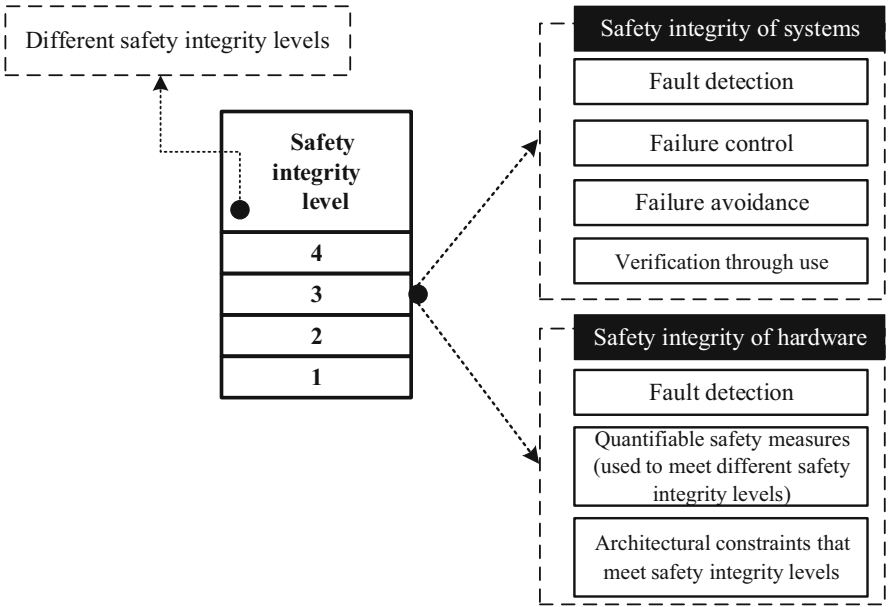


Fig. 4.7 Technologies and measures to achieve SIL

4.4.1 Safety Integrity Technology

In response to random hardware failures and system failures, the fundamental goal of functional safety activities is to comprehensively adopt corresponding hardware safety integrity technologies and measures and system safety integrity technologies and measures so that safety-related systems can achieve the required safety integrity level. Figure 4.7 shows the main technologies and measures from the perspectives of hardware safety integrity and system safety integrity. The analysis is carried out below.

4.5 Hardware Safety Integrity Technology

As shown in Fig. 4.7, hardware safety integrity can be achieved through improved fault detection, quantifiable safety measures, and architectural constraints that meet specified SILs.

- (1) **Fault detection:** Fault detection is a measure that uses other components to add safety functions. It can monitor abnormal expressions of main functions or inform relevant technical links before harm occurs to prevent or reduce losses caused by faults. Some technologies and measures for fault detection are shown in Table 4.2.
- (2) **Quantifiable safety measures:** These measures need to estimate the probability of safety function failure caused by random failure according to the structure of

Table 4.2 Structural constraints of Class B safety-related subsystems

Safe failure fraction (SFF)	Hardware failure margin		
	0	1	2
<60%	Not allowed	SIL 1	SIL 2
60% ~ 90%	SIL 1	SIL 2	SIL 3
90% ~ 99%	SIL 2	SIL 3	SIL 4
≥99%	SIL 3	SIL 4	SIL 4

Note 1: The hardware fault margin N means that $N + 1$ faults will result in the loss of the safety function.

Note 2: Safe failure fraction (SFF) refers to the ratio of the average safe failure rate of the subsystem plus the ratio of the detected average dangerous failure rate to the total average failure rate of the system.

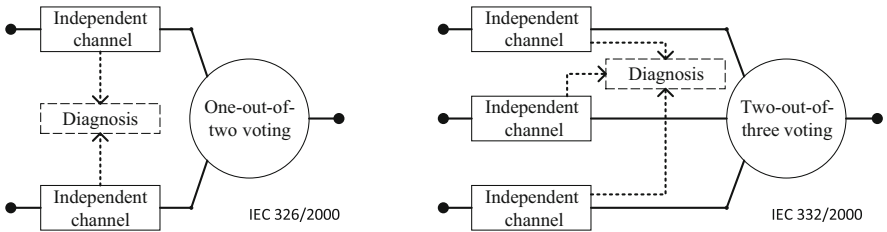


Fig. 4.8 Voting structure

E/E/PE (such as redundant voting) and failure modes of different components. This probability should be equal to or lower than the target failure amount specified in the safety specification. Depending on the specific situation, modeling methods that can be used include reliability block diagram, causality diagram analysis, fault tree analysis, Markov model, and so on.

(3) Hardware architectural constraints: This technique generally means that safety integrity is limited by hardware failure margin and safe failure fraction. Table 4.2 shows the architectural constraints of Class B safety-related subsystems. Typical Class B subsystems include microprocessor-based devices or devices with complex custom logic. As shown in Table 4.2, the corresponding hardware failure margin can be selected according to the SIL requirements and safe failure fraction. For example, when safe failure fraction (SFF) is 90% ~ 99%, if the safety integrity level is required to be 3, then the hardware failure margin is 1. In this case, the redundant voting system can choose one out of two or two out of three, as shown in Fig. 4.8.

4.6 System Safety Integrity Technology

Compared with random hardware failures, avoidance of system failures may require changing system design. To minimize system failures, it is necessary to follow a strict development process and independently review all kinds of work and products. This process is usually represented by V models with different complexity, as shown in Fig. 4.9. Technical and management measures to deal with system failures usually

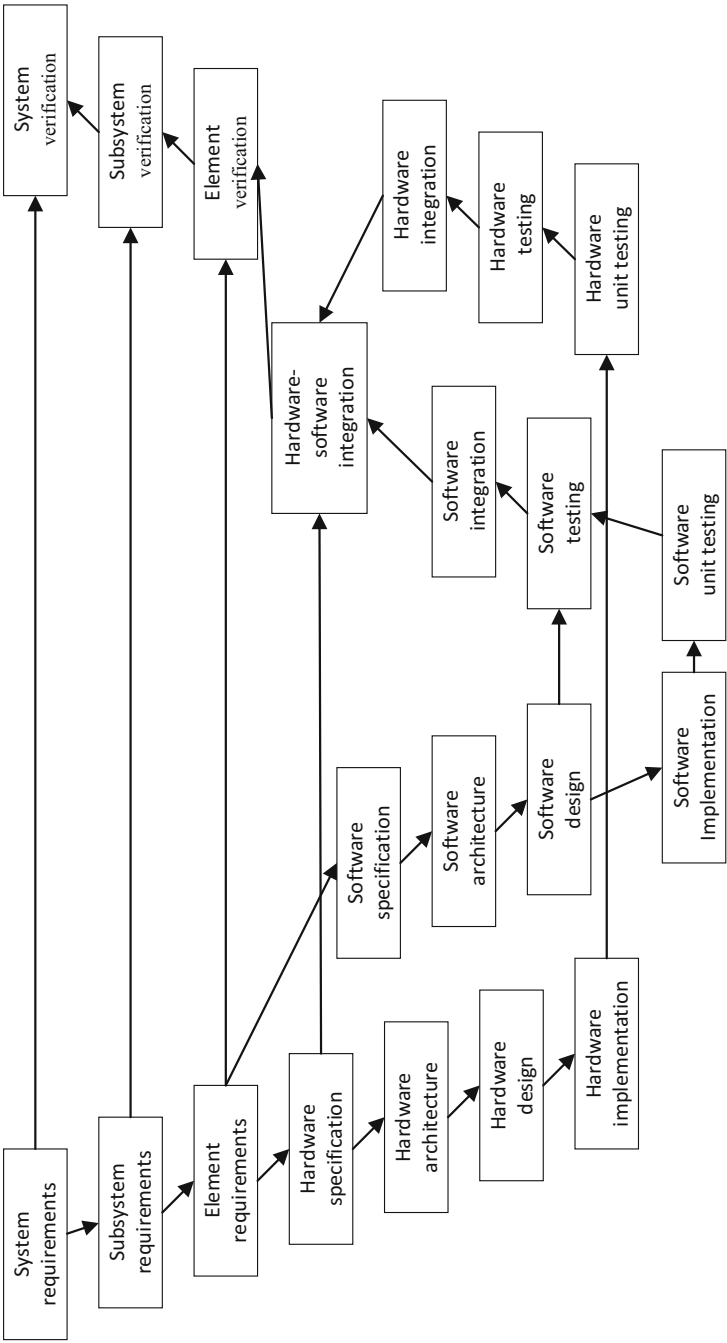


Fig. 4.9 V model development process

Table 4.3 Technologies and measures for fault detection, failure control, and failure avoidance

Category		Technologies and measures
Fault detection		Online detection, multi-channel comparison, majority voting machine, self-starting verification, standard testing of access ports and boundary scan structures, watchdog timers, transmission redundancy and information redundancy detection of information transmission errors, fault detection and diagnosis, error checking, performance trend analysis, fault tree, state diagram, etc.
Failure control	Software/hardware failure control	Sequential or logical monitoring of program sequence, standard testing of redundant hardware, access port and boundary scan structure, code protection, diversified hardware, diversified programming, failure assertion programming, error checking and correction, code protection
	Environmental impact control	Isolation of power lines and information lines, measures against physical environment (temperature, humidity, vibration, dust, corrosion), redundant hardware
	Operation process failure control	Modification of protection technology, online monitoring and detection of failure, input confirmation, etc.
Failure avoidance	Measures to avoid mistakes in safety specification	Standardized project management, strict document management requirements, standardized inspections, computer-aided specification tools, formal methods, etc.
	Failure avoidance measures in design and development	Project management, documentation, use of well-proven components, structured and modular design, semi-formal methods, checklists, simulation, etc.
	Failure avoidance measures during integration	Project management, documentation, testing technology, black-box testing, statistical testing, etc.
	Failure avoidance measures in operation and maintenance	Clear management requirements for operators, prevention of operators' errors, setting of operation privileges, etc.
	Failure avoidance measures in safety confirmation	Static analysis, dynamic analysis and failure analysis, simulation and failure analysis, etc.; black-box testing, surge immunity testing, fault insertion testing; static testing, field experience, worst-case testing, etc.

include fault detection, failure avoidance, failure control, verification through use, etc. The specific measures are shown in Table 4.3. These technologies and measures are usually qualitative rather than quantitative. The higher the SIL requirements, the stricter, more reliable, and more credible these technologies and measures should be.

1. Fault Detection

At present, great progress has been made in the research and practice of hardware fault detection, but there is a lack of systematic and effective methods for fault detection of complex systems containing software. Software fault detection generally obtains various kinds of information in the software through static inspection, dynamic analysis, and other methods, analyzes and obtains various characteristics of possible software faults, and identifies whether the software is running normally or has faults. After a dangerous fault is detected, a specified action should be taken to achieve or maintain the safe state of the system.

2. Failure Avoidance

In order to avoid failure, a set of appropriate management measures should be used in safety specification, design, development, integration, operation, maintenance, and safety confirmation to prevent the introduction of faults in the whole life cycle. In safety specifications, measures such as requirement management, project management, document management, and review management can be adopted to avoid failure. In design and development, system complexity can be controlled through modularization, standardization, etc.; and accurate definitions can be made through relevant functions, interfaces, concurrency, synchronization, and other relationships, and failures can be avoided through standardized document management, strict verification, confirmation, and other measures. In the process of integration, failures can be avoided by using a variety of testing methods. In operation and maintenance, failures can be avoided through measures such as operation management and privilege management. In safety confirmation, failures can be avoided through measures such as static analysis, dynamic analysis, and failure analysis.

3. Failure Control

Safety-related systems are required to tolerate software/hardware design faults and residual faults; to cope with environmental stresses, including electromagnetic interference, ambient temperature and humidity, heavy ion radiation intensity, mechanical vibration, etc.; and to cope with operators' errors, etc., as shown in Table 4.3. It should be pointed out that with the continuous expansion of software in various safety-related systems, the failure control of software becomes more and more important. Software failures are usually caused by human design defects. Since different people's experience, knowledge, skills, and understanding of system applications vary greatly, software failures often cannot be accurately described by probability. Because of this, software failure control is extremely challenging and can be implemented from two aspects: management and technologies.

From the perspective of management, software failure control needs to use a package of management norms and measures to enhance system safety in such links as preparation of software safety specifications and software safety confirmation plans; software design, development, and implementation; hardware-software integration; software operation and modification procedures; and software safety confirmation, so as to ensure that risks are eliminated or controlled within an acceptable level of safety integrity.

From the technological perspective, the software fault-tolerant technology is a relatively effective means of controlling the impact of software failure. Fault tolerance refers to the ability of software to ensure the provision of services by shielding or correcting faults when they occur. One approach to fault tolerance is to rely on the redundancy technology. Software fault tolerance is mainly divided into two categories: software redundancy and temporal redundancy. Software redundancy refers to the use of multiple components or modules with the same function, so that even if one component or module fails, the whole system can still complete the specified tasks. Multi-version programming (NVP) is a kind of static shielding redundancy, which makes use of the diversity of different programs that accomplish the same function for complementary fault tolerance. It is a common software redundancy method that can reduce common-cause failures. Temporal redundancy is based on the idea of retry-on-failure. By setting a checkpoint and rollback mechanism in the software, it can be rolled back to the appropriate checkpoint for re-execution when a failure occurs.

4. Verification Through Use

Where there is sufficient evidence that a previously developed subsystem is safe, this can be considered evidence of proven use. This requires that previously developed subsystems have clear functional specifications, that there is sufficient evidence to prove that the specific configuration of the subsystem has actually been applied before, and that all required analysis and testing have been taken into account. In addition, in order to achieve the required safety integrity level, the evidence is required to show that the subsystem's previous use conditions are the same or similar to those that will be faced in the E/E/PE safety-related systems.

The engineering technology is the core support for functional safety. The urgency of technological innovation in functional safety can be seen from the changes of functional safety research hotspots. Literature [12] selected the core database of Web of Science as the data source, took the functional safety research papers published between 1986 and 2019 as the research object, and drew a knowledge map of the evolution path of functional safety research hotspots. It can be found from the map that in addition to following the life cycle management process of functional safety including safety strategy, risk assessment, and risk management, it is also necessary to rely on various technical measures to support the achievement of safety integrity goals. It can also be found from the map that detection and testing technologies represented by the automatic test vector generation (ATPG) and built-in self-test and architecture technologies represented by redundancy and diversity are still the most commonly used technical means for functional safety. This, to some extent, reflects the urgent need to make major innovations and breakthroughs in technical measures for functional safety.

4.6.1 Random Hardware Failures and System Failures

An important feature of functional safety is to consider both random hardware failures that can be quantified and system failures that cannot be quantified, leading to a semi-quantitative method: SIL (safety integrity level).

Random hardware failure may occur due to degradation of one or several functions during the life cycle of a hardware component. In various hardware components, there is functional degradation, such as aging, caused by physical factors and occurring at different rates, as a result of which, the equipment containing the relevant component will fail at a predictable rate but at an unpredictable time (or randomly). Because random hardware failures follow the law of probability distribution, they can be analyzed quantitatively.

System failures are related to a certain cause and occur in a definite way. Only by modifying design or manufacturing process, operating procedures, documents, or other relevant factors can such failures be ruled out. Errors and omissions in safety requirements and specifications, errors in hardware design and implementation, installation operations, and defects in software design and implementation can all cause system failures. Therefore, system failures cannot be described by probability, but the empowerment of the DHR architecture will change this situation to a considerable extent.

4.7 Development Trend of Functional Safety

With its principle and method being widely accepted internationally, functional safety has developed rapidly, its influence has been expanded, and related theories, methods, technologies, and management have been optimized.

4.7.1 Industry-Specific Functional Safety

IEC 61508 is the basic standard for functional safety. The first edition was released in 2000, and the second edition was released in 2010 after revision. In 2019, the international maintenance working group of IEC 61508 decided to revise the second edition and prepare the third edition. IEC 61508 can be used as the basis for formulating functional safety standards in automobiles, robots, medical care, nuclear power, and other industries. It can also be directly applied where there are no industry-specific functional safety standards.

Taking the automobile industry as an example, IEC 61508 had been used in the development of automobile functional safety until 2011 when the first edition of ISO 26262, Road Vehicles—Functional Safety, was released [13]. Then the second edition of ISO 26262 was released at the end of 2018.

Comparing the first edition of ISO 26262 and the first edition of IEC 61508, we see that both standards have the same goal, that is, ensuring personal and environmental safety, and the basic theories, methods, processes, management, and technologies are similar in architecture. However, IEC 61508 is more suitable for small-scale systems, while ISO 26262 is obviously more suitable for complex automobiles. IEC 61508 does not specify a specific method for hazard analysis, while ISO 26262 specifies a risk map. IEC 61508 adopts SIL that ranges from SIL 1 to SIL 4, while ISO 26262 adopts ASIL that comprises ASIL A, ASIL B, ASIL C, and ASIL D. In many cases, ASIL D is similar to SIL 3, partly because the maximum number of casualties in a car accident is generally less than six, and SIL 4 is not required, while the latter is usually used in railways, process control, and nuclear industry, where an accident may cause 10, 100, or even 1000 casualties. Road vehicles in ISO 26262 have a lower duty cycle than industrial systems in IEC 61508. For automobiles, a typical case is that the system is powered on for an hour and then powered off for a few hours, while industrial systems usually work 24 hours a day. Among the reliability enhancement measures, the tens of milliseconds of diagnosis after the automobile is started is very important for safety improvement, although the driver usually doesn't care about this diagnosis. One of the safety challenges of automobiles is mobility, that is, all sorts of unexpected external risks during driving.

Of course, ISO 26262 has many more specific refinements and customizations than IEC 61508, which are not to be listed here. Currently, many other safety-related industries have introduced functional safety standards that are more in line with their respective technologies, management, and production modes.

4.7.2 Complexity of Safety-Related Systems

With the advancement of technologies and the expansion of safety functions, the development of safety-related systems is becoming increasingly complex, which can be seen from the development environment and development tools. For example, tools and environments such as Eclipse, Green Hills AdaMULTI, Visual studio, Simulink and DOORS, PDM, MS Project, LDRA, Workflow, and Parasoft are used in the aerospace field, while Eclipse, Jenkins, Maven, OpenCV, ROS and DOORS, MS Project, MATLAB, and so on are used in the automotive field. In terms of programming language, software/hardware programming languages such as C/C++/ADA/JAVA/Verilog are used in the aerospace field, while languages such as C/C++/JavaScript/JAVA/Python/Fortran/JOVIAL/Verilog are used in the automotive field. From the perspective of operating systems, Linux/Windows and VX Works are used in the nuclear industry, Linux and QNX are used in the automotive field, and Linux/Windows and QNX are used in the aerospace field.

Machine learning technologies represented by deep learning have developed rapidly and are increasingly being used in safety-related systems such as autonomous driving and industrial safety monitoring. It plays a key role in image classification, target detection, speech recognition, and other aspects. Following

programmable logic controllers, various embedded processors and CPUs, GPUs, FPGAs, and ASICs began to be used in safety-related systems on a large scale. In many cases, multi-core processor platforms carry both safety functions and non-safety functions. At the same time, with the continuous increase and enhancement of functions, the size of systems' software code has also expanded rapidly.

In short, heterogeneous hardware platforms, multiple operating systems, diverse programming languages, various development environments, and tool chains, as well as related data, algorithms, models, frameworks, codes, and communications, are constantly integrated, making the complexity of safety-related systems increase dramatically.

4.7.3 Deepening of the Unity of Opposites

Functional safety is a systematic and complex complete system, and its development process contains the unity of opposites in many aspects. Various safety-related industries obtain a benign dynamic balance or reconciliation by constantly resolving various contradictions and promote the continuous development of functional safety.

1. The Unity of Certainty and Flexibility

Certainty in functional safety practice includes the standardization of safety function requirements, the clarity of safety integrity levels, the orderliness of full life cycle safety design, and the completeness of functional safety management.

The essence of functional safety practices also includes flexibility. For example, for a certain subsystem, there may be multiple solutions to achieve the required safety integrity level. Each solution may use a different architecture (such as 1oo2 or 2oo3 redundant voting) and different software/hardware implementations. There is no right or wrong between different solutions, the key is to see which solution is safer, more economical, more maintainable, and easier to implement.

2. The Unity of Integration and Balance

In safety-related industries represented by the process industry, safety functions are usually accomplished by a special protection system, and the protection system is often physically separated from the conventional control system. Therefore, the development of each system is often completely different from that of the other. For emerging industries such as robots and intelligent networked automobiles, safety functions and conventional control functions usually share a technical architecture or even the same physical carrier. Functional safety must be integrated with product development, including the integration of safety requirements and product requirements, as well as the integration of architecture, process, design, testing, and maintenance. Functional safety must be integrated with product research and development, including the integration of safety requirements and product requirements, as well as the integration of architecture, process, design, testing and maintenance.

Balance is a topic that must be considered in the integration of safety and non-safety functions. Functional safety is only one attribute of a product. Product attributes also include usability, reliability, maintainability, testability, scalability, and reusability, all of which need to be considered in product development. The purpose of introducing functional safety into the product development process is to reduce the safety risk of the product to people and the environment. However, unrealistic pursuit for safety may result in a serious imbalance between safety and other attributes of the product. For example, if a car is designed to reach a high level of functional safety but cannot be mass-produced, then it is commercially infeasible, let alone put it into use. Therefore, products that cannot be put on the market or used are meaningless, even if they are absolutely safe.

3. The Unity of Cost and Safety

Functional safety is a full-life-cycle safety activity, which means that it will definitely increase costs, including development costs of safety-related systems, hardware costs, certification costs, testing costs, tool costs, management costs, labor costs, etc. However, reducing costs is of great importance to enterprises.

Product safety is one of the core concerns of users. Under the same or similar product quality and price, the sales of products with stronger safety are usually greater. The greater the sales volume, the lower the amortized costs are. The lower the cost, the more advantages the product has in marketing, which leads to even more sales. This is a benign process of mutual promotion. However, if the implementation cost of functional safety far exceeds the reasonable range, then safety, cost, and product sales will be in vicious interaction that hinders each other.

It should be noted that with the advancement of electronic information technologies, the cost of functional safety implementation under the same safety function conditions will gradually decrease. Therefore, in general, product safety in safety-related industries will continue to improve.

4. The Unity of Development Cycle and Safety

Safety is one of the fundamental attributes or functions of a product. In industries such as automobiles, safety is its primary attribute. Currently, market competition is fierce. Quickly delivering new products with certain fault tolerance and pressure-bearing capabilities and seizing the market, timely discovery of product problems, and quick repairs are one of the core competitiveness of products. The reality is that the product development cycle is so short that even the delivery pressure of regular functions is very great. How to ensure the development cycle after adding functional safety? This is a common problem faced by many industries.

It is one of the important directions for the future development of functional safety to keep up with the product development cycle and achieve rapid delivery. Agile development may become an important requirement in functional safety development. Besides, in the development of safety-related systems, enabling the system to have endogenous or inherent GFS so that the system can self-discover

and self-repair problems (i.e., has cyber resilience) can not only speed up the delivery of products but also provide valuable opportunities and more time for iterative updating of products.

4.7.4 Resilience of Safety-Related Systems

As intelligence, networking, and other related technologies continue to be introduced into safety-related systems, data, models, algorithms, software/hardware, communications, and various sensors are integrated into the systems, and it is almost impossible to find definitive answers to all safety boundaries of complex systems.

The uncertainties faced by functional safety-related systems, including the uncertainty of perceived boundaries, are multifaceted. The physical boundary of perception capabilities after multi-sensor fusion is not an absolutely clear result, and the accurate boundary of identification, classification, and detection of multiple AI models is not an absolute value. In addition, system failure is uncertain. System failures often originate from design defects. For complex systems, software/hardware design defects are inevitable. However, with the current level of human technologies, it is almost impossible to thoroughly investigate all software/hardware ESS problems. In addition, the application of AI technologies in safety-related systems also contains many uncertainties, including the uncertainty caused by the deviation between training data and actual data, the uncertainty in the model, etc. Finally, functional safety problems caused by cyber-attacks obviously also contain more and greater uncertainties.

In addressing these uncertainties, both designers and regulatory authorities are seriously inexperienced and are faced with the dilemma of “you don’t know what you don’t know.” For example, some data noise is enough to make a huge difference in the recognition results of artificial intelligence algorithms, which may bring safety risks to AI systems that are closely related to personal safety, such as nursing robots and autonomous driving. It is difficult to find the answer to similar problems in today’s definition of functional safety.

A positive prospect for dealing with uncertainty is the adoption of new concepts and methods in the field of resilience engineering. One of the characteristics of this approach is its emphasis on complexity as a source of all kinds of accidents, aiming to build, design, develop, implement, maintain, and sustain the trustworthiness of the system so that it has the ability to predict, withstand, recover from, and adapt to adverse conditions, pressures, and attacks [14]. The technologies often applied in resilience engineering include analytical monitoring, dynamic positioning, privilege restriction, diversification, redundancy, reconstruction, etc.

It is an important advancement to upgrade the concept of robustness of traditional safety-related systems into the concept of cyber resilience. Although both concepts share the goal of resisting stress, interference, or disruption, the concept of cyber resilience is more committed to “being ready to accept accidents” than “being ready to avoid accidents.” This is more proactive as the complexity of safety-related

systems increases, problems caused by various uncertainties continue to emerge, and the safety risks faced by the system become more multi-dimensional.

Grasping the meaning of the concept of cyber resilience is also beneficial to actively dealing with the intertwining of CPS functional safety and cyber security problems. This is because cyber resilience can not only effectively deal with random errors and uncertain faults but also reliably cope with man-made explicit or implicit cyber-attacks.

However, cyber resilience is sometimes regarded as a trendy but empty term. This may be because of its ambiguity or lack of integrated theoretical foundation and technical architecture. From this perspective, therefore, cyber resilience engineering requires fundamental creation and innovation at the levels of the theoretical framework, methodology, and practical specifications, in order to better support the implementation of this advanced concept and empower the new generation of cyber-physical systems with GFS.

4.8 The Basic Concept of Cyber Resilience

As today's social activities increasingly rely on complex and interconnected network systems and digital infrastructure, cybersecurity threats are becoming more diverse and larger in scale, causing more and more serious losses. Due to the unpredictability, high degree of uncertainty, and rapidly evolving nature of potential security threats, people have gradually realized that it is unrealistic to ensure absolute security or "no breach of defense" in cyberspace. In the face of increasingly complex cyber-attacks, the focus of cyber security should shift from preventing network accidents to mitigating the harm caused by them. It should shift from merely resisting attacks to ensuring service continuity and availability, building rapid recovery capabilities, and maintaining normal operations. It is in this context that the concept of cyber resilience comes into being.

4.8.1 *The Origin of the Concept of Cyber Resilience*

Resilience is originally a concept in physics, which refers to the property of an object that deforms under the action of an external force and can return to its original size and shape when the external force is removed. In 1973, Holling introduced the concept of resilience into ecosystem research for the first time in his seminal paper *Resilience and Stability of Ecological Systems* [15], which attracted widespread attention in the social engineering field. Subsequently, the term resilience began to be widely used in more fields, such as ecology, metallurgy, manufacturing, individual and organizational psychology, supply chain management, strategic management, and safety engineering [16]. Although the meaning of resilience may differ somewhat in different application fields, the concept of resilience in all these fields is

closely related to the ability of an element to return to a stable state after being disrupted.

In recent years, as cyber security threats have become increasingly severe, the concept of resilience has been gradually introduced in the fields of systems engineering, information technology, and computer networks. In 2009, Madni et al. proposed in *Toward a Conceptual Framework of Resilience Engineering* [17] that resilience engineering is “to build systems that can predict and avoid accidents, survive disruptions through appropriate learning and adaptation, and return as soon as possible to the state before the interruption.” In 2010, Sterbenz et al. put forward the concept of network resilience in their paper *Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines* [18], in which network resilience is defined as “the ability of a communication network to provide and maintain an acceptable service level in the face of various failures and challenges that jeopardize normal operations.” The paper holds that communication network resilience involves many concepts, as shown in Fig. 4.10. The left part of the figure is the engineering design of the challenge tolerance and handling system, which enables the system to continue to provide services in the face of challenges; the right part is trustworthiness, which describes the measurable characteristics of system resilience. What connects the two is robustness and complexity, which is formally the performance of the control system when it is disturbed or the trustworthiness of the system when it is challenged.

In 2010, Massachusetts Institute of Technology Research and Engineering (MITRE) in the United States published a paper *Building Secure, Resilient Architectures for Cyber Mission Assurance* [19], which is considered a pioneering work in

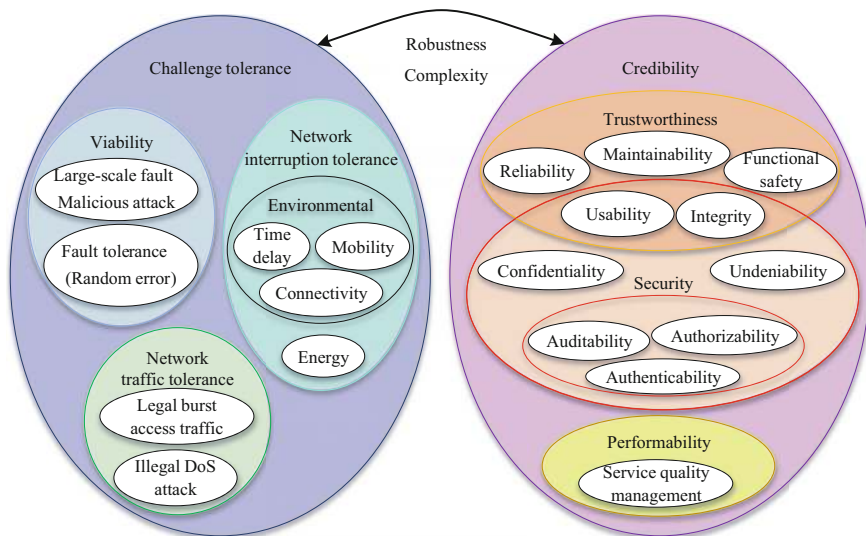


Fig. 4.10 Schematic diagram of concepts related to communication network resilience

the field of cyber resilience. The paper holds that the idea of achieving 100% protection in the field of cyber security is not only unrealistic but also leads to a false sense of security. The paper proposes that more attention should be paid to protecting the continuity of mission-critical functions, and compensatory measures should be considered in the event of protection failure to ensure that mission-critical functions can still be achieved in the event of an attack. To this end, the paper explores the idea of cyber resilience architecture, including five goals: protection/deterrence, detection/monitoring, containment isolation, maintenance/recovery, and adaptive evolution. On this basis, the paper discusses some key technologies that help to achieve these resilience goals: variety, redundancy, integrity, isolation/segmentation/containment, detection/monitoring, minimum privilege, non-persistence, distributed and moving target defense, adaptive management and response, randomization and unpredictability, deception, and so on. The paper also holds that cyber resilience architecture and operations should be combined to deal with unknown unknown attacks.

4.8.2 The Connotation of Cyber Resilience

In 2011, MITRE published *Cyber Resilience Engineering Framework* [20], which regards cyber resilience engineering as a combination of resilience engineering, cyber security engineering, and mission assurance engineering, as shown in Fig. 4.11. Cyber resilience engineering introduces the concepts of anticipation, endurance, recovery, and evolution goals from resilience engineering; the idea of resolving threats including advanced, persistent, and changeable threats from cyber

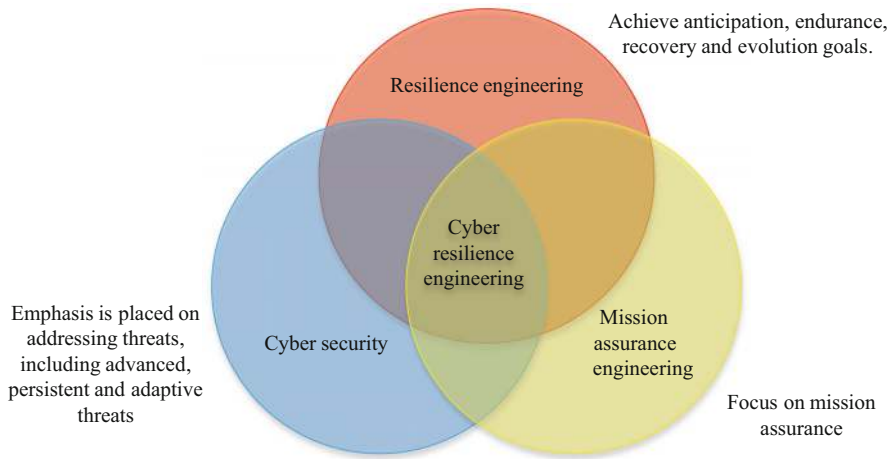


Fig. 4.11 Schematic diagram of the main connotation sources of cyber resilience engineering framework

security engineering; and the concept of mission assurance from mission assurance engineering.

As the core technical research support institution of the National Institute of Standards and Technology (NIST), MITRE carried out a series of in-depth researches on cyber resilience engineering, which supported NIST's official release of the *Developing Cyber-Resilience Systems: A Systems Security Engineering Approach* (NIST SP 800-160V2R1) in December 2021 [21]. This is an important milestone in NIST's adoption of systems engineering methods to build cyber security capabilities, marking the official release of the first authoritative technical document on cyber resilience.

NIST defines cyber resilience as the ability of entities containing network resources to prevent, resist, recover from, and adapt to adverse conditions, stresses, attacks, or damage. Cyber resilience has five main characteristics: ①focusing on the mission or service functions; ②focusing on the impact of APT attacks; ③assuming that the environment is constantly changing; ④assuming that the adversary will definitely breach the system; ⑤assuming that the adversary exists in the system or organization for a long time. Cyber resilience targets both adversarial and non-adversarial threats from cyber and non-cyber sources.

Cyber resilience is closely related to resilience engineering, mission assurance, cyber security protection, service continuity, backup and recovery, etc. It not only focuses on responding to external cyber-attacks but also focuses on its own robustness and reliability. Not limited to preventing or eliminating cyber-attacks, cyber resilience also takes into account the ability to coexist with and recover from such attacks and maintain network availability in case of cyber-attacks. Cyber resilience and cyber security are both related and different. Cyber security is defined as "the ability to keep the network in a stable and reliable operation state and ensure the integrity, confidentiality and availability of network data by taking necessary measures to prevent attacks, intrusions, interference, destruction, accidents and illegal use of the network" [22]. Cyber security aims to protect network resources to the greatest extent from the threat of adverse conditions, while cyber resilience focuses more on ensuring the basic mission functions of network resources under adverse conditions, emphasizing the ability to maintain basic functions even if some network resources are damaged and the ability to recover from and adapt to adverse conditions.

According to MITER [23], the conceptual relationships between cyber security, cyber resilience, traditional security (mainly focusing on confidentiality, integrity, and availability of information), traditional resilience and reliability (mainly focusing on non-adversarial threats), safety security, privacy, and mission guarantee are shown in Fig. 4.12. Cyber security includes traditional security in the consideration of prevention and protection, and cyber resilience overlaps with traditional reliability and resilience in terms of recovery.

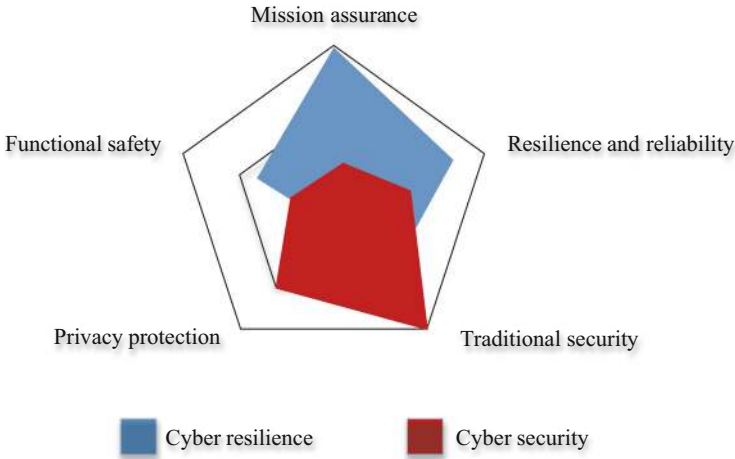


Fig. 4.12 Conceptual relationship between cyber resilience and cyber security

4.9 Framework of Cyber Resilience Engineering

To develop cyber-resilient systems, NIST [21] proposed a systems engineering framework for understanding and applying cyber resilience, including the concept of cyber resilience engineering, the elements, engineering practices, solutions of cyber resilience, etc.

The framework of cyber resilience engineering (Table 4.4) includes 4 top-level goals of cyber resilience, 8 cyber resilience demand objectives (correspondingly broken down into several sub-objectives and demand capabilities), 14 cyber resilience support technologies (correspondingly broken down into 50 typical cyber resilience support methods), 5 cyber resilience strategic principles, and 14 cyber resilience design principles. The top-level goals and demand objectives of cyber resilience determine what attributes and characteristics a cyber-resilient system needs to have. Cyber resilience support technologies and construction principles describe the paths and methods to achieve cyber resilience. Cyber resilience engineering practices are the methods, processes, models, and analysis techniques used to identify and propose solutions. These practices provide a sufficient level of cyber resilience during the system’s life cycle to meet the needs of stakeholders and reduce risks to the organization’s mission or service capabilities in the presence of various threats, including APTs.

4.9.1 The Goal of Cyber Resilience

The cyber resilience engineering framework contains four top-level goals of cyber resilience (Table 4.5), which provide the connections between systems, missions and

Table 4.4 Elements of cyber resilience engineering framework

Composition	System-level definition, goal, and application
Goals	Definition: Supporting (or focus on) one aspect of cyber resilience attributes (anticipation, withstanding, recovery, adaptation)
	Description: Aligning the definition of cyber resilience with definitions of other types of resilience; used to express the concerns, objectives, or priorities of high-level risk stakeholders
Objectives	Definition: Objectives are more specific than goals and more related to threats. They mainly describe the requirements that the system must achieve in its operating environment and throughout its life cycle to meet the needs of risk stakeholders for mission assurance, cyber resilience, and security
	Description: Enabling stakeholders and system engineers to reach a common understanding of cyber resilience problems and priorities; facilitate the definition of metrics or effectiveness of measures
	Application: Used for scoring methods or analysis summaries (e.g., cyber resilience status assessment)
Sub-objectives	Definition: A subsidiary statement of cyber resilience objectives, emphasizing different aspects of the objectives or identifying the ways to achieve the objectives
	Description: Serving as a step in hierarchical refinement of objectives into activities or capabilities that can define performance measures, sub-objectives
	Application: Used in scoring methods or analysis and may be reflected in the system's functional requirements
Activities or capabilities	Definition: A statement of capabilities or actions to support and achieve the achievement of sub-objectives
	Note: Facilitating the definition of the effectiveness of measurement indicators or measures
	Application: Used for scoring methods or analysis; reflected in the system's functional requirements
Strategic principles	Definition: Reflecting the requirements of one aspect of a risk management strategy and providing information for the organization, mission, or system safety engineering practice
	Description: Guiding and informing the engineering analysis and risk analysis throughout the system's life cycle and guiding the use of different cyber resilience design principles, cyber resilience techniques, and implementation methodologies
	Application: Included, quoted, or restated in the system's non-functional requirements
Design principles	Definition: A statement that captures the experience of defining system architecture and design
	Description: Guiding and informing design and implementation decisions throughout the system's life cycle; highlighting different cyber resilience techniques and implementation methods
	Application: Included, quoted, or restated in the system's non-functional requirements, used to guide the use of technology, implementation methods, techniques, and practices in systems engineering

(continued)

Table 4.4 (continued)

Composition	System-level definition, goal, and application
Technology	Definition: A set or category of technologies, processes, or practices that provide the capability to achieve one or more cyber resilience objectives
	Description: Characterizing technologies, practices, products, controls, or requirements in order to understand their contribution to cyber resilience
	Application: Used in engineering analysis to screen technologies, practices, products, controls, solutions, or requirements; used in systems by implementing or integrating technologies, practices, products, or solutions
Method of implementation	Definition: A subset of cyber resilience techniques and processes, defined by how functionality is implemented
	Description: Characterizing technologies, practices, products, controls, or requirements to understand their contribution to cyber resilience and their potential impact on threat events
	Application: Used in engineering analysis to screen technologies, practices, products, controls, solutions, or requirements; used in systems by implementing or integrating technologies, practices, products, or solutions
Solution	Definition: A combination of technologies, architectural decisions, systems engineering, and operational processes, procedures, or practices in the field of cyber resilience
	Description: Providing a sufficient level of cyber resilience to meet the needs of stakeholders and reducing risks to the mission or service capabilities in the presence of advanced persistent threats
	Application: Integrated into a system or its operating environment
Mitigation	Definition: An action or practice that uses a technology, control, or solution to reduce the level of risk associated with a threat event or threat scenario
	Description: Characterizing actions, practices, methods, controls, solutions, or a combination of these in terms of their potential impact on a threat event, threat scenario, or risk
	Application: Integrated into the system when used

service processes, and organization-level risk management decisions. For cyber resilience engineering analysis, cyber resilience goals are the starting point and an inseparable basic connotation for understanding the concept of cyber resilience, none of which is dispensable.

4.9.2 The Objectives of Cyber Resilience

The eight objectives of cyber resilience demand (Table 4.6) are specific descriptions of the system’s realization of the above four goals of cyber resilience in the operating environment and life cycle to meet stakeholders’ needs for mission assurance and resilient safety. The objectives can be further divided into a number of

Table 4.5 Description of the four top-level goals of cyber resilience

Goals	Description
Prevention Definition: Maintaining a state of informed preparedness for adversity	Adversity refers to unfavorable conditions, pressure, attacks, or compromises on network resources. Adverse conditions may include natural disasters and structural failures (such as power outages). Pressure can include unexpected high-performance loads. Adversity can be caused or exploited by APT actors. Informed preparedness includes contingency planning, including plans to mitigate and investigate threat events and respond to discovered vulnerabilities or supply chain compromises. Cyber threat intelligence provides critical information for informed preparedness Implementation strategies include deterrence, avoidance, and prevention strategies against potential threats, as well as planning (i.e., identifying available resources and developing plans to use those resources when a threat occurs), preparation (i.e., changing the set of available resources and training plans), and morphing (i.e., continuously changing the system to change the attack surface) and other strategies
Resistance Definition: Continuing to perform important missions or service functions in adversity	The meaning and realization of this goal do not need to rely on detection. The activities of APT actors may go undetected, or they may be detected but incorrectly attributed to user errors or other pressure. In order to achieve this goal, it is necessary to identify the basic organizational missions or service functions. In addition, supporting processes, systems, services, networks, and infrastructure must be identified. The criticality of resources and the ability of basic functions can change with time Implementation strategies include strategies to defend against the realization of potential threats even if they are not detected, specifically including absorption (accepting some level of damage to a given set of system elements, taking steps to reduce the impact on other system elements or the system as a whole, and automatically repairing the damage), deflection (transferring threat events or their effects to different system elements or systems other than the target or the initially affected system), and discarding (removing system elements or even the entire system according to signs of damage and replacing these elements or making the system or mission or service process run without them)

(continued)

Table 4.5 (continued)

Goals	Description
Recovery Definition: Restoring missions or service functions in and after adversity	The recovery of functions and data can be incremental. A key challenge is determining how much trust can be placed in the recovered functionality and data as the recovery progresses. Other threats or conditions in the operational or technical environment may interfere with recovery, and APT actors may seek to exploit obfuscation about the recovery process to establish a new foothold in the organization's system Implementation strategies include restoration (replicating a previous state that is known to be acceptable), reconstruction (replicating critical and supporting functionality to an acceptable level or using existing system resources), and replacement (replacing broken, suspect, or damaged system elements with new ones or reusing existing system elements to serve different functions to perform critical support functions)
Adaptation Definition: Modifying missions, service functions, and/or support capabilities in response to predicted changes in technology, service, or threat environment	Changes may occur on different scales and in different time frames, so tactical and strategic adaptation may be needed. Modifications may be applied to processes and procedures as well as technologies. Changes in the technology environment may include emerging technologies (e.g., artificial intelligence, 5G, and IoT) and the retirement of obsolete products. Changes in an organization's operating environment may result from changes in regulations or policies, as well as the introduction of new service processes or workflows. Analysis of these changes and the interactions between them can reveal how they modify the attack surface or introduce vulnerabilities Implementation strategies include remediation (removing or applying new controls to make up for identified vulnerabilities or weaknesses), hardening (reducing or controlling the attack surface), and updating (proactively directing controls, practices, and capabilities to anticipated, emerging, or potential threats) and other strategies. These strategies may result in redefinition (changing system requirements, architecture, design, configuration, acquisition processes, or operating procedures).

Table 4.6 Description of eight objectives of cyber resilience

Objectives	Description
Stop/ avoidance	Preventing the successful execution of attacks or the occurrence of adverse scenarios: This objective relates to an organization's preference for different approaches to risk. In some cases where restricting capabilities are available, risk or threat avoidance is a feasible way to respond to risks, and preventing threat events from occurring is another possible risk response measure
Preparing	Developing a realistic contingency plan to deal with predicted or anticipated adversities: This objective is driven by awareness of attacks and involves the organization's contingency plans, operational continuity plans, training, drills, and incident response and recovery plans for critical systems and infrastructure
Sustaining	Maximizing the duration and viability of critical missions or service functions in adversity: This objective specifically relates to basic functionality. Its evaluation is consistent with the definition of performance parameters, analysis of functional dependencies, and identification of critical assets. Note that shared services and public infrastructure, although not identified as essential per se, may be necessary for essential functions and therefore relevant to this objective
Restraining	Limiting the harm caused by adversity. This objective applies particularly to critical or high-value assets: network assets that contain or process sensitive information either directly carry critical missions or provide infrastructure services for critical missions
Understanding	Maintaining effective representation of missions, service dependencies, and resource status under possible adversities: This objective supports the achievement of all other objectives, most notably preparing, rebuilding, transforming, and reconstructing. An organization's plans for continuous diagnostics and mitigation, infrastructure services, and other services support this objective. Detecting anomalies, particularly suspicious or unexpected events or conditions, also supports the realization of this objective. However, this objective includes understanding resource dependencies and detection-independent states. The objective also involves the use of forensics and the sharing of cyber threat intelligence information among various organizations
Rebuilding	Restoring the mission or service functionality as much as possible after adversity: This objective involves essential functions, critical assets, and the services and infrastructure on which they rely. One key to achieving this objective is to ensure that recovery or reconstruction efforts generate trustworthy resources. This objective is not based on the analysis of the sources of adversity. Even if adversity is not detected, it can be triggered by continuously monitoring the timeliness, correctness, and availability of resources
Transforming	Modifying mission or service functions and the support processes to more effectively handle adversity and responding to environmental changes: This objective is especially applicable to the workflow and supporting process of key functions and the incident response and recovery plan of key assets and essential functions. Tactical modifications are typically procedural or configuration-related; longer-term modifications may involve restructuring operational processes or governance responsibilities while keeping the underlying technical architecture unchanged
Reconstructing	Modifying the architecture to more effectively deal with adversity and responding to environmental changes: This objective applies specifically to system architecture and mission architecture, including the technical architecture of systems that support mission or service functions. Furthermore, this objective applies to the architecture of critical infrastructure and services, which often support a variety of essential functions

sub-objectives, and the sub-objectives can be further divided into a number of representative activities or capabilities, as shown Table 4.7.

The demand objectives of cyber resilience reflect the specific requirements for implementing the top-level goals of cyber resilience. There is a many-to-many correspondence between the demand objectives of cyber resilience and the top-level goals of cyber resilience, as shown in Table 4.8.

4.9.3 Cyber Resilience Strategies and Design Principles

The framework of cyber resilience engineering includes 5 cyber resilience strategic principles (Table 4.9) and 14 cyber resilience design principles (Table 4.10). Among them, cyber resilience strategic principles are applied in the entire system engineering process to guide the direction of cyber-resilient system engineering analysis; cyber resilience design principles directly guide the architecture and design of cyber-resilient systems.

For a certain system, only some of these principles need to be considered. Cyber resilience strategic principles need to be consistent with the risk management strategy of the project, system, or mission owner, that is, the cyber resilience strategic principles required for the cyber-resilient system need to be selected and prioritized according to the relevant risk management strategy.

After determining the combination and priority of cyber resilience strategic principles, the combination and priority ranking of network flexibility design principles are determined accordingly. The correlation between the two is shown in Table 4.11. The selection strategy of cyber resilience design principles should also be consistent with the resilience design principles of other disciplines.

4.9.4 Cyber Resilience Technologies

Cyber resilience technologies describe how to achieve the top-level goals and requirements of cyber resilience. These approaches are complex and varied, depending on the type of systems (e.g., corporate information systems, cyber-physical systems, weapon systems) and other factors (e.g., architecture, management, and maturity of the technology). Cyber resilience technologies reflect an understanding of threats and the technologies, processes, and concepts associated with improving cyber resilience against threats.

Twelve of the cyber resilience techniques in Table 4.12 can be applied to adversarial or non-adversarial threats (including network-related and non-network-related threats), and the other two (deception and unpredictability) are specifically designed to address adversarial threats. Table 4.12 isolates the definition of cyber resilience technologies from ever-changing technologies and threats, thereby limiting frequent changes to the technology set.

Table 4.7 Description of cyber resilience objectives, sub-objectives, and representative indicators

Objectives	Representative sub-objectives	Representative indicators
Stop/ Avoidance	• Applying basic protection and control measures tailored to relevant system risks to limit exposure to threat events and reducing the gains of attackers • Modifying configuration based on threat intelligence	• Time for patching and modifying application • The percentage of resources to be configured and changed randomly • Percentage of resources to apply lifetime limit • Percentage of sensitive data assets encrypted • How long the attack spends in the deceptive environment • Percentage of resources that have more restrictive permissions automatically applied in response to threat indicators
Preparing	• Creating and maintaining network emergency plans • Maintaining resources required to execute network emergency plans • Verifying the effectiveness of network emergency plans through tests or drills	• Number of cyber emergency plans in the cyber tactical manual • Percentage of cyber playbooks that have at least one plan addressing identified threat types, threat action categories, or tactics, techniques, and procedures (TTPs) (refer to the identified threat model) • Percentage of network resources backed up • The time since the backup communication path was last used • Percentage of administrative staff who have been trained in cyber emergency response responsibilities • Time since the last (random, scheduled) drill or simulation of one or more cyber emergency plans
Sustaining	• Minimizing the degradation of service delivery • Minimizing disruption to service delivery • Ensuring normal operation	• Time to perform mission or service function impairment assessment • The length of time a specified mission or service function is performed below an acceptable level • Time from initial interruption to when basic functionality is available (at a minimum acceptable level) • Percentage of critical data assets whose data quality has been verified • Percentage of basic processing services that have verified functional correctness
Restraining	• Identifying potential threats • Isolating resources • Mobile resources • Changing or remove resources and	• Percentage of critical components with tamper resistance, shielding, and power line filtering; time from initial indication or warning to completion of scanning for potentially corrupt

(continued)

Table 4.7 (continued)

Objectives	Representative sub-objectives	Representative indicators
	the way they are used in order to limit future or further damage	resources • Time from initial indication or warning to completion of component isolation • Time from initial indication or warning to completion of resource relocation • Time from initial indication or warning to completion of switching to alternatives
Understanding	• Understanding the attack • Understanding dependencies between systems containing network resources • Understanding resource status related to threat events • Understanding the effectiveness of security controls that support cyber resilience	• Time from receipt of threat intelligence to determination of its relevance • Residence time of an attack in a deceptive environment • The time since the last refresh of mission dependency or functional dependency diagram • Time since last network desktop exercise, red team drill, or automatic interruption of execution control • Percentage of system elements for which a fault or potential fault indication can be detected • Percentage of network resources monitored • Number of attempted intrusions that stopped at the network boundary. • Average length of time to recover from an incident
Rebuilding	• Identifying untrusted resources and damage • Recovering functions • Enhancing protection during reconstruction • Determining the trustworthiness of recovered or reconstructed resources	• Time taken to identify unavailable resources and represent corruption in status visualization • The time between initiation of recovery procedures and completion of milestones documented in the recovery, contingency, or continuity operation plan • Percentage of network resources that maintain access control throughout the recovery process • Percentage of network resources to which additional auditing or monitoring is applied during and after the recovery process • Time to bring the backup network intrusion detection system online • Percentage of reorganized network resources placed in restricted enclaves over a period of time after reorganization

(continued)

Table 4.7 (continued)

Objectives	Representative sub-objectives	Representative indicators
		• Percentage of recovered or reconstructed (mission-critical, security-critical, support) data assets that are checked for data integrity/quality
Transforming	• Redefining missions or service process threads for agility • Redefining missions or service functions to reducing risks	• Percentage of missions or service process threads that have been analyzed for common dependencies and potential single points of failure • Percentage of missions or service process threads for which alternative courses of action have been documented • Percentage of essential functions that can be determined not to rely on resources shared with non-essential functions • Percentage of problematic data feeds for which risk mitigation measures have been applied since the last analysis
Reconstructing	• Restructuring systems or subsystems to reduce risks • Modifying systems or subsystems to reduce risks	• Size of the attack surface (hardware, software, supply chain, users, privileged users) • Percentage of the system components whose source can be determined • Percentage of system components that can be selectively isolated • Percentage of network resources for which customized analytics have been developed • Percentage of mission-critical components for which one or more customized alternatives have been implemented

Table 4.13 gives examples of typical methods for each cyber resilience technologies. The set of methods for a specific technology is not exhaustive but represents relatively mature techniques and practices.

When selecting a cyber resilience technology or technology combination through risk management strategies, design principles, and the priority of cyber resilience objectives, synergies, and conflicts between various technologies need to be considered. In particular, a good cyber-resilient system structure should be built to take full advantage of different cyber resilience techniques.

Different cyber resilience techniques can interact in a variety of ways. For example, the difference between the unpredictability technique and other techniques

Table 4.8 Correspondence between the demand objectives of cyber resilience and the top-level goals of cyber resilience

Objectives Goals	Prevention	Resistance	Recovery	Adaptation
Understanding	√	√	√	√
Preparing	√	√	√	√
Stop/avoidance	√	√		
Sustaining		√	√	
Restraining		√	√	
Rebuilding			√	
Transforming			√	√
Reconstructing			√	√

is that it is always used in conjunction with some other techniques (e.g., using dynamic positioning techniques to establish and reposition potentially interesting targets at unpredictable times). Because there are natural synergies and conflicts between cyber resilience techniques, systems engineering trade-offs must be made. In some cases, one technology may be dependent on another, so the former cannot be realized without the latter. For example, adaptive responses depend on analytical monitoring or situational awareness because responses require stimulation. In other cases, one technology supports another, making the latter more effective. For example, variety and redundancy are mutually supportive. It is also possible that one technology uses another technology, so more design options can be used than applying these technologies individually. For example, analytical monitoring can use diversity in design, including a diverse set of monitoring tools. However, one technology may also conflict with or complicate the use of another technology. For example, both diversity and segmentation may make analytical monitoring and situational awareness more difficult. Designs that incorporate diversity require monitoring tools that can handle different sets of system elements, and implementation of segmentation may limit the visibility of such tools. No one technology or one technology set is optimal for all decision factors. Determining the appropriate technology is a decision made by the systems engineer after considering all relevant factors. A growing number of technologies are dedicated to cyber resilience engineering, such as moving target defense and deception toolkits. As these technologies are more widely adopted, decisions to incorporate them are more influenced by policies than by consideration of technical risks. The cyber resilience technology is an active research field, and technical professionals are actively exploring cyber resilience techniques for cyber-physical systems, high reliability, dedicated systems, and large-scale processing environments.

Table 4.14 presents the interaction (synergy or conflict) between different cyber resilience techniques.

Table 4.9 Cyber resilience strategic principles

Strategic design principles	Core idea
Paying attention to public-critical assets	Limited organizational and planning resources need to be applied where they can provide the greatest benefit. Focus on assets that are both important and common first, followed by either important or common assets. Need to understand which mission or service functions, missions, capabilities, and assets are critical. Need to understand which resources, assets, or services are critical to the successful execution of critical functions and missions or the protection of critical assets. Focus first on the security and cyber resilience of those essential resources that are common across multiple functions
Supporting agility and architectural adaptability	The threat pattern will change as adversaries evolve, and the technologies and the way individuals and organizations use them will also change. Both agility and adaptability are integral parts of risk management strategies in response to the risk framework assumption that unforeseen changes in the threat, technology, and operating environment will occur throughout the life of the system. Prepare for changes in the technology, operations, and threat environment. Leverage existing and emerging standards to support interoperability. Recognize that organizations can invest in capabilities or create programs for different purposes and different time frames to manage risks caused by dependencies or other interactions between programs or plans
Reducing attack surface	Large attack surfaces are difficult to defend and require continuous efforts to monitor, analyze, and respond to anomalies. Reducing the attack surface cuts the cost of continuous protection and forces attackers to focus on much fewer locations, resources, or environments that can be more effectively monitored and defended. Consider your organization's attack surface—not just the exposed elements of your systems but also your people and processes. Consider how attackers attack the development, operation, and maintenance environments. Consider the attack surface in the network supply chain. Consider social media exposure and internal threats
Assuming resources will be damaged	Attacks on components, systems, chips, software modules, or ongoing services can go undetected for a long period of time. In fact, some damage may never be detected. Even so, the system must be able to meet performance and quality requirements. Systems and mission or service processes need to minimize the hazards that may result from damage to a specific types of products or technologies. The possibility of lateral movement of attacks and cascading failures needs to be considered. The potential consequences of management-critical components, services, or technologies being compromised or attacked must be analyzed and prepared for
Anticipating opponents to evolve	It is expected that advanced cyber-attackers will invest time and effort in intelligence gathering to improve their existing capabilities and develop new ones. Attacks will evolve with the development of new technologies, the opportunities provided

(continued)

Table 4.9 (continued)

Strategic design principles	Core idea
	by the application of new technologies, and the understanding of defenders. Over time, tools developed by advanced attackers will be available for less sophisticated attacks. Therefore, systems and missions need to be resilient in the face of unexpected attacks. It is necessary to think from an adversarial perspective when analyzing architectural changes, design modifications, and changes to operating procedures and governance structures and use but not be restricted by cyber threat intelligence

4.9.5 Cyber Resilience Construction Plan

Cyber resilience construction is oriented and starts from risk management. According to the organization’s risk management strategy, system engineers determine the top-level priorities of cyber resilience solutions by analyzing the potential impact of risks, specific threat events, or malicious cyber activity types and decompose them layer by layer into a series of demand objectives, sub-objectives, and activities or capabilities and then select appropriate technologies and methods for each activity/capability under the guidance of cyber resilience strategies and design principles, as shown in Fig. 4.13.

Cyber-resilient systems are the result of the engineering selection, prioritization, and application of cyber resilience strategies and design principles, technologies, and implementation methods. Table 4.15 gives the mapping relationship between cyber resilience principles and cyber resilience objectives and technologies, and Table 4.16 gives the mapping relationship between cyber resilience techniques and cyber resilience objectives.

4.9.6 Cyber Resilience Throughout a System’s Life Cycle

Cyber resilience is a non-functional characteristic that runs through all stages and processes of a system’s life cycle. The matters needing attention in applying the concept and engineering framework of cyber resilience at each stage of the system’s life cycle, including the goals, objectives, design principles, and technologies of cyber resilience, are briefly analyzed as follows.

During the conceptual phase, cyber resilience goals and objectives are customized based on the usage requirements of the relevant systems. First, the purpose of customization is to prioritize cyber resilience goals and objectives for stakeholders. The organization’s risk management strategy is used to help determine which strategic design principles are most relevant. Strategic design principles and corresponding structural design principles should be consistent with the design principles of other disciplines. The nominal or candidate system architecture is

Table 4.10 Cyber resilience design principles

Design principles	Core idea
Maintaining situational awareness	Including awareness of possible performance trends and anomalies to provide information for guidance and decision-making on network actions and ensure the completion of missions
Fully utilizing operational and status data	Health and status data can be used to support situational awareness, indicate potentially suspicious characteristics, predict and adapt to changing operational needs
Determining ongoing trustworthiness	Periodic or regular check and/or verification of the integrity or correctness of data and software increases the workload required for attackers to modify or create data and functions. Likewise, periodic or regular analysis of the characteristics of individual users, system components, and services can increase suspicion and trigger responses such as closer monitoring, tighter permissions, or isolation
Restricting the need for trust	Limiting the number of system elements that need to be trusted (or the length of time that elements need to be trusted) can reduce the level of effort required for assurance, continuous protection, and monitoring
Controlling usage and visibility	Controlling what can be discovered, observed, and used increases the effort required by attackers to expand their foothold or increase their influence in systems containing network resources
Curbing and excluding behavior	Limiting what can be done and where an action can be taken can reduce the likelihood or extent of harm or disruption propagating between components or services
Layer defense and partition resources	The combination of defense in depth and partitioning increases the effort required for attackers to overcome multiple defenses
Adaptive management (of risks)	Adaptive management of risk supports agility and provides complementary risk mitigation throughout critical operations despite component interruption or downtime
Diverse planning and management	Diversity is a proven resilience technology that eliminates single points of attack or failure. However, architecture and design should consider cost and manageability to avoid introducing new risks
Maintaining redundancy	Redundancy is key to many resilience strategies, but it can degrade over time as configurations are updated or connectivity changes
Diversifying resource locations	Resources bound to a single location (e.g., services running on a single hardware component, databases located in a single data center) can easily become single points of failure and become high-value targets
Maximizing transiency	The use of transient system elements minimizes the duration of exposure to attack activities, while

(continued)

Table 4.10 (continued)

Design principles	Core idea
	regular refreshes to a known (safe) state can clean up malware or corrupted data
Changing or destroying the attack surface	The destruction of attack surface will cause attackers to waste resources, make wrong assumptions about the system or defenders, or launch attacks or disclose information prematurely
Creating deceptive effects and unpredictability that are transparent to users	Deception and unpredictability can be highly effective techniques against attacks, causing them to expose their presence or TTP and waste their effort. But if applied incorrectly, these techniques can also confuse users

analyzed according to the realization degree of the priority of cyber resilience goals and objectives and the application degree of related cyber resilience design principles. Tailoring of objectives can also be used to identify or define the potential metrics or effectiveness metrics of cyber resilience solutions. Also, risk management strategies that limit risk response or handling (e.g., use of specific technologies, interoperability requirements with, or dependencies on other systems) are used to help determine which technologies and methods can or cannot be used for cyber resilience solutions. In addition, during the conceptual phase, it is necessary to identify the cyber resilience problems of production, integration, verification, and supply chain management systems and define strategies to address these problems.

During the development phase, relevant cyber resilience structural design principles (i.e., those principles that can be applied to the selected system architecture and support the cyber resilience strategic design principles) need to be prioritized based on the priority of the cyber resilience objectives. It is determined whether they can be used in the selected system architecture by analyzing the cyber resilience techniques and methods dictated by structural design principles. Cyber resilience solutions are defined and analyzed based on potential effectiveness and other aspects of compatibility related to trustworthiness. Potential effectiveness analysis considers the relative effectiveness of the solution to potential threat events or scenarios and effectiveness metrics of cyber resilience objectives. The analysis of compatibility with other aspects of trustworthiness takes into account the potential synergies or conflicts related to the technologies, design principles, or practices specific to other engineering disciplines, especially security, reliability, survivability, and safety. While systems containing network resources are inevitably subject to adversity, the operational processes and procedures for using technical solutions need to be defined, refined, and verified in terms of their ability to meet mission and service objectives. It is necessary to analyze and evaluate the implementation of cyber resilience solutions, including measuring the performance of mission or service functions through adversarial tests or demonstrations in a stressful environment with adversarial activities. Cyber resilience itself requires testing and other forms of check or verification, including adversarial threats and/or other stresses on the

Table 4.11 Correlation between cyber resilience strategic principles and design principles

Strategic principles Design principles	Paying attention to public- critical assets	Supporting agility and architectural adaptability	Reducing attack surface	Assuming resources will be damaged	Anticipating opponents to evolve
Maintaining situa- tional awareness	√				√
Fully utilizing opera- tional and status data	√	√		√	√
Determining ongoing trustworthiness	√			√	√
Restricting the need for trust			√	√	
Controlling usage and visibility	√		√	√	
Curbing and excluding behavior	√			√	√
Layer defense and partition resources	√			√	
Adaptive management (of risks)	√	√		√	√
Planning and manag- ing variety	√	√		√	
Maintaining redundancy	√	√		√	
Diversifying resource locations	√	√		√	√
Maximizing transiency			√	√	√
Changing or destroying the attack surface			√	√	√
Creating deceptive effects and unpredictability that are transparent to users		√	√		

system. In addition, it is necessary to develop the resources needed for implementing specific action plans (such as different implementations of key system elements and alternative treatment facilities).

During the production phase, the verification strategy is applicable to the instance or version of the system of interest and the associated spare parts or components. Verification strategies for cyber resilience requirements applicable to such instances and system elements include adversarial testing or demonstrations in stressful environments. In addition, during the production phase, cyber resilience problems of the systems supporting production, integration, verification, and supply chain management will continue to be identified and addressed.

Table 4.12 Description of 14 technologies for cyber resilience

Technologies	Intention
Definition of adaptive response: Implementing agile actions to manage risk	The ability to respond to adverse conditions, pressures, attacks or related indicators in a timely and appropriate manner, so as to maximize the maintenance of mission or service operations, limit the impact, and avoid instability
Definition of analytical monitoring: Continuously monitoring and analyzing a wide range of attributes and characteristics in a coordinated manner	Providing the ability to detect potential adverse conditions to the maximum extent; revealing the extent of adverse conditions, pressure, or attack; identifying potential or actual damage and investigating the TTP of the attacker; providing data needed for situational awareness
Definition of situational awareness: Constructing and maintaining the current status of the organization's mission or service functions while considering threat events and action plans	Supporting situational awareness; improving understanding of dependencies between network and non-network resources; revealing patterns or trends of attack characteristics
Definition of coordinated protection: Ensuring that protection mechanisms operate in a coordinated and effective manner	Threat events are required to overcome multiple protective measures (i.e., adopt a defense-in-depth strategy). In the case of an adversarial threat, increasing the cost of attacks and the possibility that attackers are detected can increase the difficulty of attacking key resources. Regardless of the type of threat event, it must be ensured that the use of any given protection mechanism does not have adverse, unintended consequences by interfering with other protection mechanisms. Cyber action guidelines must be verified to be realistic
Definition of deception: Misleading, confusing, hiding key assets from adversaries or exposing secretly contaminated assets to attackers	Mislead, confuse, or hide critical assets from an adversary so that the adversary is unsure how to conduct an attack; delay the effectiveness of an attack, increase the risk of being detected, cause the adversary to waste its resources, and expose the adversary's espionage technology as early as possible
Definition of diversity: Using heterogeneity to minimize common-mode failures, especially common vulnerability threats	Limiting the potential of losing critical functionality due to failure of replicated common critical components; in the case of an adversarial threat event, developing malware or other TTPs suitable for multiple targets will force the adversary to make greater efforts; applying TTPs to targets for which they are unsuitable will increase the possibility of the attacker wasting or exposing TTPs; maximizing the possibility that some of the organization's defense systems will survive an attack

(continued)

Table 4.12 (continued)

Technologies	Intention
Definition of dynamic positioning: Allocating and dynamically relocating functions or system resources	Improving the ability to recover quickly from non-adversarial events (e.g., fire, flood) and adversarial threat events (e.g., cyber-attacks); reducing the adversary's ability to locate, eliminate, or destroy mission or service assets and force the adversary to spend more time and effort searching for the organization's critical assets, thereby increasing the likelihood that an attack will prematurely expose its presence, operations, and espionage techniques
Definition of non-persistence: Generating and retaining resources as needed or for a limited time	Reducing exposure to vandalism, modification, or damage; in the event of an adversarial threat, providing a way to reduce the intrusion and advancement of attacks and removing malware or corrupted resources from the system; limiting attacks that can target resource availability
Definition of privilege restriction: Restricting permissions based on attributes of users and system elements, as well as environmental factors	Restricting the unexpected characteristics of authorized individuals will endanger the influence and possibility of information or services. Stop attackers by forcing them to invest more time and energy to get credentials. Restrict the attackers' ability to use the credentials they obtained
Definition of readjustment: Building systems and resources to meet the needs of mission or service functions, reduce current and anticipated missions, and adapt to the evolution of the technology, operation, and threat environment	Minimizing connections between mission-critical and non-mission-critical services to reduce the likelihood of the latter affecting the former; reducing the defense organization's attack surface by minimizing the potential for non-mission or non-service functions to be used as attack vectors; adapting to changing mission or service function needs; adapting to changes in the technological environment
Definition of redundancy: Providing multiple protected instances of critical resources	Reducing the consequences of loss of information or services; facilitating recovery from the impact of adverse cyber incidents; limiting the time during which critical services are denied or restricted
Definition of segmentation: Defining and separating system elements based on criticality and trustworthiness	Curbing adversarial activities and non-adversarial pressures (e.g., fires, floods) on the enclaves, systems, or facilities that have been established; limiting the set of possible targets to which malware can easily spread
Definition of integrity verification: Determining whether critical system elements are corrupted	Helping determine correct results when conflicts occur between different services or inputs and detect compromised data, software/hardware, and illegal modification or fabrication
Definition of unpredictability: Making changes randomly or unpredictably	Increasing attackers' uncertainty about the protection of systems they may encounter, making it more difficult for them to determine an appropriate course of action; serving as a force multiplier for other technologies

Table 4.13 Fourteen cyber resilience techniques and 50 implementation methods

Adaptive response	Adaptive response	Coordinated protection	Situational awareness	Deception	Variety	Dynamic positioning
Dynamic reconstruction; Dynamic resource allocation; Adaptive management	Monitoring and damage assessment; Sensor fusion analysis; Forensics and characterization	Calibration of defense-in-depth; Consistency analysis; Orchestration Self-Protection	Dynamic resource awareness; Dynamic threat awareness; Mission dependence and status visualization	Confuse; Fake-information; Misguiding; contamination	Architecture diversity; Design diversity; Composition diversity; Information diversity; Path diversity; Supply Chain diversity	Dynamic relocation of sensors; Migration of functional network resources; Fragmented distribution of asset liquidity
Non-persistence	Privilege restriction	Readjustment	Redundancy	Segmentation	Verification of integrity	Unpredictability
Non-persistent information	Trust-based privilege management;	Allowing uninstallation;	Protected backup and recovery;	Dynamic segmentation and isolation of predefined segments	Integrity check;	Temporal unpredictability;
Non-persistent services	Attribute-based usage restriction;	Restricting substitution;	Margin replication		Origin tracking;	Environmental unpredictability
Non-persistent connections	Dynamic privileges	Customizing evolution			Feature verification	

Table 4.14 Potential interactions between different cyber resilience techniques

	Adaptive response	Analytical monitoring	Situational awareness	Coordinated protection	Deception	Diversity	Dynamic positioning	Non-persistence	Privilege restriction	Readjustment	Redundancy	Segmentation	Integrity verification	Unpredictability
Adaptive response	D		U	S		U	U/S	U/S	U/S		U	U/S	U	U
Analytical monitoring	S		S	D	U	U	U						U/S	
Situational awareness	S	U			S					S			U	
Coordinated protection	U	S				U	U	U	U/S	U	U	U		
Deception	U/S	U/C	C/S				U					U	S	U
Diversity	S	C/S	C	C/S			S		U	U	U/S		U	S
Dynamic positioning	U/S	C/S			S	U		U			U			U/S
Non-persistence	U/S	C	C				S			S			U	S
Privilege restriction	S			U						S			U	
Readjustment	C		U	C/S		C/S			S		C			
Redundancy	S					U	S						U	
Segmentation	U/S	C		S	S									U
Integrity verification	S	S/U	S		U	S		S	S		S			
Unpredictability	C/S	C		C	S	U	U/S	U						

Meanings of symbols:

- “S” means that the technology (Technology A) in the row supports the technology (Technology B) in the column and Technology A makes Technology B more effective.
- “D” means that Technology A depends on or is enabled by Technology B. If not combined with or enabled by Technology B, Technology A will be ineffective.
- “U” means that Technology A uses or enables Technology B. Technology A can be effectively implemented without Technology B. However, if the two technologies are used together, there will be more options.
- “C” means that Technology A may conflict with or complicate Technology B. The partial or full implementation of Technology A may undermine the effectiveness of Technology B.

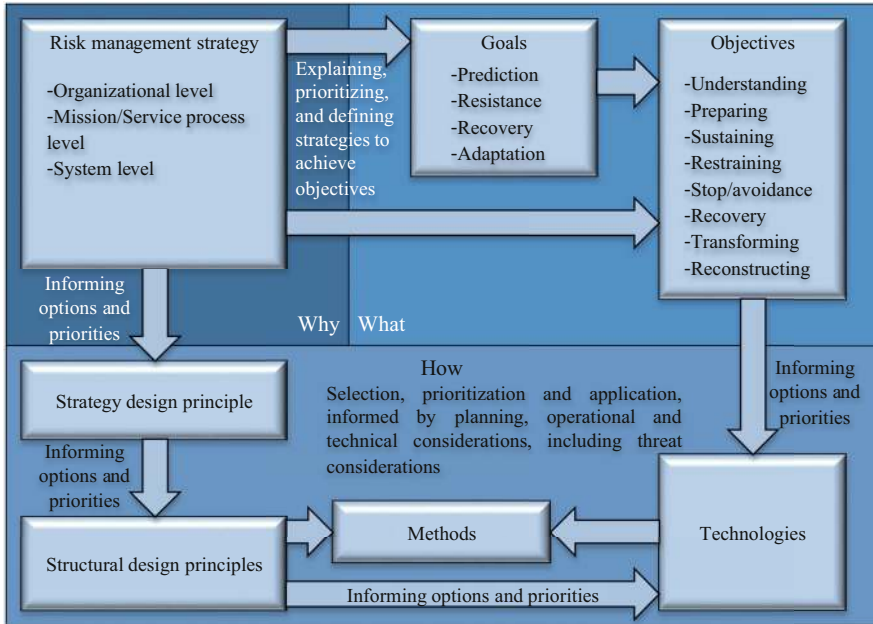


Fig. 4.13 Cyber resilience construction plan

During the phase of use, it is necessary to monitor the effectiveness of cyber resilience solutions in the operating environment. The effectiveness may be reduced due to changes in the operating environment (e.g., new mission or service processes, new stakeholders, increased number of users, configuration drift, deployment in new locations, addition or removal of systems of interest and systems or system elements that interact with them), changes in the threat environment (e.g., new threat actors, new vulnerabilities in commonly used technologies), or changes in the technology environment (e.g., introduction of new technologies into other systems that interact with the system of interest). Cyber resilience solutions may need to be adjusted to cope with such changes (e.g., defining new action plans, reconfiguring system elements, changing mission or service processes and procedures). The relative priority of cyber resilience objectives may change according to the changes in stakeholders and their concerns, mission or service processes, or project funding. Finally, changes in the threat or technical environment may make certain technologies or methods less feasible, while changes in technology or operating environment may make other technologies or methods more feasible.

During the support phase, maintenance and upgrades of systems or elements enable the integration of new cyber resilience solutions into systems of interest. This phase also provides an opportunity to re-examine the priority and adjustment of cyber resilience objectives. System functionality upgrades or modifications may include significant architectural changes to address the cumulative changes to the operational, threat, and technical environment. System changes and upgrades, especially architectural changes, may also introduce additional vulnerabilities.

Table 4.15 Mapping relationship between cyber resilience principles and cyber resilience objectives and technologies

Objectives and techniques	Understand	Prepare	Stop/avoid	Sustain	Restrain	Rebuild	Transform	Reconstruct	Analytical monitoring	Situational awareness	Integrity verification
Strategy and design principles											
Pay attention to public-critical assets	S		S	S		S		S	U	U	U
Agility and architectural adaptability		S				S	S	S			
Reduce attack surface	S		S		S		S	S			
Assume resources will be damaged		S		S	S		S	S		U	
Anticipate opponents to evolve	S	S					S	S	U	U	
Maintain situational awareness	S						S		R	R	
Utilize historical operating data	S						S		R	R	U
Ensure continuous trustworthiness	S		S	S	S	S					R
Limit the need for trust			S		S		S				U
Control usage and visualization			S	S	S						
Curb and exclude behavior				S	S			S	U		U
Layer defense and partition resources			S	S				S	U		
Adaptive management (of risks)		S		S		S					
Plan and manage variety		S		S	S	S					
Maintain redundancy				S	S	S					
Diversify resource locations				S	S	S					
Maximize transiency			S		S						U
Alter or destroy the attack surface			S								
Deception and unpredictability	S		S								

Objectives and techniques Strategy and design principles	Privilege restriction	Segmentation	Coordinated protection	Redundancy	Diversity	Dynamic positioning	Adaptive response	Non- persistence	Unpredictability	Deception	Readjustment
Pay attention to public-critical assets		U	U		U			U			R
Agility and architectural adaptability				U	U	U	U	U			R
Reduce attack surface		U									U
Assume resources will be damaged	U	U					U	U			U
Anticipate opponents to evolve					U				U	U	U
Maintain situational awareness											
Utilize historical operating data											
Ensure continuous trustworthiness			U								
Limit the need for trust	R		U								U
Control usage and visualization	R	U						U		U	
Curb and exclude behavior	U	R			U		U	U			
Layer defense and partition resources		R	R	U	U	U					
Adaptive management (of risks)	U	U	U	U		U	R	U	U	U	U
Plan and manage variety			R	U	R						C
Maintain redundancy			U	R	U						U
Diversify resource locations				U	U	R	U	U	U		
Maximize transiency						X	U	U	U		
Alter or destroy the attack surface					U	U	R	U	U	U	
Deception and unpredictability			R				X		R	R	

S support the realization of objectives; R resilience technology required; U resilience technology available. X cannot be used in combination

Table 4.16 Mapping relationship between cyber resilience techniques and cyber resilience objectives

Objectives	Technologies	Understand	Prepare	Stop/avoid	Sustain	Restrain	Rebuilt	Transform	Reconstruct
	Adaptive response	✓	✓	✓	✓	✓	✓		
	Analytical monitoring	✓			✓	✓	✓		
	Situational awareness	✓	✓		✓		✓		
	Coordinated protection	✓	✓	✓	✓		✓	✓	✓
	Deception	✓		✓					
	Diversity		✓	✓	✓	✓			✓
	Dynamic positioning	✓		✓	✓	✓	✓		
	Non-persistence			✓		✓		✓	✓
	Privilege restriction			✓		✓	✓		
	Readjustment			✓				✓	✓
	Redundancy		<	✓	✓		✓	✓	✓
	Segmentation			✓		✓	✓		✓
	Integrity verification	✓			✓	✓	✓		
	Unpredictability			✓		✓			

Table 4.17 Cyber resilience throughout the system’s life cycle

Phases in the life cycle	The role of cyber resilience structures
Conceptual phase	– Prioritizing and adjusting objectives – Prioritizing design principles and aligning them with other disciplines – Defining a set of techniques and methods for use in the solution
Development phase	– Using design principles to analyze and form designed structures – Using techniques and methods to define alternative solutions – Developing the ability to achieve prevention/avoidance, sustainment, containment, and reconstruction and to understand objectives
Production phase	– Implementing cyber resilience solutions and evaluate their effectiveness – Providing resources (or ensure resources will be provided) to achieve preparation objectives
Application phase	– Monitoring cyber resilience solutions’ effectiveness using the ability to achieve understanding and preparation objectives – Reordering and adjusting the priorities of objectives as needed, and adjust mission, service, and/or security processes to cope with environmental changes (change objectives)
Support phase	– Reviewing and adjusting the priority of objectives; use monitoring results to identify new or modified requirements – Re-examining the limitations on techniques and methods – Modifying or upgrade the capabilities consistent with the above changes (reconstructing objectives)
Decommissioning phase	– Prioritizing and adjusting the objectives for the operating environment – Ensuring that the disposal process can achieve these objectives and modifying or upgrading the capabilities of other systems (reconstructing objectives) if necessary

During the decommissioning phase, system elements or the entire system of interest are removed from the operating environment. The decommissioning process impacts other systems with which the system of interest interacts and may reduce the cyber resilience of these systems and the supported mission or service processes. Decommissioning strategies may include phased removal of system elements, whole-sale removal of all system elements, phased replacement of system elements, and wholesale replacement of the entire system of interest. The cyber resilience objectives and priorities for systems, missions, and service functions in the operating environment should be determined, so as to analyze the potential or expected impact of different decommissioning strategies on the ability to achieve these objectives. Like the support phase, the decommissioning phase may also introduce significant vulnerabilities, particularly during the disposal and decommissioning of unexpected residues in assets.

Table 4.17 lists the key changes in different cyber resilience structures during each phase of the system’s life cycle.

4.9.7 Application Fields of Cyber Resilience

Cyber resilience can be applied to different fields [23] (or different layers in the system concept hierarchy), as shown in Table 4.18. The application of cyber

Table 4.18 Application fields of cyber resilience

Application field/layer	Examples	Resilience methods in relevant disciplines
Hardware/firmware	FPGA, MPSoC, general and dedicated processors, embedded firmware	Fault-tolerant hardware
Network/communication	Communication media, network protocols	Communications network resilience, especially the use of redundancy
System/network components	Firewall, servers, thin clients	Fault-tolerant design
Mobile system/network components	Laptops, tablets, smartphones, PDAs (temporarily or intermittently part of a system or network)	Fault-tolerant design (especially tolerance for connectivity degradation)
Operating system	General operating system, RTOS	Fault-tolerant design
Cloud computing, virtualized computing and/or middleware infrastructure	VMM, hypervisor, SOA infrastructure/shared services	Fault-tolerant design; provide predictable and load-balancing service middleware
Mission/service function, application/service	Customized DBMS, workflow management software; specialized mission applications	Fault-tolerant design
Software	Software running on system/network components (including operating systems, cloud, virtualization, middleware, DBMSs, applications, and services)	Fault-tolerant design
Information storage	Databases, knowledge bases, unstructured collections (big data)	System resilience, using redundancy for backup, failover, and recovery/rollback
Information flow	RSS feeds, Twitter, instant messaging/chat, video feeds	Communications network resilience, especially the use of redundancy
System	A comprehensive set of the above, within the control of a single management or administration	System resilience using redundant functions for backup, failover, and recovery
System of systems (hierarchy)	A multi-layer-controlled system supports a given mission or a group of missions through interoperability. Within an organization, systems of systems are at level 2 of the risk management hierarchy; however, systems can span multiple organizations	System resilience using redundant functions for backup, failover, and recovery; use of redundancy for the recovery of backup communication path network

resilience to different fields and layers can draw on methods from a wider range of disciplines, such as fault-tolerant computing, communications network resilience and redundant backup, failover, and system recovery.

In addition, the cyber resilience techniques and methods that are most relevant and useful to the system depend on the type of the system. A brief description of typical systems and applicable technologies and methods includes, but is not limited to, the following.

1. Corporate Information Systems, Shared Services, and General Infrastructure

Corporate information systems are typically general-purpose computing systems with significant processing capability, storage, and bandwidth, capable of providing information resources that meet the business or other needs of enterprises or large stakeholders. Therefore, all cyber resilience techniques and related methods are feasible, although their selection also depends on policy, cost, and other considerations.

2. Large-Scale Processing Environment

Large-scale processing environments can handle large number of events and data (such as process transactions) with a high degree of trust in service delivery. The scale of such systems makes them highly sensitive to service interruption or degradation. Therefore, the selective use of uninstallation and limiting the methods of implementation can make the scale of such systems more manageable. In turn, this will support the application of analytical monitoring, task dependency, and state visualization methods to situational awareness in a way that does not significantly affect performance. Large-scale processing environments provide the possibility for dynamic positioning and repurposing of network resources through functional relocation, fragmentation, and distribution to help improve network resilience.

3. Systems of Systems

Many cyber resilience techniques may be applicable to systems of systems, but some technologies and methods may provide better application than others. For example, situational awareness implemented through mission dependencies and status visualization can be used to predict the potential impact of adversary activities on the mission of constituent systems or network of system elements. Calibrated defense-in-depth and coherence analysis approaches to coordinated protection technologies can help ensure that the different protections that make up a system operate in a consistent and coordinated manner to prevent or delay adversaries' damage to the system. Re-adjustment may also be beneficial for systems involving systems that are not intended to work together and have different missions, functions, and risk frameworks. In particular, methods of uninstallation and limitation can be used to ensure that core system elements are properly aligned with the overall mission of system architecture.

4. Critical Infrastructure Systems

Critical infrastructure is typically a specialized, highly-trusted, and dedicated system with highly deterministic properties. The availability and integrity of

system functionality are very important, because damage to or lack of some critical system elements can lead to significant damage. For these reasons, employing system resilience techniques such as redundancy (especially protected backup, recovery, and surplus capacity methods) combined with variety techniques (e.g., architectural variety and supply chain variety) can prevent the mission or service consequences caused by attacks and maximize the probability of continuous operation of key or basic missions and services and can also isolate highly critical system elements in segments to protect them from attacks. Approaches such as trust-based privilege management and attribute-based usage restriction can limit the potential damage attacks can cause to the system.

5. Cyber-physical System

Cyber-physical systems (CPS) may have limitations in storage capacity, processing power, and bandwidth. In addition, many of these systems have a high degree of autonomy and limited human-computer interaction. Some cyber-physical systems operate without active network connections, although they may be connected to the network under certain circumstances (such as regular maintenance). Non-persistent services support periodic refreshing of software and firmware from trusted sources (such as offline redundant components), thus effectively removing any malware. However, this approach only works when the organization allows refreshes to cause periodic downtime. Similarly, key software encryption verification and physical integrity verification can help embedded systems to detect damaged software components.

6. Internet of Things

An IoT system consists of system elements that have network connectivity and communicate with software applications that have Internet access. These software applications are part of the IoT system, responsible for coordinating the characteristics of data provided by the system elements or aggregating data. System elements have limitations in power consumption, processing, storage capacity, and bandwidth, which in turn may limit the potential of such processing-intensive cyber resilience approaches, such as device-level obfuscation or adaptive management. Because many “things” (e.g., light bulbs, door locks) are small and relatively simple, they usually lack the ability to provide basic protection. However, integrity checking methods for proving integrity are still feasible when applied in conjunction with reliability mechanisms. IoT systems assume an Internet connection, although the set of “things” can usually run independently without connection. Since many IoT systems do not assume the technical expertise of users, cyber resilience techniques and methods involving human interaction (e.g., disinformation, misleading) may not be applicable. Furthermore, IoT systems are designed to accommodate the flexibility and reuse of the functionality of the constituent “things.” Therefore, an application that orchestrates the properties of one set of “things” may be upgraded to orchestrate the properties of other sets whose members are not designed with the application in mind. Such changes to the IoT system to which the application or add-on set originally belonged can benefit from the realigned application. At the IoT system level (rather than at the level of individual system elements), segmentation and consistency analysis can be applied.

4.10 Cyber Resilience Analysis and Assessment

4.10.1 *Cyber Resilience Analysis*

In the context of system security engineering, cyber resilience analysis seeks to determine whether a system's cyber resilience attributes and characteristics, regardless of the stage in its life cycle, are sufficient to allow an organization using the system to meet its mission assurance. In essence, the analysis of cyber resilience attributes and characteristics needs to be carried out in the procedural, operational, architectural, and threat environment, in which it is possible to explain the elements of cyber resilience and evaluate the relative priority of cyber resilience objectives and the applicability of cyber resilience design principles, techniques, and methods. Cyber resilience analysis can lead to architectural changes, integration of new products or technologies into the system, changes in the use of existing products or technologies, operating procedures, etc., which are in line with the organization's risk management strategies.

Table 4.19 gives a general, customizable cyber resilience analysis process, including steps and tasks.

4.10.2 *Static Assessment of Cyber Resilience*

Cyber resilience assessments should first answer the question of which aspects of cyber resilience are important. As shown in Table 4.20 [24], the aspects of cyber resilience that can be prioritized and evaluated include attributes, capabilities, and behaviors.

Through an assessment of the system's cyber resilience, what risks have been solved for stakeholders by the information and projects supported by the system and the information processed by the system? What other risks are there?

To carry out semi-quantitative or quantitative cyber resilience assessment, it is necessary to set some cyber resilience indicators, that is, measurement parameters or values calculated from other measured parameters. The cyber resilience problem domain overlaps with the problem domains of system security, resilience, and risk, and many indicators from these domains can be reused or improved to support cyber resilience analysis. Currently, more than 500 representative cyber resilience indicators have been included, some of which (especially stop/avoidance-related indicators) are derived from cyber security indicators, such as the percentage of network resources/users/system services whose priority can be changed dynamically. Other indicators (especially those related to recovery goals and constraints and reconstruction objectives) are derived from system resilience and risk indicators, such as the percentage of mission-critical applications whose integrity/behavior has been verified. Each item in this indicator contains the identification of the network resilience structure related to it; the type of system used or customized; the type of decision

Table 4.19 Customizable process for cyber resilience analysis

Analysis steps	Problem and motivation	Task
Understanding the relevant environment	How do the concerns and priorities of stakeholders translate into the constituents and priorities of cyber resilience?	• Determining the back-ground of the plan • Determining the architec-tural environment • Determining operating environment • Identifying the threat envi-ronment • Explaining the consider-ation of priorities of cyber resilience constituents
Establishing an initial cyber resilience baseline	What is the performance of the system in terms of cyber resilience (i.e., how well it meets the needs of stakeholders and address their concerns)? Is this important to stakeholders?	• Determining existing capa-bilities • Identifying gaps and prob-lems • Defining evaluation criteria and conducting preliminary assessment
Analyzing the system	How does cyber risk affect mission, ser-vice or operation risk?	• Identifying critical resources, sources of vulner-ability, and attack surface • Representing the adversary’s point of view • Identifying and prioritizing improvement opportunities
Developing cyber resilience solutions	How can mission or operational resilience be improved by increasing cyber resilience?	• Defining potential technical and procedural solutions • Defining potential solutions to support systems and pro-cesses • Analyzing potential solu-tions based on criteria
Making recommendations	What is the recommended action plan?	• Identifying and analyzing alternatives • Evaluating alternatives • Recommending action plans

Table 4.20 Evaluable or measurable aspects of a system’s cyber resilience

Attributes	Capabilities	Behaviors
How well —how well are cyber resilience goals or objectives achieved? How effective is the application of cyber resilience techniques or design principles? Qualitative or semi-quantitative assessment or measurement of behaviors Associate a set of predefined capability sets	How well —how well are cyber resilience sub-objectives achieved or expected to be achieved? To what extent are cyber resilience activities implemented? Qualitative or semi-quantitative assessment or measurement of behaviors Associate expected or unex-pected behaviors	How well, how fast, how complete, how confident —how an event unfolds? How is an activity performed? Direct observation or calcula-tion from a set of measure-ments Correlation of system resil-ience and mission performance

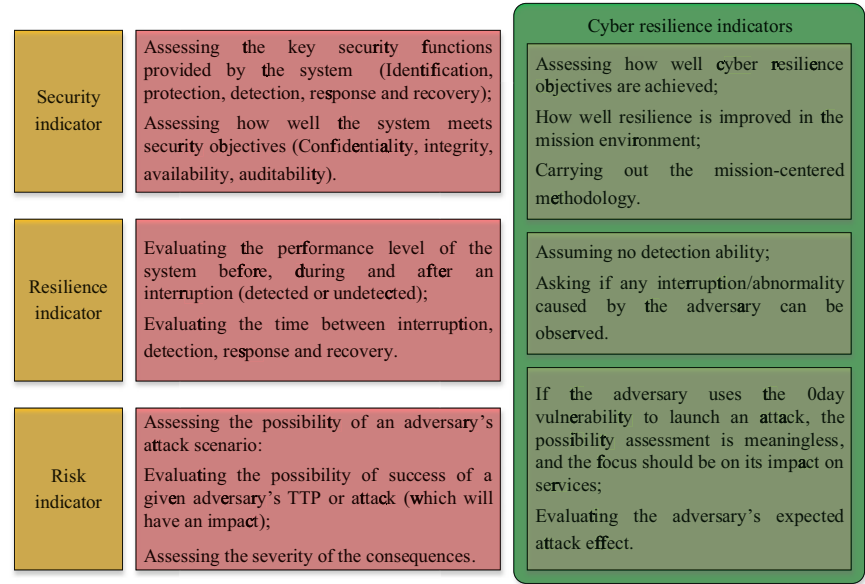


Fig. 4.14 Cyber resilience indicators can reuse security, resilience, or risk indicators

supported; the domain it belongs to; and how to evaluate it. This information can help users determine which common indicators may be relevant to a specific system or environment. The basic correspondence relationship is shown in Fig. 4.14.

Cyber resilience assessment [24] includes high-level qualitative assessment (for cyber resilience goals and objectives), a combination of qualitative and semi-quantitative overlay assessment (for cyber resilience sub-objectives and the application of related cyber resilience techniques), and detailed quantitative scoring assessments (evaluating cyber resilience enhancement activities, supporting sub-objectives and identifying them through cyber resilience techniques, assessing the impact of different solutions on activities, technologies, and sub-objectives).

High-level qualitative assessment sets several key differentiating factors for each cyber resilience objective (such as the depth, breadth, identification of the key factors, etc.) and then gives an assessment grade in a high-/medium-/low-qualitative way. Cyber resilience is a system-level attribute that cannot be determined solely from the attributes of individual components. The complexity of the system makes it more difficult to measure cyber resilience and the corresponding functions and behaviors. The typical behaviors of complex systems, such as composition, cascading, and feedback, need to be considered in analyzing and supporting cyber resilience measurement. Cyber resilience is closely related to the mission execution environment and must be assessed in a replicable and repeatable manner.

A combination of qualitative analysis and semi-quantitative ratings yields an overlay of the cyber resilience structure of related systems. The concerns and

Objective Option	Understanding	Preparing	Stop/avoidance	Sustaining	Restraining	Rebuilding	Transforming	Reconstructing
Option1								
Option2								
Option3								
Option4								

Fig. 4.15 Conceptual overlay of multiple resilience options

Cyber Resilience goals	Relative importance to	Baseline status	Cyber resilience objectives	Relative importance to stakeholders	Baseline status	Relative priority of improvements	Cyber resilience techniques	Relevance to the given architecture	Current use	Opportunity
Prevention	N/A		Understanding	M	L	M	Adaptive response	H	L	Important
Resistance	H	L	Preparing	L	L	L	Analytical monitoring	N/A		
Recovery	H	M	Stop/avoidance	M	L	M	Deception	N/A		
Adaptation	N/A	-	Sustaining	H	L	H	Diversity	L	L	Insignificant
			Restraining	H	L	H	Dynamic positioning	N/A		
			Rebuilding	H	M	M	Non-persistence	M	L	Important
			Transforming	M	L	M	Privilege restriction	H	L	Important
			Reconstructing	N/A			Segmentation	H	M	Moderate
							Coordinated protection	L	L	Smallest
							Dynamic representation	N/A		
							Readjustment	M	L	Important
							Redundancy	H	M	Moderate
							Integrity verification	H	L	Important
							Unpredictability	N/A	—	—

Fig. 4.16 An example of overlay of cyber resilience components

priorities of stakeholders are used to prioritize cyber resilience objectives and sub-objectives and, according to the identified threat environment, evaluate the capabilities or activities required by related sub-objectives, the potential applicability of cyber resilience design principles, cyber resilience techniques, and (depending on the level of detail in which the architecture is defined) the potential applicability of implementation methods. How well a system’s capabilities cover at least one attacker activity can be represented by a heat map of threat or threat coverage. Through coverage evaluation, it is possible to visually compare the extent to which cyber resilience solutions can achieve and improve the system’s (priority weighted) cyber resilience objectives or sub-objectives, as shown in Figs. 4.15 and 4.16 [24].

The “System Structure Model—Comprehensive Resilience” (SSM-CR) is a customizable and relatively easy scoring method [25] to enable project managers to measure the network resilience of a given system. SSM-CR performs positioning or context adjustment in two ways. First, it reflects the priorities of stakeholders (i.e., which objectives, sub-objectives, and capabilities are important). Second, performance assessments (i.e., the extent to which prioritized capabilities are provided or how well prioritized activities are actually performed) are conducted based on established assumptions about the operational and threat environment. For any given system or project, cyber resilience objectives, sub-objectives, and activities must be customized or interpreted to explain their significance within the mission or service function of the system and its operating environment. Similarly, metrics related to cyber resilience design principles, techniques, and methods must be customized to reflect the system’s technical environment—its architecture and constituent technologies.

4.10.3 Cyber Resilience Attack and Defense Assessment

The RAND Corporation of the United States proposed an indicator framework for measuring cyber security and cyber resilience from the perspective of red-blue confrontation [26]. This framework can be used to measure the performance of US Air Force missions or systems in a confrontational network environment. These indicators can also indicate or influence the purchaser’s decision-making during the entire life cycle of the weapon system. The indicator framework features two types of network indicators: work-level indicators for confronting adversary cyber operations and institutional-level indicators for capturing organizational defects in the network. The work-level indicator framework focuses on blocking the red team’s network operations. Only by following the path of network attack can the red team successfully launch a cyber-attack. The attack path includes accessing the target system, obtaining the target information needed to launch the attack, developing the capabilities required to execute the attack, and identifying attack targets that play an important role in the execution of peace-time or wartime missions. These four activities are not necessarily carried out in sequence and can overlap in execution time. The blue side responds to the red side’s actions through a large number of complex countermeasures. Therefore, the indicator framework is mainly developed around four actions of the red side: access to the target, sufficient knowledge of the target information, the ability (resources) to execute the attack, and the impact of the red team’s actions on the mission of the blue side. In addition, the RAND Corporation proposes three levels of assessment methods: self-assessment, external assessment, and mission- and system-level rating. RAND also defines five assessment results: highly mature, mature, moderately mature, immature, and most immature. The institutional-level indicator framework is used to assess the problems at the institutional level of an organization. Organizations that have successfully avoided catastrophic failures can reduce the drift phenomenon (the difference between how

senior leaders think about the way things have happened and the actual way things have happened) by their members gathering, classifying and evaluating information, and then passing critical information to senior leaders outside the normal chain of command. Institutional-level indicator assessment also has the same five assessment results.

Since 2010, Netflix has developed a chaos experiment evaluation tool [27–29] to strengthen the resilience of its service systems deployed on Amazon’s public cloud AWS. Later, Microsoft and Alibaba used their own chaos evaluation tools to improve the security and resilience of their public clouds. Chaos Mesh, an open-source project, provides a cloud-native chaos engineering platform, which has the ability to conduct chaos testing on the Kubernetes platform and provide a variety of fault injection capabilities in a safe and controllable manner.

Chaos evaluation is a system evaluation method that actively introduces abnormal states (disturbances) of software/hardware into the system, creates fault scenarios, and determines optimization strategies based on the system’s behavior under various pressures. The purpose of chaos evaluation is to detect system risks in advance through experiments and solve system risks through architecture optimization and improvement of operation and maintenance models, thereby achieving a resilient architecture, improving fault immunity, and building confidence in the system’s ability to withstand out-of-control conditions in the production environment. Chaos evaluation emphasizes five principles: establishing a hypothesis around steady-state behavior; diversifying real-world events; running experiments in the production environment; continuously automating experiments; and minimizing the blast radius.

4.11 Challenges to and Shortcomings of Existing Cyber Resilience

4.11.1 Major Hard Flaws Exist in the Mindset of Avoiding Unknown Attacks

The core purpose of the existing cyber resilience engineering framework is to ensure the smooth execution of key missions and services. It emphasizes the importance to overcome abnormalities and trigger the recovery of services in a timely manner by actively sensing and detecting abnormalities (especially obvious abnormalities such as interruption). However, it pays insufficient attention to the hidden abnormality in services caused by unknown network attacks, even giving up resistance to imperceptible abnormalities.

For example, the first paper on cyber resilience engineering [19] emphasized completing the missions as required regardless of cyber-attacks. Although the paper also mentioned that there are many unknown unknowns in cyberspace, its main idea is to almost give up tracking unknown unknown attacks and mainly rely on network resilience to minimize the impact of cyber-attacks.

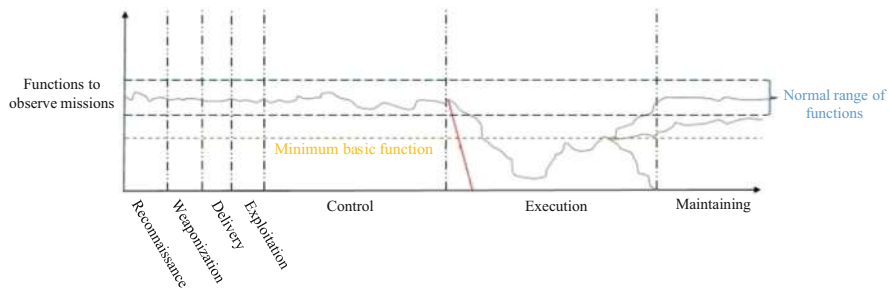


Fig. 4.17 Cyber resilience against destructive malware

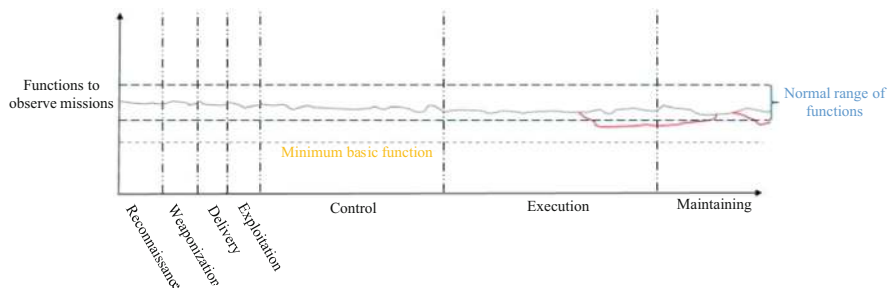


Fig. 4.18 Cyber resilience against data leakage or forgery

MITRE's 2017 paper on cyber resilience design guidelines [30] pointed out that when network adversaries perform activities at different stages of the cyber-attack's life cycle before taking obvious destructive actions, the attack may not be detected for a period of time (especially in the reconnaissance stage and weaponization stage in the early stage of the attack). If the attack focuses on penetration, the interference caused by the adversary may not occur. In the control phase, the adversary expands his knowledge of the system and the mission it supports, hide or delete the evidence of his activities, and gain control of more system components. Because stealth is an inherent feature of the adversary's plan, the adversary avoids causing damage (in fact, he may even take actions to improve system performance) or cause brief and minor disruptions to trick the performance and intrusion detection tools and redefine "normal." In the execution phase, the adversary instructs the malware to take actions (such as denial of service, corrupting data in a way that renders it useless, causing physical damage), which may include actively hindering recovery, as shown by the red line. Destructive malware can reduce the function of a component or system to zero, as shown in Fig. 4.17.

Figure 4.18 [30] illustrates the life cycle of a cyber-attack where an adversary targets data leakage or forgery or undetected usurpation capabilities (such as using an IoT device to launch a DDoS attack, without causing the owner/operator of the device to suspect that their device has been compromised). In this case, the adversary tries to avoid any damage. This may significantly delay the detection of adversary activities, preventing defenders from responding in time.

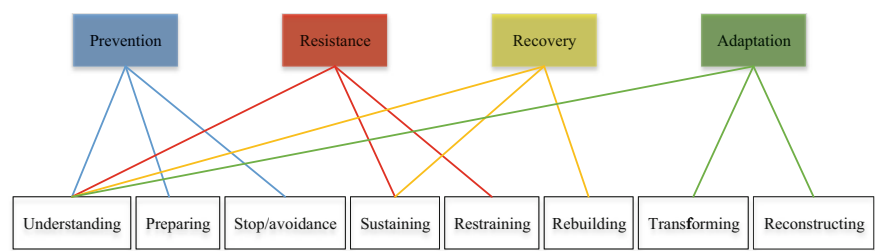


Fig. 4.19 Correspondence between the goals and objectives of the 2011 version of cyber resilience

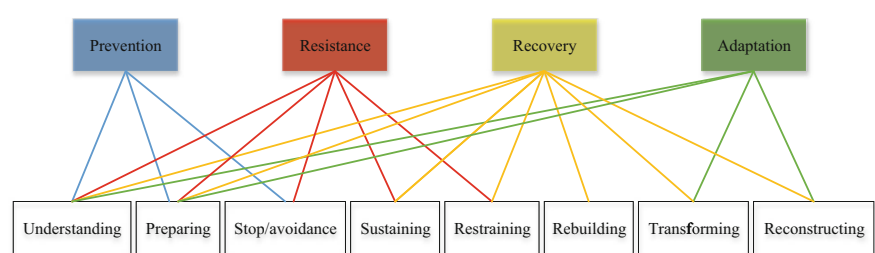


Fig. 4.20 Correspondence between the goals and objectives of the 2021 version of cyber resilience

Literature [31] pointed out that cyber resilience and cyber security need to work together. Most cyber resilience measures assume, exploit, or enhance various cybersecurity measures. Cybersecurity and cyber resilience measures are most effective when applied together in a balanced way. From the perspective of cyber resilience, modern systems are large and complex entities, so systems, operating environments, and supply chains inevitably have defects and weaknesses that adversaries can exploit. Therefore, achieving effective protection of network components often requires investments in both cyber security and network resilience. On the one hand, too much emphasis on traditional cybersecurity measures aimed at keeping adversaries out may leave systems without adequate cyber resilience to effectively counterattack and respond once adversaries gain a foothold. On the other hand, placing too much emphasis on cyber resilience investments can leave the security boundary full of loopholes, as a result of which there will be a larger attack surface; adversaries can gain and exploit a foothold to achieve immediate effects; and the cyber resilience measures may be overwhelmed by more and more low-level attacks.

Judging from the development history of cyber resilience over the past 10 years, the 2021 version (Fig. 4.20) of the cyber resilience engineering framework focuses more on service recovery compared with the 2011 version (Fig. 4.19).

This result-oriented perspective of over-reliance on tasks or business-specific recovery thinking after an incident has major flaws: First, it lacks careful attention to clues before the incident and misses the opportunity to “treat the disease before it happens”; second, there exist major hidden dangers in tasks or businesses which pretend to be normal without being detected; thirdly, in some cases, the interruption loss of a single key task or business is difficult to make up for.

What's more serious is that because existing cyber resilience research lacks technical methods to deal with "unknown unknown" network attacks, it cannot guarantee the security of key business system hosting platforms and further cannot guarantee the credibility of key business systems, let alone achieve the core goal of "mission assurance." In the existing cyber resilience framework, only the fourth strategic design principle "Assume Compromised Resources" contains the design strategic principles that should be adopted for unknown threats. Unfortunately, although this design principle clearly points out that there may be undetectable failures in the system, the suggestions it gives are limited to passive failure impact analysis and do not provide proactive response methods. In addition, in the tenth structural design principle "Adaptive Resource and Risk Management," there is a brief mention of "responding to unknown hazards through the unpredictable application of risk management strategies." In addition to the above two design principles, the existing cyber resilience framework does not mention how to deal with unknown threats (Fig. 4.20).

As pointed out in Chap. 1, in the digital age, software, hardware, algorithms, and protocols inevitably introduce unknown vulnerabilities and backdoors in many links of the entire industry chain such as design, development, processing, manufacturing, sales, application, and after-sales service. Especially with the popularization of isomorphic digital or information infrastructure, cyber-attacks based on the above-mentioned unknown vulnerabilities and backdoors are widespread and destructive, becoming the biggest "public hazards" in the development of the digital economy or digital society. An untrustworthy, continuously running critical service system may even cause greater harm than its downtime. For example, a banking business system that is infiltrated and controlled by hackers may cause serious consequences such as loss of key business data, customer data leakage, and loss of customer assets. Compared with a banking service system that goes down after being attacked, this can cause more serious consequences. For another example, a self-driving vehicle controlled by a hacker is at risk of becoming a "killing machine," which is more dangerous than a vehicle that cannot continue driving after a failure.

4.11.2 Major Hard Problems with Engineering Systems Lacking Architectural Guidance

To promote the development of the scientific basis of cybersecurity, the National Security Agency Research Directorate of the United States launched a Science of Security and Privacy Initiative (SoS) in 2011, sponsoring six universities in the United States to carry out basic scientific research on cyberspace security and safety. In 2012, the SoS Initiative released five hard problems in cyberspace [32], including scalability and contestability; policy-controlled security collaboration; security-metric-driven assessment/design/development/deployment; resilient system architecture; and understanding and interpretation of human behavior.

Table 4.21 Recommended priorities of cyber resilience techniques

Priority	Cyber resilience techniques
High priority	Analytical monitoring, coordinated protection, variety, privilege restriction, redundancy
Medium priority	Adaptive response, dynamic positioning, dynamic adjustment, non-persistence, segmentation, integrity verification
Low priority	Deception, readjustment, unpredictability

Among them, cyber resilience architecture as one of the five hard problems has gone through 10 years of research and development. Up to now, the research progress on this problem (2015 report [33] and 2022 report [34]) that can be retrieved from the SoS Initiative is still limited, and no obvious breakthrough has been made yet.

In fact, since the early days of cyber-resilient system development, exploring and building excellent cyber-resilient system architecture has always been a matter of great concern in the academic and industrial circles of cyber resilience engineering.

MITER’s first paper on cyber resilience [19] discussed this problem. The paper pointed out that the virtualization technology, as a shift in computing paradigm, has the agility needed to change the structure of a mission system in real time; can provide cost-effective redundancy, resource allocation, recovery, and security; can be used to realize the replication and scaling technology of isolation, non-persistence, and availability; and can provide diversity in hardware platforms, chipsets, operating systems, applications, and services, and randomness in deployment practices, so, it is the cornerstone of building a secure and resilient architecture.

The report to the Third Annual Secure and Resilient Cyber Architectures Workshop [35] made a conceptual prioritization on how to apply the most beneficial cyber resilience technology in the target system architecture. The report holds that ① all technologies will be useful and beneficial, but some may be more difficult to adopt than others; ② some of these technologies already represent good engineering practices; ③ some technologies will perform poorly if they are not clearly defined in the early stage of the system’s life cycle; ④ careful coordination is required between missions and enterprise communities in order to effectively incorporate cyber resilience; ⑤ costs and benefits must be balanced to determine the best matching technology and calculate cyber resilience return on investment. Based on these principles, the conceptual priorities of cyber resilience techniques are proposed, as shown in Table 4.21. (A few technologies were later adjusted in the 14 resilience techniques released by NIST).

In its cyber resilience industry guidance [36], MITRA proposed to strengthen system architecture design, security management, access control, equipment hardening, backup strategy, network cooperation, etc. Regarding structural design, the focus is on segmentation (physical or logical separation or isolation of resources based on trustworthiness and criticality), collaborative defense (managing various mechanisms in an adaptive and coordinated manner, mainly including defense in depth and defense in coordination), and diverse techniques (using a heterogeneous

set of hardware/software components and data sources) to thwart adversary attacks and limit the damage caused by malware.

When releasing the hard problems of resilient architecture, SoS emphasized that a recognized challenge is to develop a system architecture that can “tolerate” successful attacks on system components, including 1. resisting attacks; 2. continuing to provide basic services when attacked (possibly at a reduced level); 3. quickly restoring functionality after an attack.

SoS believes that candidate strategies include the following: 1. the principle of separating and minimizing privilege to allocate the least privilege to separated components and minimize the impact of compromise; 2. the principle of defense in depth through differentiated deployment of multiple layers of protection, so that a successful attack has to penetrate multiple levels of component protection; 3. tolerating intrusive protocols and algorithms through replication, voting, and methods from Byzantine fault tolerance.

SoS points out that more mature, reliable, and available analysis methods should be used to design and analyze an intrusion-tolerant system architecture with quantifiable service levels. However, there will be significant differences between the new methods and the existing reliable analysis methods. A pressing challenge is that unpredictable network intrusions affect system components in a completely different way from random hardware/software failures with statistical characteristics. The interaction between attack and defense is fertile ground for the application of the game theory. When system resilience is the primary objective, it is more necessary to carry out game theory research. It needs a substantial breakthrough to combine the abstract and idealized assumptions of the game theory with the uncertainty and complex details of the operating system. The application of various well-conceived defense principles (privilege separation, defense in depth, and diversity) and mechanisms in real systems needs more tests. For example, privilege separation is not used in most cases where performance is to be improved. Similarly, software diversity may be at the expense of performance and deployability, so that it is urgent to incorporate these techniques into real systems without exceeding the linear performance overhead.

Due to the lack of a dominant cyber resilience system architecture, the current cyber resilience engineering is like “opening a TCM shop,” where progress has been unsatisfactory despite rich medicinal materials.

In July, 2021, IBM [37] conducted its sixth annual cyber resilience survey among more than 3600 IT and security professionals in North America, South America, Europe, Asia, and Australia. According to the survey, 67% of the respondents said that the number and severity of cyber security incidents had increased or increased significantly over the previous 12 months; 51% said they had suffered data leakage in the previous 12 months; and 46% reported that they had suffered at least one ransomware attack in the previous 2 years.

As for why cyber resilience had not improved, many interviewees believed that too many tools are an important reason. As shown in Fig. 4.21, 30 percent of the respondents stated that their organization had deployed more than 50 kinds of security tools and technologies, but only 30% of them said their organization had the right mix of security tools (Fig. 4.22).

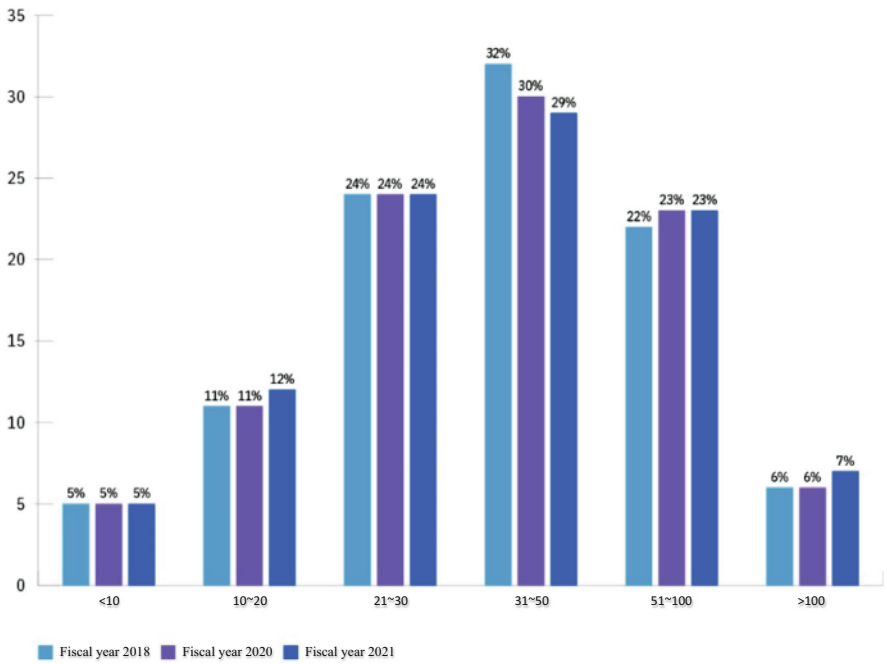


Fig. 4.21 The previous three surveys revealed that organizations had deployed a variety of security tools and technologies

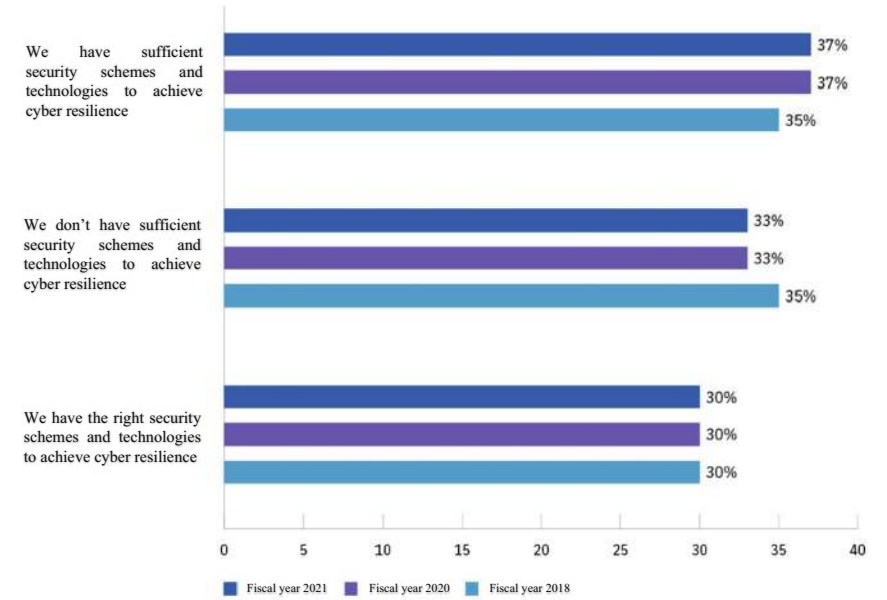


Fig. 4.22 Only 30% of respondents said their organization had the right mix of security tools

4.11.3 Significant Hard Challenges in Evaluating the Lack of Core Competence Measurement

As mentioned above, cyber security, metric-driven assessment, design, development, and deployment are one of the five hard problems of cyber security released by the National Security Agency Research Directorate in the Science of Security and Privacy Initiative (SoS). This problem is also closely related to the hard problem of a cyber resilience architecture, because the evaluation of architecture resilience depends in part on the research of metrics that quantify that resilience.

MITER and NIST always attach great importance to cyber resilience assessment and measurement and have made positive progress in high-level qualitative assessment, qualitative and semi-quantitative assessment (such as cyber resilience overlay maps), and detailed quantitative scoring assessment. However, their 2018 document [24] holds that there are three major challenges in the existing cyber resilience analysis and assessment in verifying and measuring the core competence of system cyber resilience, i.e., there are still major defects such as insufficient persuasiveness and feasibility in clearly proving whether a system has good cyber resilience, comparing the cyber resilience of one system with that of another, and clarifying how to detect a system's core competence in cyber resilience.

1. Challenge 1: The Relationship Between the Complexity of Assessment and the Selection of Core Competence Indicators

Generally speaking, it is often considered feasible to use core competence indicators to evaluate resilience or cyber security. Similarly, some studies also recommend using the expected availability of the service capabilities required by the system as a core competence indicator to evaluate cyber resilience. However, it should be noted that additional metrics need to be specified to evaluate other aspects. Generally, to support engineering decisions, any single metric will either mask the complexity of the problem domain or require a large number of input measurements that may vary widely in quality (e.g., timeliness, accuracy), and thus the results will be highly uncertain. In order to treat complexity fairly, the formulas and models that produce core competence indicators should represent a large number of potential unfavorable conditions and consequences. Complex formulas and models provide value to the topics of discussion between stakeholders and engineers to clarify what assumptions are important. However, obtaining high-quality (e.g., timely, consistent) information at a reasonable cost is a major challenge. In addition, the threat model may not represent the actual adversary and may be based on the old information of the adversary's TTP or on information of the adversary's intentionally manipulated TTP.

2. Challenge 2: Assessment of the Comparability of Measurement Results

There are also significant challenges in assessing the comparability of measurement results. In organizations with similar missions or service functions, facing common threats and applying similar standards of good practice, if these indicators are evaluated in a consistent way throughout the organization, then the

comparison of organizational indicators can be meaningful and informative. However, this consistency is often difficult to achieve (or prove). Different missions (and the system or the system of multiple systems supporting those missions) face different threats and are executed in different operating environments. Therefore, metrics that are meaningful and useful for one mission may not be meaningful or evaluable in the context of another mission. Metrics that are tracked and compared over a long period of time for the same organization, mission, service function, or system may help identify trends. Beyond this, comparisons become more challenging and must be placed within a common threat and operating environment to be meaningful.

There may be great differences between cybersecurity and cyber resilience indicators in the level of detail in defining a threat model or stating assumptions about a threat. Similarly, indicators vary depending on the assumptions about the operating environment and the level of detail with which the operating environment is represented. In other words, metric values are sensitive to the context in which the metric is evaluated. Metrics evaluated in different contexts, even if defined in the same way, may not be comparable.

3. **Challenge 3: How to Evaluate Competitiveness and Emergence**

Security, resilience, safety, and cyber resilience are all emergent attributes of the system. While resilience can be an attribute of a device or platform, new attributes and behaviors may emerge when system elements are assembled into increasingly complex systems or systems of systems: “Emergence and complexity refer to the emergence of advanced attributes and behaviors of a system that apparently arises from the collective dynamics of the system’s components. These properties cannot be directly deduced from the low-level motion of the system. Emergence is an attribute of the ‘whole’ and is not owned by any individual part that constitutes the whole.”

All types of attribute property measurement are faced with huge challenges in how to do it and to what extent it can be done. Whether combining the measures of attributes or behaviors of subsystems or system elements, or deriving the measures of larger systems or systems of systems by other means, there is a methodological dilemma that low-level, low-dimensional eigenvalues are difficult to describe high-level, high-dimensional eigenvalues.

The emergence of emergent situations brings challenges to the definition of cyber resilience metrics. “Emergent attributes are often qualitative in essence, subjective in nature and evaluation, and require consensus based on evidence analysis and reasoning.” Cyber resilience measurement needs to face the overall presentation of cyber resilience architecture as an evaluation of emergent attribute.

The relationship between observable values, atomic measurement values, derivative values, and the subjective evaluation they support must be well defined. Like security, cyber resilience is an emergent attribute of a system in its operational and threat environment—the overall behavior of a system depends on the behavior of its users, operators, adversaries, and defenders as part of it. Therefore, the definition of cyber resilience measurement needs to determine

the assumptions about these environmental characteristics, and the combination of individual indicators is meaningful only when the environmental assumptions are consistent.

4.12 Cyber Resilience Engineering and Manufacturing-Side Network Security

Cyber resilience engineering belongs to the field of network security on the manufacturing side. The so-called network security on the manufacturing side refers to developing digital products with default ESS capabilities by designing security methods, establishing a digital ecosystem with inherent resilience (toughness), and promoting the transfer of security responsibility from the user side of digital products to the supply side. With the rapid development of digital economy and the rapid advancement of digital transformation of all industries, the normal operation of the human society is inseparable from information infrastructure; and various key infrastructures including finance, transportation, energy, water conservancy, and national defense are also carried by a digital base. The traditional “two-skin” model of network security and information technologies can no longer meet the requirements of the digital age, and it has become the basic consensus of advanced countries around the world to vigorously develop manufacturing-side security.

In the National Cybersecurity Strategy issued by the White House in 2023, it was pointed out that the existing underlying architecture that drives the digital ecosystem in the United States has serious defects, and it is necessary to fundamentally change the underlying driver of the digital ecosystem to the dominant defensive side [39]. In July 2023, the White House issued the National Cybersecurity Strategy Implementation Plan, which proposed the principles and practices to promote the development of security-by-design and security-by-default, and designated a special department to be responsible for the implementation and set time nodes to keep dynamic updates every year. As a response to the cybersecurity strategy, CISA, FBI, and other federal agencies, together with the relevant security departments of Australia, Britain, Canada, Germany, the Netherlands, and New Zealand, jointly issued *Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design* [40], which mainly provides guidance and suggestions on the security design and configuration of software products. The document points out that only by combining safe design practices can the vicious circle of repeatedly repairing the programs be broken. In October, 2023, CISA released an updated version of this document [41], with eight new participating agencies from other countries such as Czech Republic, Norway, Israel, Singapore, South Korea, and Japan, further explaining the safe development of software.

Cyber-informed engineering (CIE) promoted by the US Department of Energy is a manufacturing-side security project in the field of energy infrastructure. Idaho National Laboratory (INL) under the US Department of Energy put forward CIE [42]

in 2015 and released CIE Technical Report [43] in 2017. In addition, INL has issued a series of technical reports to explain how to apply and implement CIE. The theory of CIE believes that “security-by-design” is necessary and extends the concept of security-by-design from traditional information technologies and software engineering to cyber-physical systems engineering. CIE believes that cybersecurity considerations should be introduced into security-by-design, and engineers can solve these problems in the early stage of system engineering. The focus of CIE is to help engineers who traditionally conceive, plan, design, implement, and operate systems with digital elements, understand network risks (instead of becoming network experts), and use the professional knowledge of network security experts to minimize network risks in the life cycle of various energy systems, thus achieving security-by-design.

The EU also attaches great importance to the manufacturing-side security of digital products and actively promotes it through policies and legislation. At the end of 2020, the European Union issued *The EU's Cybersecurity Strategy for the Digital Decade* [44], pointing out that the key infrastructure and basic services of the European Union are increasingly interdependent and digitized and requiring all its networked devices to be secure by design and have cyber resilience. Since 2020, the Council of the European Union has been planning legislation to require manufacturers of digital products to implement cyber resilience design. Its legislative logic is manufacturers of digital products lack the motivation to take security issues seriously, and without the appropriate intervention of policy makers, the market cannot cope with the rising cybersecurity risks. In September, 2022, the European Union published the draft of *Cyber Resilience Act*, which indicated that cyber resilience/security-by-design would be upgraded from “optional” to “mandatory” and become a rigid indicator to judge whether digital products are qualified or not [38]. In 2020, the Joint Research Centre (JRC) of the European Commission published the research report “Cybersecurity, our digital anchor (European Perspective)” [45], which summarized the development of cybersecurity in Europe in the past 40 years and pointed out that “Cybersecurity is no longer a technical choice, but a societal need.” From the perspective of cybersecurity economics, the report expounds the necessity of security-by-design on the manufacturing side of digital products and points out that although the practice of security-by-design increases costs, it will also reduce the subsequent harmful effects of digital products, and in the long run, it can still reduce the maintenance and repair costs of manufacturers and the use costs of end users throughout their life cycle.

Manufacturing-side security represents the latest development of global cyberspace security in the digital age. Cyber resilience engineering is a concrete practice to implement the vision of manufacturing-side security. The former points out the goal vision, while the latter tries to point out the way to achieve it. Manufacturing-side security emphasizes that digital product manufacturers should take the responsibility for security and provide digital products with default ESS capabilities through security-by-design, instead of only relying on “plug-in” defense means to obtain security capabilities. The goals and visions of manufacturing-side security and ESS are highly consistent. Since it was put forward in 2013, ESS has always

adhered to the methodology of “structure determines security” and given information systems default ESS capabilities through innovative system architecture design and collaborative application of related security technologies and methods under the traction of system architecture. As pointed out in Sect. 4.9, the current cyber resilience engineering is difficult to realize the vision of manufacturing-side security due to a number of major defects, but ESS-empowered cyber resilience engineering can effectively eliminate the above defects, provide strong support for building a secure and resilient digital base, and enable manufacturing-side security to take root in the digital age.

4.13 Summary of This Chapter

This chapter first introduces the development history, basic concepts, and basic connotations of functional safety, as well as functional safety integrity technologies such as hardware safety integrity and system safety integrity, and random failure and system failure analysis technologies. Then, it expounds the important development trend of functional safety from the perspectives of the extension of functional safety in various industries, the technical complexity of safety-related systems, the functional safety problems caused by cyber-attacks, the law of unity of opposites in the practice of functional safety, and resilience engineering. It analyzes the overall, multi-level, and multi-dimensional challenges faced by safety-related systems in the empowerment and transformation of digitalization, Internet, and intelligence and the technical challenges from the aspects of quality crisis brought by software expansion, GFS problems, and uncertainty brought by AI technologies.

Subsequently, this chapter briefly introduces the basic concepts, engineering framework, analysis, and evaluation of cyber resilience evolved from functional safety resilience engineering. Cyber resilience is oriented to confrontational and non-confrontational threats from networks and non-networks, focusing on the security of mission or service functions under threat conditions such as advanced persistent network attacks, expecting the system to have the ability of prevention, resistance, recovery, and adaptation. To this end, it puts forward 8 cyber resilience objectives, 5 cyber resilience strategic principles, 14 cyber resilience design principles, 14 cyber resilience support technologies, and more than 50 cyber resilience implementation methods and proposes qualitative analysis, semi-quantitative and quantitative measurement, and attack-defense confrontation assessment, forming a basic cyber resilience engineering framework.

Finally, this chapter summarizes and analyzes the challenges and shortcomings of current cyber resilience in terms of theoretical perspective, system architecture, evaluation, measurement, and analysis. These challenges and shortcomings indicate that cyber resilience engineering still needs fundamental breakthroughs and innovations in theoretical frameworks, methodologies, and practical norms. Recently, the European Union issued a Cyber Resilience Act [38], and the United States issued a National Cybersecurity Strategy [39], both of which emphasized the development of

manufacturing-side security, requiring the producers—not just the end users—of digital products to take responsibility for security. The current cyber resilience engineering tries to provide a method to realize manufacturing-side security, but unfortunately, due to some major defects, the current cyber resilience engineering is still not enough to support the realization of the goal and vision of manufacturing-side security. How to break through the limitations of the current cyber resilience engineering and build a safe, secure, and resilient digital infrastructure is a major problem to be solved urgently in the current digital age.

References

1. German Institute for Standardization. DIN V 19250: control technology – fundamental safety aspects to be considered for measurement and control equipment [S]. Berlin: Institute for Standardization (DIN); 1994.
2. The International Society of Automation. ANSI/ISA S84. 01-1996: application of safety instrumented systems for the process industries [S]. North Carolina: The International Society of Automation (ISA); 1996.
3. International Electrotechnical Commission. International standard IEC 61508: functional safety of electrical/electronic/programmable electronic safety related systems [S]. Geneva: International Electrotechnical Commission; 2000.
4. Cybersecurity and Infrastructure Security Agency (CISA). 2023–2025 strategic plan [EB/OL]. 2022. https://www.cisa.gov/sites/default/files/publications/StrategicPlan_20220912-V2_508c.pdf
5. State Administration for Market Supervision. GB/T 8223–1987: basic terminology and general working procedures of value engineering [S]. Beijing: National Bureau of Standards; 1987.
6. Miles LD. Techniques of value analysis and engineering [M]. New York: McGraw-Hill; 1972.
7. Burns A, McDermid J, Dobson J. On the meaning of safety and security [J]. Comput J. 1992;35(1):3–15.
8. Firesmith DG. Common concepts underlying safety security and survivability engineering [R]. Pittsburgh: Carnegie Mellon University; 2003.
9. Standardization Administration of China. GB/T 2900.99-2016 electrotechnical terminology-dependability [S]. Beijing: National Technical Committee for Standardization of Electrotechnical Terminology; 2016.
10. Baike. Functional safety [EB/OL]. 2022. <https://baike.baidu.com/item/%E5%8A%9F%E8%83%BD%E5%AE%-89%E5%85%A8/6936558>
11. Health and Safety Executive (HSE). Out of control: why control systems go wrong and how to prevent failure [R/OL]. 2003. <https://www.hse.gov.uk/pUbns/priced/hsg238.pdf>
12. Xin Z, Zouqing M, Xinyun Z, et al. Visualization analysis of research hotspot and evolution paths of functional safety [J]. China Instrum. 2021;3:21–6.
13. International Organization for Standardization. ISO 26262: road vehicles-function safety [S]. Geneva: International Organization for Standardization; 2011.
14. Hollnagel E, Woods DD, Leveson N. Resilience engineering: concepts and precepts [M]. Aldershot: Ashgate Publishing Ltd; 2006.
15. Holling CS. Resilience and stability of ecological systems [J]. Annu Rev Ecol Syst. 1973;4:1–23.
16. Bhamra R, Dani S, Burnard K. Resilience: the concept, a literature review and future directions [J]. Int J Prod Res. 2011;49(18):5375–93.
17. Madni AM, Jackson S. Towards a conceptual framework for resilience engineering [J]. IEEE Syst J. 2009;3(2):181–91.

18. Sterbenz JPG, Hutchison D, Çetinkaya EK, et al. Resilience and survivability in communication networks: strategies, principles, and survey of disciplines [J]. *Comput Netw.* 2010;54(8): 1245–65.
19. Goldman HG. Building secure, resilient architectures for cyber mission assurance [R]. McLean: MITRE Corporation; 2010.
20. Bodeau DJ, Graubart R, Picciotto J, et al. Cyber resiliency engineering framework [R]. Bedford: MITRE Corporation; 2011.
21. Ross R, Pillitteri V, Graubart R, et al. Developing cyber-resilient systems: a systems security engineering approach. NIST Special Publication 800-160, vol. 2, rev. 1 [EB/OL]. 2021. <https://doi.org/10.6028/NIST.SP.800-160v2r1>.
22. Standardization Administration of China. GB/T 22239-2019: information security technology – baseline for classified protection of cybersecurity [S]. Beijing: National Information Security Standardization Technical Committee; 2019.
23. Bodeau D, Graubart R. Cyber resiliency assessment: enabling architectural improvement [R]. Bedford: MITRE Corporation; 2013.
24. Bodeau DJ, Graubart RD, McQuaid RM, et al. Cyber resiliency metrics, measures of effectiveness, and scoring: enabling systems engineers and program managers to select the most useful assessment methods [R]. Bedford: MITRE Corporation; 2018.
25. Bodeau D, Graubart R. Structured cyber resiliency analysis methodology (SCRAM) [R]. McLean: MITRE Corporation; 2016.
26. Snyder D, Mayer LA, Weichenberg G, et al. Measuring cybersecurity and cyber resiliency [R]. Santa Monica: Rand Arroyo Center; 2020.
27. Netflix. Chaos monkey [EB/OL]. 2017. <https://github.com/Netflix/chaosmonkey>
28. Netflix. Simian army [EB/OL]. 2012. <https://github.com/Netflix/SimianArmy>
29. Blohowiak A, Basiri A, Hochstein L, et al. A platform for automating chaos experiments [C]. 2016 IEEE international symposium on software reliability engineering workshops (ISSREW), Ottawa, 2016:5–8.
30. Bodeau D, Graubart R. Cyber resiliency design principles: selective use throughout the lifecycle and in conjunction with related disciplines [R]. Bedford: MITRE Corporation; 2017.
31. The MITRE Corporation. Fifth annual secure and resilient cyber architectures invitational [R/OL]. 2015. <http://www2.MITRE.org/public/sr/2015-Secure-and-Resilient-Cyber-Architectures-Report-16-1199.pdf>
32. Nicol DM, Sanders WH, Scherlis WL, et al. Science of security hard problems: a lablet perspective [EB/OL]. 2012. <http://cps-vo.org/node/6394>
33. Nicol DM, Sanders WH, Katz J, et al. Science of security lablet progress on the hard problems [EB/OL]. 2015. <http://cps-vo.org/node/21590>
34. Science of Security and Privacy. Science of security and privacy 2022 annual report [R/OL]. 2022. <https://cps-vo.org/group/sos/annualreport2022>
35. The MITRE Corporation. Third annual secure and resilient cyber architectures workshop [R/OL]. 2013. <https://www.MITRE.org/news-insights/publication/third-annual-secure-and-resilient-cyber-architectures-workshop>
36. The MITRE Corporation. Architect to protect: creating a foundation for resiliency [R/OL]. 2015. <http://www2.MITRE.org/public/industry-perspective/documents/01-ex-architect-protect.pdf>
37. The IBM Corporation. Cyber resilient organization study 2021 [EB/OL]. 2021. <https://www.ibm.com/resources/guides/cyber-resilient-organization-study>
38. European Union. Cyber resilience act [EB/OL]. 2022. <https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act>
39. The White House. National cybersecurity strategy [EB/OL]. 2023. <https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf>
40. Cybersecurity and infrastructure security agency, shifting the balance of cybersecurity risk: principles and approaches for security-by-design. 2023. https://www.cisa.gov/sites/default/files/2023-06/principles_approaches_for_security-by-design-default_508c.pdf

41. Cybersecurity and infrastructure security agency, shifting the balance of cybersecurity risk: principles and approaches for secure by design software. 2023.
42. Anderson R, Price J. Cyber-informed engineering: the need for a new risk informed and design methodology [EB/OL]. 2015. <https://www.osti.gov/biblio/1236850>
43. Anderson R, Benjamin J, Wright VL, et al. Cyber-informed engineering [EB/OL]. 2017. <https://www.osti.gov/biblio/1369373>
44. Joint Communication to the European Parliament and the Council. The EU's cybersecurity strategy for the digital decade. 2020.
45. NaiFovino I, Barry G, Chaudron S, et al. Cybersecurity, our digital anchor. EUR 30276 EN, Publications Office of the European Union, Luxembourg; 2020. ISBN 978-92-76-19957-1. <https://doi.org/10.2760/352218>, JRC121051. https://www.cisa.gov/sites/default/files/2023-10/SecureByDesign_1025_508c.pdf

Chapter 5

The Framework of Cyber Resilience Engineering Empowered by Endogenous Security and Safety



5.1 Overview of Cyber Resilience Empowered by Endogenous Security and Safety

As mentioned in Chap. 4, there are many challenges and shortcomings in the current cyber resilience development, including major hard flaws in the thinking of cyber resilience that focuses on post-event recovery processing, major problems in the cyber resilience engineering system that lacks a dominant architectural guidance, and major challenges in the analysis and evaluation of cyber resilience that lacks core competence measurement.

This section expounds the main advantages of ESS-empowered cyber resilience from four aspects: the expansion and deepening of thinking perspectives; the support of theoretical foundation; the innovation of the system architecture; highly efficient and trustworthy analysis and evaluation.

5.1.1 *New Thinking Perspectives*

According to NIST's latest authoritative literature [1] on cyber resilience, all discussions on cyber resilience can be distinguished by their focus and prior threat assumptions. Existing mainstream cyber resilience can be summarized as resilience in a cyber threat environment, mainly focusing on mission/service functions and the impact of advanced persistent threats (APT). The main threat assumptions include assuming that the environment is constantly changing and that adversaries will compromise or destroy the system or organization, assuming that the adversary will maintain a presence in the system or organization, etc. In the author's view, truly effective cyber resilience should not only focus on the impact of advanced persistent threats but should also face "unknown unknown" cyber-attacks; empower

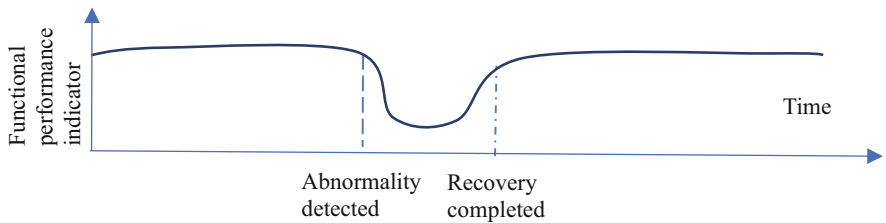


Fig. 5.1 Evolution from damage to and recovery of traditional cyber resilience functions

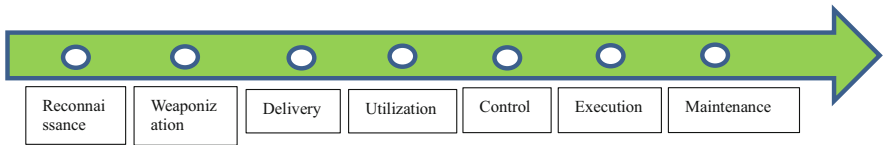


Fig. 5.2 A typical cyber-attack chain

the detection, discovery, and early containment of advanced persistent threats; and ensure cyber resilience by eliminating the root causes.

Cyber resilience engineering not only aims on conventional network threats but also focuses on “advanced persistent threats” (unconventional threats) on the basis of existing traditional cyber security protection (usually plug-in security protection against known cyber threats), especially advanced network threats that can penetrate and break through perimeter defense, access control, and privilege management mechanisms and intrusion detection mechanisms. These threats may persist on the target system for a long time, which may not only penetrate and destroy data but also reduce or destroy the performance of missions or systems.

However, the basic thinking perspective of existing cyber resilience engineering is that APT is difficult to detect, so people can only try their best to prevent it in advance (such as reducing the attack surface, making segmentation, intensifying access inspection, etc.), patch the vulnerabilities and recover service as soon as possible afterward (by redundant replacement and reorganization, transfer, etc.), and strengthen the sensitivity to and comprehensive monitoring of the abnormalities of mission and service functions and performance indicators during the execution of missions and services, so as to trigger the abnormality handling and recovery process as soon as possible. The typical expectation and vision of cyber resilience engineering are shown in Fig. 5.1, which indicates that the system starts the repair process in a timely manner after monitoring a decline of functional performance indicators, so that missions and services can be continued.

In order to better describe the process of APT attacks, a typical cyber-attack chain covering each attack stage is illustrated in Fig. 5.2.

Current cyber resilience engineering lacks the ability to detect APT attacks. Therefore, the system cannot detect cyber-attacks in the early stages of an attack, such as reconnaissance, weaponization, delivery, utilization, and control, i.e., it is

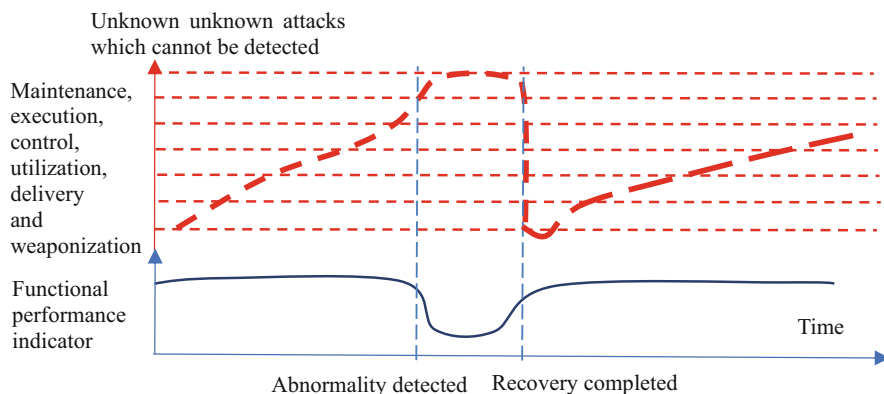


Fig. 5.3 Existing cyber resilience lacks early awareness of APT attacks

insensitive to planned attacks, as shown in Figs. 4.17 and 4.18 in Chap. 4. It can only rely on the monitoring results of explicit functional performance indicators of the system's mission and service operations and indirectly infer a network attack in the operation and maintenance stages when discovering system degradation or failure. This indirect detection of network attacks lags far behind the adversary's progress in launching network attacks. It may have a certain defensive effect against distributed DoS attacks, but it is often ineffective for defending against hidden or covert hijacking, tampering, or other “unknown unknown” attacks.

In order to more intuitively display the deficiency of the existing cyber resilience, the progress annotation of network attacks can be added to the typical cyber resilience evolution diagram shown in Fig. 5.1, so that the diagram not only shows the evolution of functional performance but also reveals the evolution of network attacks, as shown in Fig. 5.3. The method shown in Fig. 5.3 is an innovation in itself, and the red part of the diagram shows the stages of evolution of the network attack process.

Figure 5.3 shows that network attacks are the “cause” and abnormal performance of mission/service functions is the “result.” The red dotted line in the upper part shows that network attacks go through the latent process of reconnaissance, weaponization, delivery, utilization, and control, followed by the eruption at the execution stage, which results in abnormal performance of corresponding mission/service functions, triggering cyber resilience intervention and recovery mechanisms. After the performance of mission/service functions is restored, the maintenance phase of the last wave of network attacks may come to an end, but the next wave of attacks may begin to brew in a more subtle and evolutionary way.

In order to break this endlessly recurring cycle and realize the grand vision of cyber resilience engineering, we must change the mere dependence on mission/service function performance monitoring to both detecting “unknown unknown” attacks and monitoring mission/service function performance and change the mere passive resilient recovery after the attack to endogenous resilience of curbing bad

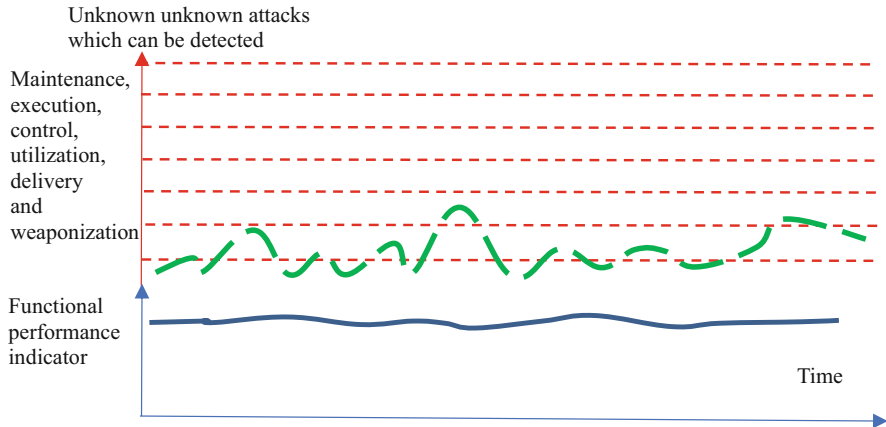


Fig. 5.4 Evolution of cyber resilience with the ability to detect and defend against “unknown unknown” attacks

signs/early hidden dangers of attack (curbing and eliminating network attacks before they produce a destructive effect). This empowerment can transform the imperceptible red attack curve in Fig. 5.3 into the perceptible green attack curve in Fig. 5.4.

It can be seen clearly from Fig. 5.4 that the adversary’s multiple attempts of an APT attack, represented by the green curve, are suppressed in the early stages of the attack (which greatly deters the adversary from launching subsequent attacks), and the system’s functional performance, represented by the blue curve, remains stable without big ups and downs, better avoiding abnormalities in the performance of the system’s mission and service functions.

5.1.2 New Theoretical Basis

Cyber resilience engineering is part of mission assurance engineering, focusing on ensuring the continued operation of critical services under various adverse conditions. At the same time, cyber resilience engineering is based on system security engineering and draws on concepts and technologies from related fields such as reliability, survivability, fault tolerance, resilience engineering, cyber resilience, critical infrastructure system resilience, emergency support, etc. The relationship between cyber resilience engineering and other disciplines is shown in Fig. 5.5.

Cyber resilience engineering first borrows some technologies and methods such as adaptive response, analytical monitoring, coordinated protection, privilege restriction, redundancy, and segmentation from the disciplines of cybersecurity, resilience engineering, and operation continuity, but these technologies and methods need to be implemented in a completely new way. Cyber resilience engineering also borrows some techniques and methods such as situational awareness, variety,

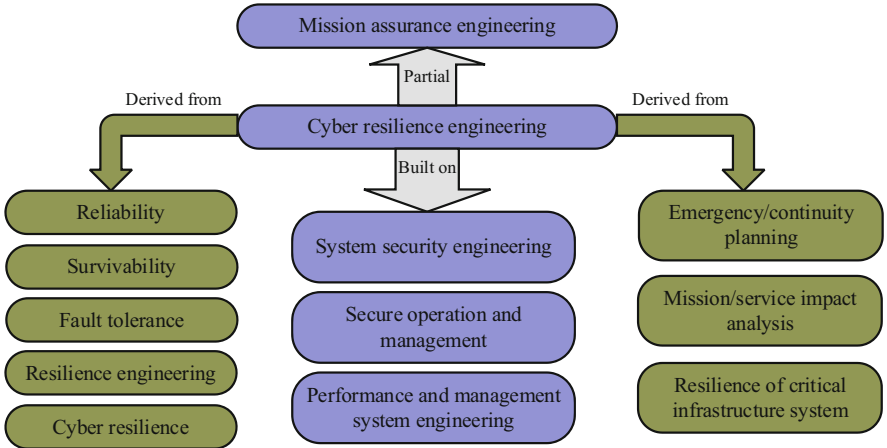


Fig. 5.5 Relationship between cyber resilience engineering and related disciplines

Security design principles		Design principles of resilience engineering	
Modularization and layering	Defense in depth	Redundant functions	Local capacity
Minimize common mechanisms	Isolation	Layered defense	Manual backup
Safe evolution	Privilege restriction	Avoid complexity	Reorganize
Evolution design principles		Design principles of survivability engineering	
Imitation	Redundancy	Mobility	Edge
Decentralization	Edge	Hiding	Distributed
Modularization of objectives	Relaxation	Redundancy	Evolution

Fig. 5.6 Typical design principles in disciplines related to cyber resilience engineering

non-persistence, readjustment, and integrity verification from disciplines (e.g., security, reliability) that deal with non-adversarial threats, but these techniques and methods need to be modified for APT threats, etc. Finally, cyber resilience engineering also borrows some techniques and methods such as deception, dynamic positioning, and unpredictability from disciplines (e.g., medicine, military/defense, sports) that deal with adversarial threats, but these originally non-networked techniques and methods need to be transformed through networking. Because these derived adversarial techniques and methods do not have the genes of traditional disciplines such as security, resilience engineering, or operation continuity, it may be more challenging to integrate them into existing systems.

The design principles of cyber resilience engineering obviously also need to draw on the design principles of several related disciplines, as shown in Fig. 5.6. This diagram illustrates that different disciplines often share some design principles. For example, resilience engineering, survivability, and evolution all share the principle of redundancy; modularization and layering are security design principles; and layered defense is a resilience engineering design principle. However, these

apparently common design principles do not necessarily mean the same thing, and in terms of design principles, each discipline has its own assumptions, systems, risk models, and priorities. Therefore, the relationship between apparently identical or similar design principles from different disciplines needs to be deliberated. How to combine design principles from different disciplines into universal principles of cyber resilience needs more careful research.

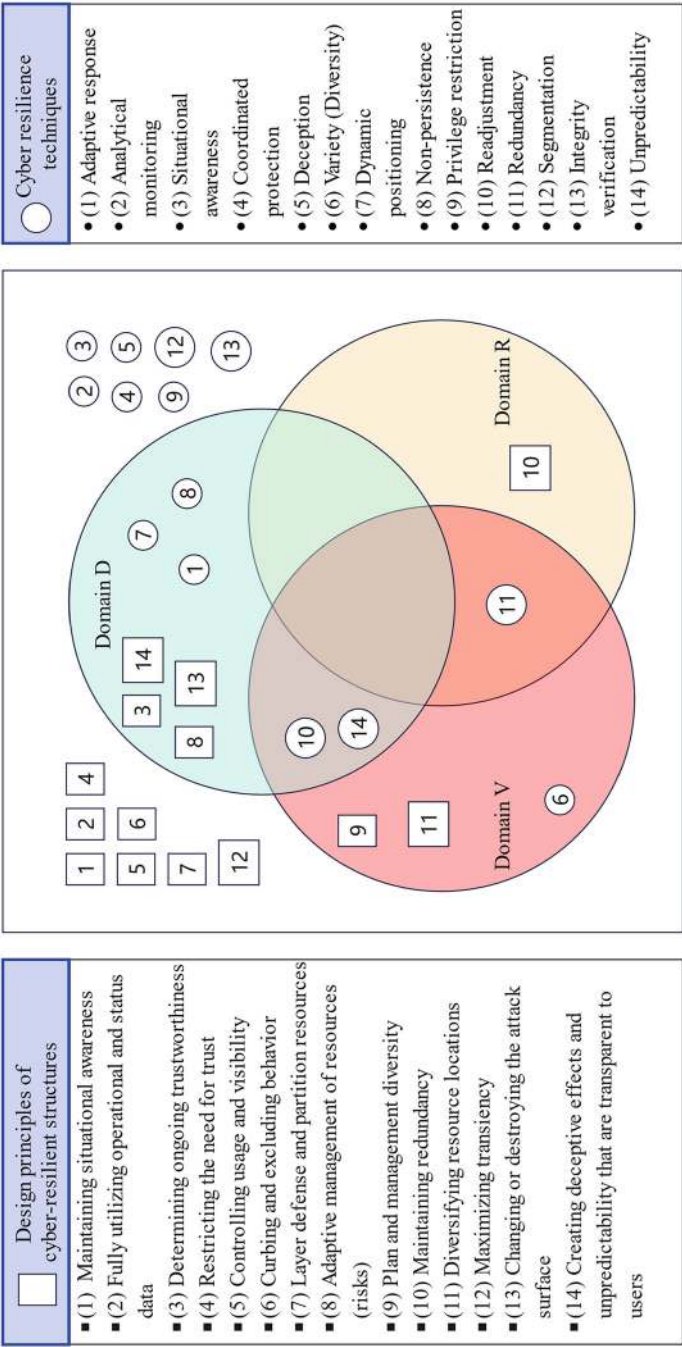
Fortunately, the basic theoretical analysis results of ESS in Chap. 2 provide theoretical guidance for supporting the interaction of the above-mentioned design principles of cyber resilience engineering and the re-adaptation and re-innovation of related technologies and methods in various disciplines.

The incomplete intersection principle (IIP) mentioned in Chap. 2 states that it is impossible to protect against unknown security threats in cyberspace without relying on prior knowledge as long as there is an incomplete DVR intersection. Figure 5.7 lists the mapping results of the 14 technologies and 14 design principles given by NIST in the DVR domain. As can be seen from Fig. 5.7, none of the design principles and technologies in the existing cyber resilience framework falls within the DVR intersection. In fact, most of the design principles and technologies fall out of the DVR intersection (including the two-two intersection and the three-three intersection). Only two technologies, “re-adjustment” and “unpredictability,” are located in the set DAV/R, and the “redundancy” technology is located in the set VAR/D, as shown in Fig. 5.7. Obviously, although the current cyber resilience framework provides a number of design principles and technologies involving dynamics, diversity, and redundancy, respectively, the existing cyber resilience framework is not able to solve the problem of unknown security threats in cyberspace due to the lack of architectures and techniques that organically integrate dynamics, diversity, and redundancy.

Furthermore, the DVR complete intersection principle (CIP) mentioned in Chap. 3 states that all unknown security threats in a cyber-physical system, regardless of whether they originate from random or uncertain perturbations or from man-made or non-man-made disturbances, can be expressed as differential-mode or common-mode problems in the DVR domain.

The ESS and safety existence theorem (ESS-existence theorem, ESS-ET) derived from this show that if a structure or algorithm possesses a fully intersecting expression of dynamics (D), variety (V), and redundancy (R) at the same time, it can effectively deal with differential-mode attacks based on any unknown vulnerabilities, backdoors, and virus Trojans within the construction, as well as functional failures caused by differential-mode disturbances triggered by random or uncertain factors, even in the absence of prior knowledge.

In summary, the ESS-related research results mentioned in Chap. 2 provide theoretical guidance from both positive and negative aspects for the construction of a young cyber resilience engineering discipline by drawing on the theories and technologies of other disciplines. As shown in Fig. 5.8, it can be said that the “arrow” of DVR empowers the “target” of cyber resilience engineering and lays a solid theoretical foundation for cyber resilience engineering!



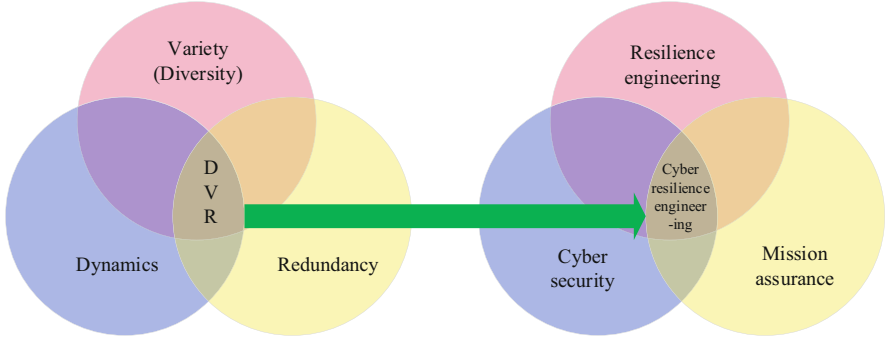


Fig. 5.8 The “arrow” of DVR empowers the “target” of cyber resilience engineering

5.1.3 New System Architecture

According to the discussion in Chap. 3, under the guidance of DVR complete intersection principle (CIP) and ESS-ET, a new DHR architecture is invented by introducing the SR-FC system and mechanism on the basis of DRS, which can not only transform the complex ESS problems into probabilistic differential-mode or common-mode problems in the DVR domain but also effectively suppress “known unknown” or “unknown unknown” cyber threats or attacks. The holistic “four-in-one” approach meets the core connotation of the four goals of cyber resilience engineering, prevention, resistance, recovery, and adaptation, and provides a leading, fundamental, and overall architectural innovation solution for overcoming the hard challenges faced by cyber-resilient systems and structures. In the engineering sense, this is the first time to empower the quantifiable design and verifiable measurement of cyber-physical systems in an integrated manner.

Architectural Feature 1 Cyber-attack awareness amplification which is $1 + 1 > 2$. The DHR architecture breaks the traditional assumption that variety and redundancy only provide the capability of resource backup and substitute repair. Through the cross-verification of heterogeneous redundant configurations, it provides unprecedented super-awareness of “unknown unknown” cyber-attacks. At the same time, the strategic rulings in the DHR architecture also have the ability to sensitively perceive the performance indicators of mission/service functions. In this way, the DHR architecture seamlessly combines the traditional cyber resilience’s mission/service running status awareness with ESS’s cyber-attack awareness, providing a basic technical framework for supporting the thinking perspective of cyber resilience with ESS described in 5.1.1.

Architectural Feature 2 Cyber-attack surface reduction $1 + 1 < 1$. The DHR architecture breaks the “conventional consensus” that variety (diversity) and redundancy techniques will increase the surface of cyber-attacks. It greatly reduces the attack surface through its own structural encryption features. At the same time, the DHR architecture has its own adversarial properties such as obfuscation and

deception. The cyber resilience mechanism can naturally integrate non-adversarial cyber resilience adversity response and adversarial cyber resilience attack suppression technologies, and its GFS can be quantitatively designed and verifiably measured.

Architectural Feature 3 The cyber security and resilience of the system are greater than that of the sum of its parts. The DHR architecture is based on an open system interface and has a modular, loosely coupled, and efficient cohesive system structure. It supports COTS-level components and products of the global commercial supply chain and does not require that the components themselves have no vulnerabilities, backdoors, or common ESS problems such as the existence of viruses and Trojans. It enables the system to have greater cyber security and cyber resilience than the sum of its parts, so that the contradiction between advancement and trustworthiness, openness, and security can be unified unprecedentedly on the basis of DHR architecture.

The DHR architecture is inclusive of a variety of traditional cyber security and cyber resilience technologies. After a matrix clustering analysis, the current cyber resilience design principles and design technologies can be summarized into two major principles and technology clusters (Fig. 5.9): the zero-trust category of principles and technologies in the upper left corner in blue and the ESS category of design principles and technologies in the lower right corner in red. However, there is a lack of hierarchy and coordination among the various technologies in the current technology clusters. In fact, the number of technologies used is not necessarily related to the quality of cyber resilience. The DHR architecture responds well to the underlying questions of which technologies should be used as the backbone for constructing cyber resilience and how these cyber resilience technologies should be used.

The DHR architecture is like a “reinforced concrete” skeleton, whose inherent convergence characteristics can naturally accept a variety of traditional cyber security technologies in a “concrete filler” manner, capable of providing the system with exponential security gains and making the target object based on the DHR architecture a “building with a reinforced concrete texture”. This is a goal that is difficult to achieve by simply stacking or deploying layers of additional security technologies.

As shown in Fig. 5.10, the DHR architecture takes dynamics, variety (diversity), and redundancy as the cornerstone and absorbs diverse security technologies such as intrusion detection, threat perception, and access control. Through innovation at the system structure level, it plays the role of stringing pearls into a necklace and realizes cyber resilience in which “structure determines security.”

The DHR architecture realizes the natural integration of traditional cyber security and cyber resilience, avoiding the dilemma that over-emphasis on traditional cyber security may cause the system to have insufficient cyber resilience and over-emphasis on cyber resilience may make the security boundary full of vulnerabilities and lead to a larger attack surface. More importantly, the DHR architecture can completely change the current “two-skin” development pattern of cyber-physical systems and cyber security technologies. The use of “the concrete filler” will greatly

by one vote; if the system architecture is sound, a comprehensive evaluation of the system's cyber resilience and complexity should be carried out step by step.

Second, when comparing the cyber resilience assessment levels of different systems, a "key vote" winning system based on system architecture assessment should be established. The system architecture is used to focus on and solve the common problems of the system. When comparing the resilience levels of different systems, we must first look at the system architecture's ability to solve common problems and should put the ability and level of the system to solve individual problems in a secondary position. Based on the same level of handling common problems, the level of handling individual problems should be further compared. Common problems can also be divided into two categories: common problems in different fields and common problems in the same field. Obviously, dealing with common problems in different fields requires a better system architecture.

Third, a "critical vote" scoring system for white-box testing must be established when assessing the overall performance of a system's cyber resilience. Since cyber resilience has the characteristics of integrity and emergence, we should pay attention to the research on the integrity evaluation and analysis methods of the system throughout its life cycle and under different environments. On the premise of good overall performance, we can then evaluate the performance of each sub-function and the quality of each performance indicator. The overall evaluation of the system cannot be replaced by the evaluation of each component of the system (the performance of cyber resilience cannot be derived by simply adding up the scores of each component or comprehensively aggregating it). It is necessary to innovatively develop a variety of destructive testing, white-box testing, red-blue confrontation, and other analytical assessment methods to test the system's network resilience capability in practice.

Table 5.1 provides a reference table for qualitative assessment of the overall performance of a cyber resilience system architecture.

The analysis and assessment of a system's cyber resilience architecture should run through the entire life cycle of the system, especially focusing on the performance of the system as an actual product in the actual operating environment. The assessment of the cyber resilience of system products in the actual operating environment can be carried out based on the system's commitment to the use of resilience technologies and the comprehensiveness and effectiveness of the implementation. This assessment includes not only the technical mechanism but also how it is operated and used in practice. Table 5.2 gives a qualitative evaluation reference for the cyber resilience performance of system products.

The ESS white-box testing methodology provides a new system security and resilience testing model for evaluating cyber resilience and provides a verifiable metric for evaluating system security and resilience under human attack conditions. ESS functional white-box testing is essentially a sensitive functional section of the code running against the executor. By manually placing "hooks," the function codes of unknown and non-destructive vulnerabilities, backdoors, and virus Trojans of the tested object are implanted into the corresponding executors of the current running environment, so that testers can inject or modify the function codes needed for

Table 5.1 Reference table for qualitative assessment of the flexibility and adaptability of cyber resilience system architectures

Level	Decryption
Very high	The system architecture explicitly integrates a strategically selected set of components, technologies, and processes to enable cyber resilience techniques; it can detect early signs of unknown unknown attacks and implement effective blocking; it can evaluate the effectiveness of all cyber resilience techniques and processes and explicitly provide the flexibility to integrate other technologies or components as they become available. The availability of missions or services in this system architecture is verifiably measurable
High	The system architecture explicitly includes the components, technologies, and processes required to achieve resilience and provides some flexibility to improve integration capabilities as other technologies or components become available
Medium	The system architecture accommodates or includes the components, technologies, and processes required to achieve resilience and provides some flexibility to incorporate other technologies or components as they become available
Low	The system architecture does not exclude components, technologies, and processes that enable resilient technologies and provides limited flexibility to utilize other technologies or components as they become available
Very low	This system architecture excludes many of the components, technologies, and processes that can enable resilient technologies and provides very limited opportunities for future consideration of other technologies or available components

white-box test cases through the channels between the system and the outside. Then through the input channel of the system, the tester sends the activation sequence of vulnerability, backdoor, and virus Trojan test items that meet the channel compliance check and observes whether the output response expected by the test case design appears at the output end. It should be emphasized that all the implanted test function codes are “unknown” to the test object.

5.2 Incremental Capability for Achieving the Goals of ESS-Empowered Cyber Resilience

ESS-empowered cyber resilience can not only achieve the cyber resilience objectives proposed by NIST but also gain capacity increment in the dimensions of the four goals [1] of cyber resilience, as shown in Fig. 5.11. In terms of prevention, the DHR architecture of ESS can realize the perception and discovery of unknown threats (“see the unseen”). In terms of resistance, the DHR architecture of ESS can provide comprehensive defense against human attacks and random failures, especially against human attacks based on 0-day vulnerabilities and preset backdoors, and can discover and reject persistent trial-and-error attacks of APTs (“refusal trial-and-error”). In terms of recovery, the DHR architecture of ESS can utilize the component resources provided by the heterogeneous resource pool to quickly replace failed components and reconfigure the system and quickly restore service

Table 5.2 Cyber resilience performance evaluation table of system products in the actual operating environment

Level	Commitment to cyber resilience fulfillment	Comprehensiveness of cyber resilience	Effectiveness of cyber resilience
Very high	Policies, contracts, and agreements support cyber resilience techniques; cyber resilience techniques are well integrated into operations; sufficient resources are allocated to support the use of cyber resilience techniques (life cycle costs, training), and investment/architectural plans include anticipated future mechanisms/capabilities	All technologies or methods permitted based on political, operational, economic, and technical considerations are adopted	Effectiveness is verified through penetration testing, white-box testing, attack-defense drills, and indicator tracking
High	Policies, contracts, and agreements accommodate cyber resilience techniques; cyber resilience can be integrated into operations; resources are allocated to support the use of cyber resilience techniques (life cycle costs, training); investment/architectural development plans include technology	Most of the technologies or methods permitted based on political, operational, economic, and technical considerations are adopted	Effectiveness is verified through penetration testing and limited attack-defense drills
Medium	Policies, contracts, and agreements accommodate some cyber resilience techniques that can be integrated into operations; limited resources are allocated to use some cyber resilience techniques (life cycle costs, training)	Some of the technologies or methods permitted based on political, operational, economic, and technical considerations are adopted	Effectiveness is verified through testing
Low	Plans are in place to amend policies, contracts, and agreements to accommodate the use of some cyber resilience techniques	It is planned to consider some technologies or methods	Verified by testing
Very low	There are no applications of cyber resilience technologies promoted based on political, operational, economic, and technical considerations	The adoption of technologies or methods is accidental rather than planned	Effectiveness not evaluated

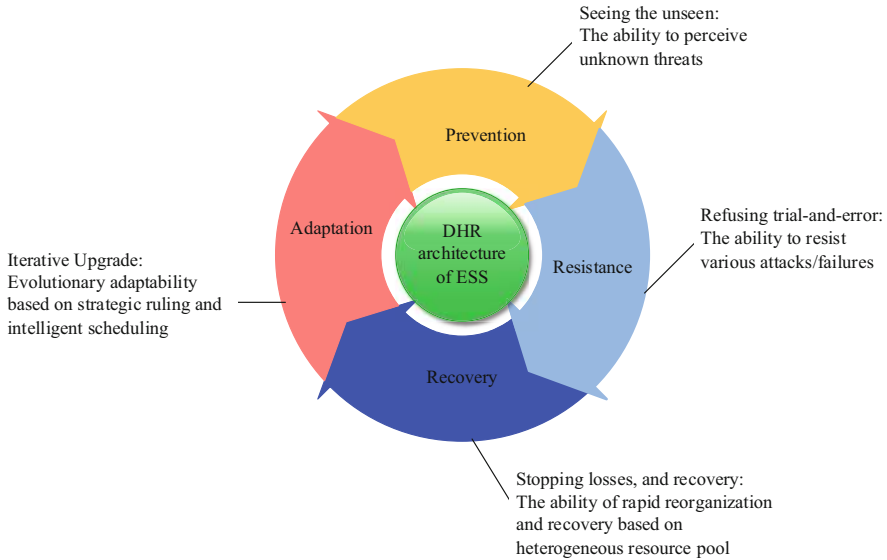


Fig. 5.11 ESS empowers the four goals of cyber resilience: prevention, resistance, recovery, and adaptation

capabilities (“stop loss and recover”). In terms of adaptation, the DHR architecture of ESS can, through intelligent strategic ruling and executor scheduling, achieve adaptive system evolution, automatically avoid frequent attacks, and achieve the adaptability similar to that of organisms to environmental changes (“iterative upgrade”). The following details the specific empowerment principles and effects of ESS mimic defense in various aspects of cyber resilience.

In order to refine goals into objectives, sub-objectives, and capabilities, NIST defined eight objectives of cyber resilience [1], all of which can be enabled through ESS to achieve the capabilities that the current network resilience method and framework cannot provide, as shown in Table 5.3. The sub-objectives and capabilities in Table 5.3 are unique to ESS-enabled cyber resilience and are crucial for achieving the critical “mission assurance” vision of cyber resilience.

In the temporal dimension, the first step is to prepare resources and to stop/avoid a potential attack through the use of defensive and deterrent means, in the hope of achieving the effect of “conquering the enemy without fighting.” After an attack occurs and before the threat is eliminated, three objectives are involved, namely, restraining, rebuilding, and sustaining. In the system dimension, firstly, the first is to restrain the attack and limit its impact, and the second is to rebuild the system so that it can be restored as soon as possible from the damaged state to the normal state. In the service dimension, the continuity of critical services must be ensured after the attack occurs and before it is eliminated. After the attack has been eliminated, the system needs to be reconfigured, the service process needs to be transformed, and the weak links need to be strengthened, so that the system is constantly adapted to

Table 5.3 Summary of ESS-enabled cyber resilience sub-objectives and capability increments

Objectives	Sub-objectives	Capabilities
Understanding	Perceiving cyber-attacks based on unknown vulnerabilities and backdoors	Discovering differential-mode attacks through strategic ruling
		Discovering common-mode attacks through executor random scheduling and strategic ruling
	Retrospectively analyze standardized logs and discover unknown threats	Outputting standardized logs
		Discovering unknown threats through collaborative analysis of standardized logs from multiple sources
Stop/avoidance	Blocking known and unknown cyber-attacks	Fixing known bugs in the executors
		Constructing heterogeneous executors by incorporating existing security protection techniques
		Running multiple heterogeneous executors in parallel
		Controlling output selection through strategic rulings
Stop/avoidance	Dynamically change the execution environment to avoid persistence attacks	Scheduling executors dynamically
		Restructuring executors dynamically
		Reconfiguring executors dynamically
Restraining	Identifying compromised resources	Identifying compromised resources through strategic ruling
		Identifying compromised resources through log analysis
	Limiting the impact of cyber-attacks	Limiting the attack surface to the executors
		Putting the abnormal executor scheduling offline
		Cleansing the attacked executors
Sustaining	Ensuring the correct functional operation in the event of an attack	Creating heterogeneous resource pools and component libraries
		Restructuring executors dynamically
		Reconfiguring executors dynamically
		Scheduling executors dynamically
		Enforcing strategic ruling
Rebuilding	Recovery from differential- or common-mode attacks	Perceiving differential-mode attacks via strategic ruling
		Discovering common-mode attacks through executor random scheduling and strategic ruling
		Cleansing the attacked executors
		Putting the abnormal executor scheduling offline
		Migrating services to standby executors quickly

(continued)

Table 5.3 (continued)

Objectives	Sub-objectives	Capabilities
Restructuring	Restructuring executors according to the threat environment	Adjusting heterogeneous resource pools and component libraries
		Adjusting the number and configuration of executors
Preparing	Providing resources and strategies to deal with unknown threats	Creating heterogeneous resource pools and component libraries
		Designing scheduling and strategic ruling policies for various abnormal situations
Transforming	Dynamically adjusting the parameters of scheduling and strategic ruling based on the threat environment	Adjusting scheduling policies and parameters
		Adjusting strategic ruling policies and parameters

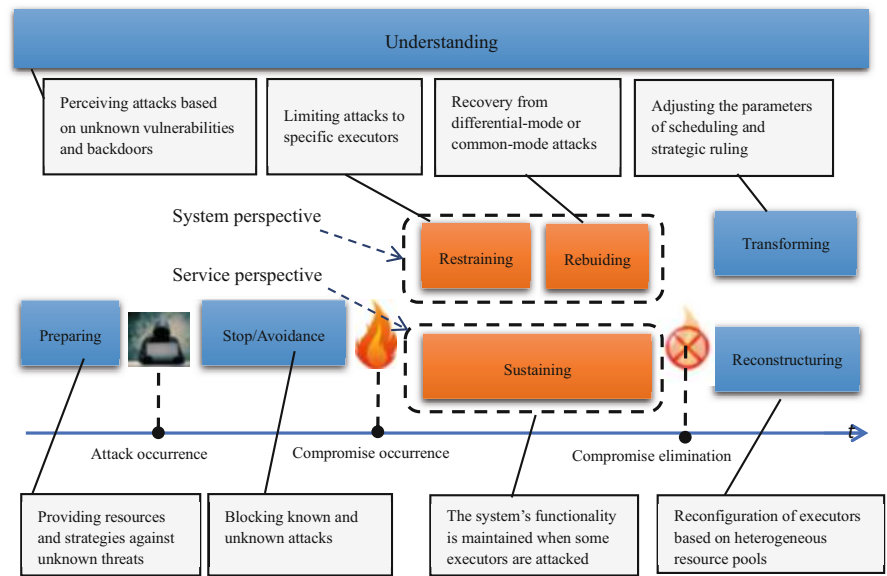


Fig. 5.12 Endogenous security empowers cyber resilience goals

the changing environment. Throughout the system’s life cycle, it is necessary to continuously understand the behavior of attackers and the state of defense, as an old saying goes: “know your enemy, know yourself, and you can fight a hundred battles with no danger of being defeated.” The relationship between the above objectives is shown in Fig. 5.12.

5.2.1 Seeing the Unseen

According to the DVR complete intersection principle mentioned in Chap. 2, as an implementation of DVR transformation, the DHR architecture of ESS has defense capabilities against differential-mode problems within the DVR domain that do not rely on any prior knowledge. DVR transformation hides the diversity and complexity of events in the domain of functional safety and cyber security problems. Whether it is a cyber-attack based on known vulnerabilities or a cyber-attack based on unknown vulnerabilities or backdoors, the event mapped to the DVR space has only two dynamic forms of expression: either the differential mode or the common mode, and this dimensionality reduction processing method makes it possible to discover cyber-attacks based on unknown vulnerabilities and backdoors. In the DHR architecture, the strategic ruling module receives output messages, system status, standardized logs, and other information from each reconfigurable operating scenario and achieves intelligent threat perception and output control through comprehensive online and offline analysis and processing, as shown in Fig. 5.13.

The DHR architecture of ESS provides group situational awareness capabilities based on multiple heterogeneous executors, with the ability to reduce the dimensionality of traditional individual executor perceptions. The DHR architecture’s threat perception usually requires three prerequisites [2]: (1) the inputs of the

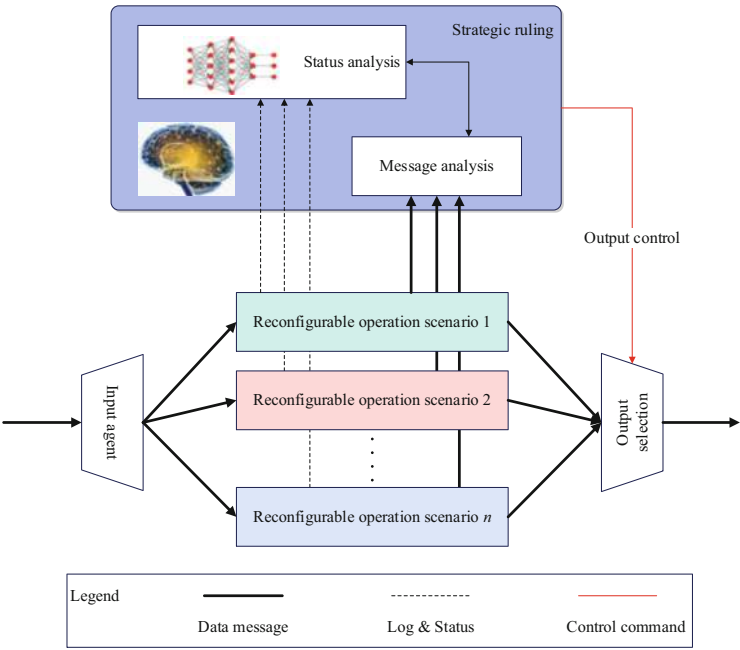


Fig. 5.13 Comprehensive processing of various types of data by the strategic ruling module

executors are standardized or normalizable and can be used to motivate multiple executors in parallel; (2) the output vectors of the executors are standardized or can be standardized after normalization and can support multi-mode ruling; (3) uncertain attacks on the executors are reflected in the output vector or system state. In engineering practice, targeted mimic structural design for different devices and system models can enable the system to satisfy the above three prerequisites in most cases. Adjusting the redundancy, the degree of heterogeneity, the comparison length, and the closed-loop response time, changing the strategic ruling policies, and changing the feedback functions can quantitatively control the common-mode escape probability and duration. Therefore, under normal conditions, the DHR architecture of ESS can transform “unknown unknown security threats” at the level of individual executors into “known unknown security problems” at the level of a group of heterogeneous executors [3], i.e., into differential-mode problems in the DVR space [4]. It can be seen that the DHR architecture of ESS has a strong ability to perceive unknown threats, and this ability is not available in traditional intrusion detection systems based on prior knowledge.

The ESS equipment system can further enhance security situation awareness by outputting standardized logs and combining it with the back-end log analysis system based on big data technology. Based on a series of standards for comprehensive ESS mimic defense devices and systems, it is necessary to propose a unified and structured expression method for ESS events in cyberspace to ensure consistent expression of events in the process of production, sharing, and consumption, thereby improving the efficiency of event information sharing, interoperability, and the overall cyber security situation awareness. Currently, a research team of Purple Mountain Laboratories is developing technical specifications for the universal log expression of ESS devices and systems. A cyberspace ESS event expression model is established to describe cyber security events in six dimensions, the event domain, event action, event object, service type, event result, and event subject, and it provides a unified vocabulary and syntax that can be used to record, share, and interpret log data, as shown in Fig. 5.14. This technical specification also provides unified engineering constraints on which events are worthy of triggering logging, how logging is serialized and de-serialized, and how the transmission protocols and storage specifications are used [4].

The back-end log analysis system is based on standardized log information to complete the paradigmatic analysis and data correlation analysis of ESS device system logs. Figure 5.15 shows the architecture of intelligent cloud management system of ESS device logs. The back-end log analysis system serves as an auxiliary system to deeply mine the log data of mimic devices and systems and carry out application innovations in such aspects as threat tracking and tracing, common-mode escape warning, dynamic security modeling of the executors, and secondary auxiliary ruling. At the same time, an engineering implementation method is adopted that combines the general specifications for log expression with the best logging practices for each specific service form in the CMD field, so that each specific mimic service can expand the standardized specification of logs with the characteristics of

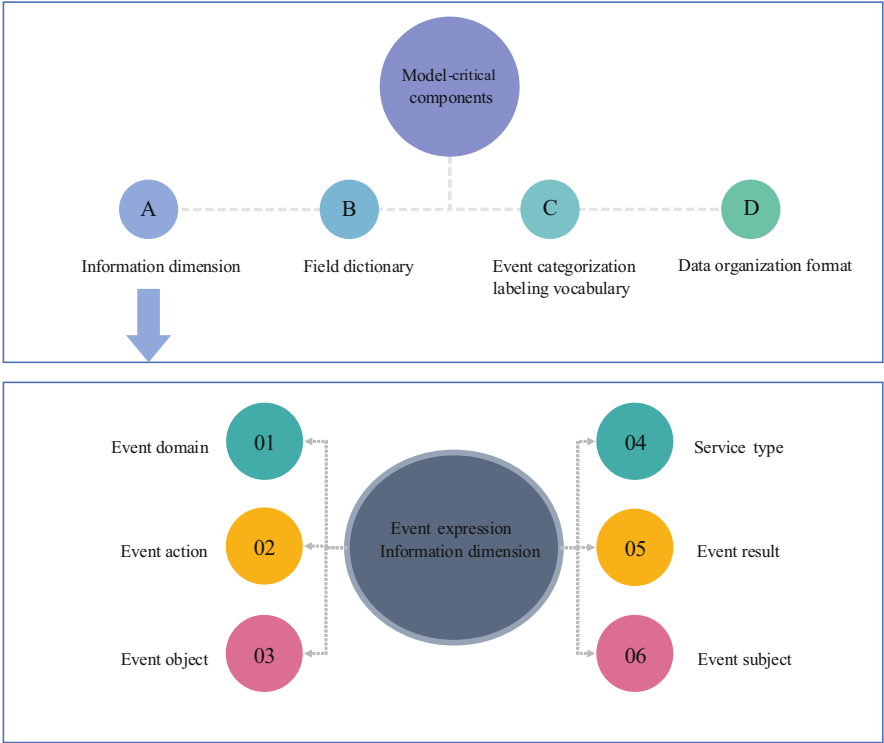


Fig. 5.14 Standardized log component model of ESS equipment and systems

the service according to the general specification and form a cluster of standard specifications of logs of ESS devices and systems.

The output messages and standardized logs of each reconfigurable operating scenario constitute the “ruling big data” required for strategic ruling. Based on “ruling big data” and various data analysis methods, multi-dimensional and intelligent strategic ruling can be realized. For example, traditional machine learning methods include using algorithms such as support vector machines, logistic regression, and weak classifier ensembles to analyze the synchronized robustness interference data logs generated by the controller after responding to threats and assess the hazard [5, 6]. The lightweight computing model of traditional machine learning methods can save computing time and cost and realize real-time analysis of logs. The deep learning function based on big data has various application modules such as supervision, management, vulnerability sensing, threat warning, etc., including trustworthiness judgment of multi-mode ruling on the standardized output vector of the executors, perception of new vulnerabilities in the reconfigured executors, and assessment and early warning of external threat levels based on prior attack data. Among them, the operation scenarios, system status, and log output of the IDS system within the executors based on the reconfigurable executors can be used for

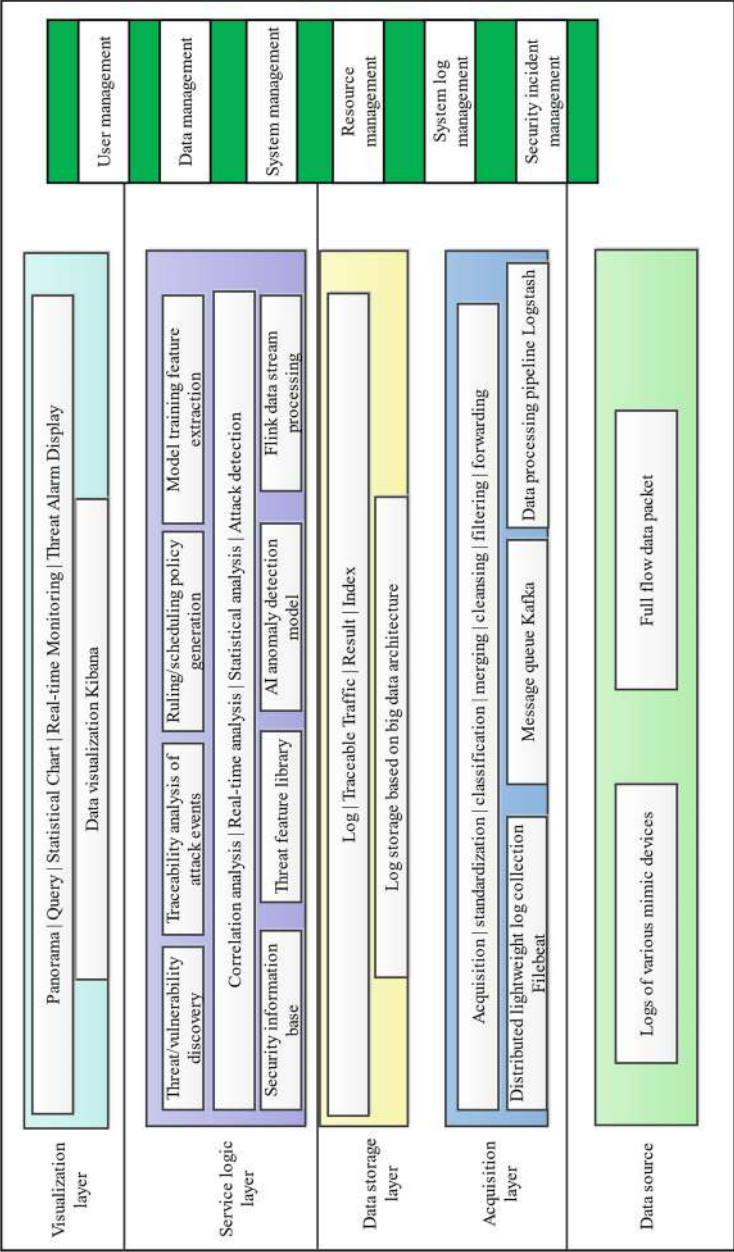


Fig. 5.15 The architecture of intelligent cloud management system of ESS device logs

the sample sets of big data training. The raw data undergoes unified regularization and normalization operations on the built-in data platform based on the intelligent cloud management system of mimic device log. For modular matrix log information, the convolutional neural network (CNN) will be trained through deep learning to predict and identify complex attacks [7–9]. For temporal attack sequences and device attack sequences, the recurrent neural network (RNN) is trained to predict the next attack time, attack location, and attack method of unknown threats, so as to reasonably configure the strategic ruling mechanism [10]. Strategic ruling utilizes a combination of traditional machine learning methods and deep learning methods to efficiently and accurately perceive unknown threats, with the support of the executor's heterogeneity information [11] and ruling big data.

A typical application example is file transfer. File transfer is the most basic data service in the Internet era, and data file encapsulation formats such as Word, PPT, and Excel introduced by Microsoft have been supported as a de facto standard by many office software developers, such as Kingsoft's WPS Office, Alibaba Baba's DingTalk, Tencent's Enterprise WeChat/Tencent Docs, Beijing Feishu, etc. However, hackers often insert virus Trojans into these files for the purpose of network attacks. Existing knowledge base-based virus Trojan detection tools cannot effectively respond to these network attacks because they cannot identify unknown virus Trojans. In addition, these Trojan viruses generally require the support of vulnerabilities and backdoors in specific scenarios to achieve the goal of attacks. In other words, different application scenarios and office software, as well as the underlying software and hardware execution environments, make it difficult for attack codes that rely on backdoors and vulnerabilities in a specific application scenario and software and hardware execution environment to work under cross-scenario and cross-platform conditions. Therefore, based on the DHR architecture, several kinds of mainstream office software and operating environments (frequently used by the target objects) can be appropriately selected as reconfigurable redundant executors. For example, when receiving a Word file packet, the DHR system will simultaneously use multiple kinds of office software to open it in different execution environments. If the Word body data packet output by each executor meets the requirements of the DHR architecture's strategic ruling, the body data packet output by the ruling will be re-encapsulated in a standard Word document format for safe use in subsequent processing stage. Obviously, after such "disinfection and inactivation" treatment, the Word file will be able to ensure the security of its data and content when it is used in the subsequent scenarios. This method has universal significance for application scenarios in which data files are received from many directions and security cannot be guaranteed. In particular, if the ruling result contains anomalous outputs, it is possible to discover the attack resources and mechanisms such as viruses and Trojans in the files as well as the vulnerabilities and backdoors in the related software/hardware environments through background processing. It should be noted that although the "file cleansing and disinfection" mechanism based on DHR does not rely on traditional cyber security methods such as knowledge bases of vulnerabilities, backdoors, and virus Trojans, it is not difficult to obtain an exponential security gain according to the intrinsic properties of the

DHR architecture if the executor environment within the DHR architecture is appropriately configured with diverse “antivirus and sterilization” software or even intrusion detection means such as honeypots and sandboxes.

5.2.2 *Refusal Trial-and-Error*

Unknown vulnerabilities and backdoors have long been the biggest threats to cyberspace security and safety. The DVR complete intersection principle (CIP) put forward in Chap. 2 states that all unknown security threats in a cyber-physical system, regardless of whether they originate from random or uncertain perturbations or from man-made or non-man-made disturbances, can be expressed as differential-mode or common-mode problems in the DVR domain. As a DVR transformation structure, the DHR architecture can transform cyber-attacks based on unknown vulnerabilities and backdoors into differential-mode or common-mode disturbances in the DVR space. Theoretically, all differential-mode disturbances can be dynamically shielded (or corrected). This makes it impossible for attackers to identify the target, difficult to evaluate the attack effect, impossible to inherit the attack experience, and difficult to reproduce the attack scenario. Literature [3] analyzed the attack resistance of the DHR architecture and revealed the contribution of heterogeneous and dynamic characteristics to reducing the probability of successful cyber-attacks. The multi-dimensional dynamic reconfiguration mechanism inherent in the DHR architecture makes the apparent environment of the target object highly uncertain. Theoretically, it is difficult to get the same output response for the same input stimulus except for a given function. For example, the number, types, characteristics, exploitability, and frequency of occurrence of vulnerabilities seen by the attacker are certain in a static environment but uncertain in a dynamic environment. This will seriously interfere with the authenticity of the information obtained by the attacker in the vulnerability detection and reconnection stages and will also cause the attack implantation or uploading process to lose reliability. It will also undermine the reproducibility of successful attacks in the spatial-temporal dimension [3].

Literature [2] makes an analogy between the effect of the DHR architecture and the theory of channel coding and quantitatively analyzes the gains brought by dynamics, heterogeneity, and redundancy to channel coding. Whether from the perspective of qualitative and quantitative analysis or actual evaluation, it is not difficult to draw the conclusion that the DHR architecture can obtain nonlinear defense gains from the intrinsic mechanisms, and the failure rate of the DHR architecture also decays exponentially. Judging from the typical engineering structure of DHR, the upper limit of its defense cost is proportional to the number of heterogeneous executors in the mimicry boundary, but its defense effect increases non-linearly. More importantly, from the perspective of the entire life cycle of the application system, the DHR architecture significantly reduces real-time protection requirements (such as defense against 0-day attacks, etc.), the frequency of version upgrade (patching), and the maintenance costs of additional specialized security

devices. From an implementation perspective, DHR allows the use of globally marketed COTS-grade hardware and software components to compose heterogeneous executors, even direct use of open or open-source products. Compared with specially designed and manufactured security components or systems, the development and after-sales service costs of DHR components can be absorbed by the large-scale market [2].

5.2.3 *Stop Loss and Recover*

In the face of failures caused by random errors or human attacks, devices and systems based on the DHR architecture can quickly build new executors from heterogeneous resource pools and schedule them online to quickly restore system service capabilities. Through functional decoupling and modularization design, the functional components required by each executor are presented in the form of heterogeneous resource pools. Theoretically, an exponential number of heterogeneous executors can be generated by combining heterogeneous components with each other. Each executor can be hosted by either a physical computing environment or a virtual computing environment. According to the result of the strategic ruling, new executors can be quickly generated and scheduled to work online, thus enabling the system to have rapid recovery capability. In contrast, traditional systems or system components based on redundancy or the DHR architecture do not have the same resilience after an attack due to the static nature of the system.

In order to make cloud computing services recoverable when a system failure occurs, Microsoft has proposed three design principles to ensure the reliability of cloud computing systems [12]:

1. Design for resilience. Cloud computing services must be able to tolerate the failure of system components without human intervention and should be able to detect system failures and automatically take corrective measures when they occur so that the user does not notice an interruption in service. When the service fails, the system should be able to provide partial functionality rather than completely collapse.
2. Design for data integrity. In the event of a system failure, services must be able to manipulate, store, and discard data in a manner consistent with normal operation and maintain the integrity of user-hosted data.
3. Design for recoverability. The system should be able to ensure that service is automatically restored as quickly as possible in the event of an anomaly, and when a service disruption occurs, the system maintainer should be able to restore service as quickly and completely as possible.

In order to achieve rapid recovery from failures, existing cloud computing systems usually use the hot backup technology to run multiple service instances at the same time and use specific consensus protocols to maintain consistent status among multiple service instances. When the primary instance fails and cannot

provide services, it immediately switches to backup instances. This technology is effective for failures caused by random errors in the system. However, for failures caused by cyber-attacks, if multiple isomorphic service instances are used, the same attack risk will always be faced, causing the system to fall into a dead cycle of “failure-switching-failure,” unable to provide services stably.

The DHR architecture of ESS solves the above problems by adopting system decoupling and component heterogeneity. Firstly, functional decoupling is carried out for each executor, and the executor is decomposed into several basic functional components; secondly, for each functional component, differentiated compilation, independent libraries, and multiple versions are used to construct a heterogeneous component library. Through different combinations of heterogeneous components, diverse functionally equivalent heterogeneous executors can be constructed. Executors can be constructed in two ways: pre-construction and real-time construction; multiple executors are stored in the executor pool and can be called at any time to provide support for the system’s rapid recovery from failures. By adopting functionally equivalent heterogeneous executors, the DHR architecture can simultaneously cope with failures caused by random errors and human attacks and has a good ability to stop loss and recover.

5.2.4 Iterative Upgrade

The adjustment and adaptation ability of information systems is of great significance to achieving system resilience. A system with good adaptability can make adaptive adjustments when faced with network attacks or failures to prevent the same attacks or failures from taking effect repeatedly. Gartner drew on the idea of “survival of the fittest” in biological evolution and proposed the Adaptive Security Architecture (ASA) [13]. He pointed out that successful IT infrastructure has the ability to adjust and adapt similar to the human body’s immune system or natural ecosystems and be able to continuously adjust and evolve in the face of cyber-attacks.

Although Gartner proposed the concept of the ASA architecture, he did not give a way for information systems to gain adaptive adjustment capability. The DHR architecture of ESS provides an effective technical approach to this. Through strategic ruling and intelligent scheduling, mimicry devices and systems will be able to better cope with the current threat environment and schedule executors online, thereby achieving good environmental adaptability. The scheduling strategy of the DHR architecture takes into account the trustworthiness, heterogeneity, and relevance of executors and automatically avoids scheduling executors with frequent ruling anomalies online, so that the DHR architecture system can demonstrate the ability to adapt to external threats. This ability is not available in traditional network equipment systems. For example, the literature [11] proposed an intelligent scheduling method for mimic-router executors. The method takes into account the trustworthiness and relevance of executors and assigns higher scheduling weights to executors that have been attacked less frequently and are highly heterogeneous. In

this way, the trustworthiness of an executor that is frequently attacked within a period of time will drop rapidly, which will lead to a reduction in its scheduling weight and a corresponding reduction in the chance of being scheduled online. The above intelligent scheduling mechanism can effectively prevent vulnerable executors from being repeatedly exploited by attackers, enabling the system to realize adaptive evolution and adjustment in the face of attacks, and obtain an evolutionary capability similar to that of living organisms.

5.3 Technologies of Cyber Resilience Engineering Empowered by ESS

As a necessary design and operational attribute of next-generation heterogeneous networks, cyber resilience is defined as the ability to provide and maintain the smooth completion of normal missions in the face of different failures or attacks. Therefore, how to improve cyber resilience has become the focus of current research. Although there are many existing cyber resilience techniques or methods, including but not limited to adaptive response techniques, analysis and detection techniques, reorganization techniques, redundancy techniques, etc., the correlation between cyber resilience techniques or methods is not strong and lacks systematization, unable to provide a complete and operable technical system to guide the implementation of cyber resilience engineering. The ESS development paradigm based on dynamic redundancy structure can provide a set of proven systems, rules, and methods for resilience engineering. The following is a technology tree that expresses resilience engineering from the perspective of ESS-empowered cyber resilience engineering.

5.3.1 Technology Tree

The technology tree provides a technical overview of ESS-empowered cyber resilience engineering, extends the ESS technology to the cyber resilience engineering technology, and explores the connection between the two. The technology tree is generally divided into three parts: root, branch, and leaf. Root technologies are all ESS technologies, including the mimic structure modeling language technology, mimic structure programming theory, mimic critical position identification technology, mimic structure compilation technology, mimic structure debugging technology, input assignment and adaptation technology, etc. Branch technologies are cyber resilience engineering technologies powered by all ESS technologies and based on ESS testing standards, technical specifications, mimic structure operating environment, etc., which correspondingly empower 12 cyber resilience engineering technologies such as situational awareness, unpredictability, analysis, diversity, and

Table 5.4 Composition of root technologies

ESS technologies	Basic technologies
Mimic structure design technologies	Mimic structure modeling language technology Mimic structure programming theory Mimic structure circuit design theory Mimic structure algorithm design theory Mimic structure software and hardware co-design theory
Mimic structure implementation technologies	Mimic critical position identification technology Mimic structure compilation technology Mimic structure platform-independent design technology Mimic structure software-defined technology Mimic structure middleware technology
Mimic structure evaluation technologies	Mimic structure debugging technology Mimic structure complexity analysis theory Mimic structure vulnerability analysis technology Mimic structure simulation technology Mimic structure testing technology Mimicry degree assessment theory and method Mimic structure power consumption control technology
Mimic structure technology mechanisms	Mimic structure debugging technology Mimic structure complexity analysis theory Mimic structure vulnerability analysis technology Mimic structure simulation technology Mimic structure testing technology Mimicry degree assessment theory and method Mimic structure power consumption control technology

technology, mimic structure evaluation technology, and mimic structure technology mechanism. The four categories comprise 24 basic technologies which are listed in Table 5.4.

ESS technologies, as the technical basis for empowering cyber resilience engineering, are mainly guided by the DVR complete intersection principle and constructed with the DHR architecture as the core to build an ESS base, thereby achieving the purpose of preventing unknown security threats. Some of in root technologies provide multi-faceted and multi-angle empowerment for diversity, redundancy, coordinated protection, reorganization, adaptive response, and other technologies in cyber resilience engineering, thereby enhancing the cyber resilience of the system.

5.3.3 *Branch Technologies*

Branch technologies as the most direct and relevant technologies for ESS-enabled cyber resilience engineering mainly include technologies of adaptive response, coordinated protection, variety, reorganization, and redundancy. These technologies contain the core elements in the field of cyber defense, with ESS-enabled security gains, and significantly embody the ability of the ESS theory in prediction, resistance, recovery, and adaptation to realize the objectives of cyber resilience engineering as well as to optimize the design strategy of cyber resilience, so that the network maintains a systematic structure and good resilience and can easily respond to subsequent attacks or damage. The branch technologies that improve cyber resilience will be described in detail below.

5.3.3.1 Adaptive Response Technology

The adaptive response technology aims to implement agile actions to manage risks, with the objective of optimizing the ability to cope with adverse conditions, stress, and attacks in a timely and appropriate manner, so as to maintain the normal operation of missions or services to the greatest possible extent. Therefore, the adaptive response technology has intelligent and resilient security protection capabilities in the face of attacks and relies on continuous monitoring and retrospective analysis to build an ESS-empowered threat warning system, which forms a closed-loop security protection and control process integrating defense, monitoring, response, and prediction and is conducive to the construction of a secure and resilient system that reduces the adverse impact on the ability to perform mission-critical functions.

5.3.3.2 Coordinated Protection Technology

The coordinated protection technology is used to ensure that the system's protection mechanisms operate in a coordinated and effective manner. When a threat event occurs, it makes it more difficult to successfully attack critical resources by increasing the cost of the attack and increasing the likelihood that the attacker is detected. Therefore, regardless of the type of threat event, this technology must ensure that the protection mechanism used will not be interfered with by other protection mechanisms, thereby disrupting or interfering with the attempts to achieve a coordinated attack through trial-and-error means and significantly reducing the stability and reliability of the attack chain.

5.3.3.3 Diverse Technologies

The diverse technologies are designed to leverage heterogeneity to minimize common-mode failures, especially threat events that exploit common vulnerabilities. Its essence is to prevent the possibility of losing critical functionality by replicating the failures of common critical components. Under certain heterogeneity or security performance requirements, diverse technologies explore how to automatically construct executors with high heterogeneity requirements by assigning multiple expressions of the same entity with the same functionality and perform mimic assemblies to build secure architectures that are resilient to both intentional and unintentional cyber-attacks.

5.3.3.4 Reorganization Technology

The reorganization technology addresses mission/service functional needs by structuring the use of systems and resources, reducing current and anticipated risks, and adapting to the evolution of the technological, operational, and threat environments. This technology is presented in the mimic system by selecting different sets of elements to construct a variety of options for heterogeneous executors or defense scenarios, so that limited resources can provide richer structural representations of the target object. The process of reorganization is equivalent to a complex orchestrated combination of different options, which can not only expand the dissimilarity of the pool of mimic resources but also provide a flexible means of implementation for the problem avoidance mechanism. Therefore, the empowerment of the reorganization technology has important engineering significance for reducing the probability of attack escape.

5.3.3.5 Redundancy Technology

The redundancy technology reduces the consequences of loss of information or services by adding multiple protected instances of critical resources, thereby speeding recovery from the impact of adverse cyber events. Therefore, a network system based on the redundancy technology can intuitively reflect the resilience of a resilient network. Compared to single-target attacks, this technology, without the need for any prior knowledge of the attack or precise identification of characteristic information, can enhance the network system's ability to "tolerate viruses," and even if there are security risks or untrustworthy elements in some of the functions of the network system, the entire system will not be affected in providing normal services. For this reason, when assessing the security risks of network systems, the design of redundancy must be considered in order to ensure the normal function of the system through alternative conversion when the target is attacked.

5.3.4 *Leaf Technologies*

The description of branch technologies in Sect. 5.3.3 reveals strong correlation between ESS and resilience engineering. This strong correlation extends the capability of branch technologies to form leaf technologies, enabling branch technologies to promote the rapid development of leaf technologies in terms of technical structure, operation mechanisms, and institutional security.

ESS's empowerment of leaf technologies is more diversified and specific, including a wide variety of implementation approaches and evolving technological means. Approaches of implementing leaf technologies include, but are not limited to, representative approaches to adaptive response technologies, representative approaches to coordinated protection technologies, representative approaches to diverse technologies, representative approaches to reorganization technologies, and representative approaches to redundancy technologies. The technical means for the evolutionary development of leaf technologies are evolving technologies built on top of branch technologies, consisting of analytical monitoring, situational awareness, non-persistence, deception, dynamic positioning, integrity verification, and unpredictability. A detailed description of leaf technologies is given below.

5.3.4.1 Representative Technologies of Adaptive Response

Representative approaches to adaptive response technologies include methods such as dynamic reconfiguration, dynamic resource allocation, and adaptive management. Dynamic reconfiguration is the process of making changes to an individual system, system element, component, or set of resources in order to change functionality or features without disrupting service. Dynamic resource allocation is the change of resource allocations to missions or functions without terminating critical functions or processes. Adaptive management means to change the use of mechanisms in response to changes in the operating environment as well as changes in the threat environment. These approaches enhance the ability of network systems to detect signs of anomalies, failures, or attacks, providing an adaptive model for building new resilient networks.

5.3.4.2 Representative Technologies of Coordinated Protection

Representative approaches to coordinated protection technology include calibrated defense-in-depth, consistency analysis, orchestration, and self-challenge approaches. Calibrated defense-in-depth is the provision of complementary protection mechanisms at different architectural layers or at different locations. Consistency analysis is a coordinated approach to application protection to minimize interference, potential cascading failures, or coverage gaps.

Orchestration is the coordination of modifications to implementation mechanisms and processes at different layers, in different locations, or for different levels of confidence to avoid cascading failures, disruptions, or coverage gaps. Self-challenge is the validation of the effectiveness of mission/service processes or system elements in a controlled manner. These approaches are empowered by ESS to optimize information about network topology, network services, and network state and enhance the maintainability and recoverability of resilient networks.

5.3.4.3 Representative Technologies of Diversity

Representative approaches to diverse technologies include approaches to architectural diversity, design diversity, synthesis diversity, information diversity, pathway diversity, and supply chain diversification. Architectural diversity is the use of multiple sets of technical standards, different technologies, and different architectural models. Design diversity is the use of different designs to meet the same requirements or provide equivalent functionality within a given architecture. Synthesis diversity is the transformation of software implementations to produce a variety of instances. Information diversity refers to providing information from different sources or transforming information in different ways. Pathway diversity is the provision of multiple independent paths for command, control, and communication. Supply chain diversification refers to the use of multiple independent supply chains for critical components. The core of the variety approach is to increase network resilience by expressing the target entity in multiple and diversified variants so that network attacks face greater complexity and difficulty.

5.3.4.4 Representative Technologies of Reorganization

Representative approaches to reorganization technologies include purpose, offloading, restriction, substitution, specialization, and evolution approaches. The purpose approach refers to ensuring that the use of network resources is consistent with the purpose of the mission or service function, thereby avoiding unnecessary sharing and complexity. The offloading approach refers to the offloading of supported but non-essential functionality to other systems or to external vendors that are able to perform the functionality more securely. The restriction approach means removing or disabling unneeded functionality or links or adding mechanisms to reduce vulnerability or failure. The substitution approach means replacing low assurance or poorly understood implementations with trusted implementations. The specialization approach means uniquely adding, configuring, or modifying the design of critical network resources used for mission or service functions to improve trustworthiness. The evolution approach refers to the provision of mechanisms and architectural resources that enable a system to be maintained, modified, extended, or used in new ways without increasing security or mission risks. The essence of the reorganization method is to flexibly change the structural representation of the

system so that limited resources can adapt to more application requirements, thereby greatly weakening the attacker's attack advantage and reducing the impact and consequences of the attack.

5.3.4.5 Representative Technologies of Redundancy

Representative approaches to redundancy technologies include methods such as protected backup and recovery, spare capacity, and replication. Protected backup and recovery means backing up information and software (including configuration data and virtualized resources) in a manner that protects their confidentiality, integrity, and authenticity. Safe and secure recovery is achieved in the event of disruption or corruption. Spare capacity is the extra capacity reserved for information storage, processing, or communication. Replication refers to duplicating hardware, information, backup, or functionality in multiple locations and keeping them synchronized. Thus, these methods improve the system's protection level through multiple alternative operations to defend against or prevent attacks.

5.3.4.6 Representative Technologies of Evolutionary Development

1. Analytical Monitoring Technology

The analytical monitoring technology refers to the continuous monitoring and analysis of a wide range of attributes and characteristics in a coordinated manner. Since it is impossible to accurately predict the attack techniques used by attackers, this technology facilitates the creation of a heterogeneous resilience model based on situational awareness so that the model can detect the occurrence of events, analyze the current state, and predict the next state, thereby enhancing the system's prevention and monitoring capabilities.

2. Situational Awareness Technology

The situational awareness technology refers to building and maintaining a current representation of the status of a mission or service function while considering threat events and courses of action. Guided by ESS technologies, these methods enable network status management and improve situational awareness of cyberspace.

5.3.4.7 Non-persistence Technologies

Non-persistence technologies refer to generating and retaining resources on demand or for a limited time. The technologies are a presentation of the uncertainty, dynamics, and adaptability of the ESS-empowered system, thus increasing the difficulty for attackers to analyze network vulnerabilities while increasing the system's ability to predict the best time of attacks.

5.3.4.8 Deception Technologies

Deception technologies are designed to mislead, confuse, hide critical assets from, or expose secretly contaminated assets to the attacker. With mimic disguise as the core idea and by introducing deceptive elements, the technologies are used to confuse the attacker, prevent vulnerabilities from being exploited, and increase the uncertainty of results.

5.3.4.9 Dynamic Positioning Technologies

Dynamic positioning technologies refer to the allocation and dynamic repositioning of functional or system resources. The technologies enable dynamic positioning of cyber resilience situational elements to respond to current cyber-attacks or predict future cyber-attacks and improve threat analysis and reconnaissance capabilities.

5.3.4.10 Integrity Verification Technology

The integrity verification technology is a technology that determines whether critical system elements have been compromised. When it is confirmed that system integrity has been compromised, the ESS-empowered cleansing and recovery mechanism will be quickly activated to remove the compromised system elements and enhance the system's recoverability.

5.3.4.11 Unpredictability Technology

The unpredictability technology is a technology where changes are made randomly or unpredictably. The unpredictability of the system can significantly improve its security and reduce deliberate harm based on unknown vulnerabilities, backdoors, etc.

5.4 Design Principles of ESS-Empowered Cyber Resilience

5.4.1 Principles for Setting Up Critical Defensive Positions

Critical position defense refers to the practice of investing better protection design and more protection cost for key parts of the information system, such as codes, modules, and components, which perform important functions or are vulnerable to low-cost attacks, while investing relatively less for other parts of the information system, so as to achieve a balance between overall security and protection cost.

Critical position defense can reduce the additional cost of cyber resilience engineering and maximize the gain in cyber resilience at the minimum cost.

In order to reduce the difficulty and cost of engineering development, the ESS mechanism needs to follow the principle of “fortifying and defending the most important positions” when it is applied in specific systems. In other words, during system design, it should be clarified which resources or functions need to be protected first and foremost as critical positions for defense. The resources and functional modules in the critical defensive positions should be incorporated into the scope of CMD by designing and developing components such as input agents, output selection, strategic ruling, and feedback control (also known as “mimetic bracket” components) to apply the DHR architecture for defense. CMD for complex systems may require multiple CMD boundaries for different security standards in order to meet actual needs. Heterogeneous components (which may also be “black-box” components) that are not trustworthy or whose supply chain security cannot be ensured and which are functionally equivalent can be assembled in accordance with the DHR architecture to achieve the purpose of being autonomous, controllable, safe, and trustworthy.

The principles for setting up critical defensive positions include the following:

1. Principle of functional normalization. Critical defensive positions are generally set up on standardized or normalizable operation interfaces that can be identified by functional compliance and consistence testing and in situations with a high degree of centralized control, such as control command distribution channels for some network devices.
2. Attack reachability principle. Critical defensive positions should be set up on the attacker’s attack path, especially the attacker’s usual attack entrances and the target components involved in the key nodes of the attack chain. All those should be included in the protection scope as critical defensive positions, such as management configuration channels and key service function processing components of some services or devices, user input channels of web applications, etc.
3. Engineering implementation complexity principle. Since mimic bracket components need to be deployed in critical defensive positions, the deployment location of the mimic bracket should be reasonably considered when selecting critical defensive positions, so as to make the processing flow and engineering implementation of the mimic bracket (especially the input agent) as simple as possible and to avoid the introduction of new vulnerabilities due to excessive complexity of the program.

5.4.2 The Principle for Unknown Threat Analysis

Many systems in cyberspace are overly complex, with huge amounts of software/hardware codes and consequently an increasing number of vulnerabilities in the system. Although verification testing of system software/hardware has received widespread attention, the current verification capabilities are not enough to keep

up with the increase in demand caused by code complexity, and it is difficult to model and test all possible paths and states. Code verification is also incapable of verifying unknown and unpredictable security risks. The more complex the codes, the greater the likelihood of unpredictable vulnerabilities will be. Therefore, identifying and eliminating all vulnerabilities in a system is an unattainable goal. In addition, due to the development of the globalized economy and the specialization and refinement of industrial division of labor, integrated innovation or manufacturing has become a common mode of production organization; the design chain, tool chain, production chain, supporting chain, and service chain and of various products are getting longer and longer; and the scope and links involved are getting wider and more numerous. Supply chains that are untrustworthy or whose untrustworthiness is difficult to accurately control have brought great challenges to security management and provided numerous opportunities for vulnerabilities and backdoors to be pre-embedded. In short, in these complex systems, unknown and unpredictable vulnerabilities and human-placed backdoors are bound to exist and cannot be avoided or eliminated. These unknown vulnerabilities and backdoors can be exploited by APT attackers to launch intelligent, long-term, and persistent attacks. Moreover, it is difficult to detect and defend against these attacks through traditional protection systems based on prior knowledge and attack characteristics. This requires that the system to remain resilient in the face of these known unknown risks and unknown unknown threats.

This principle means that the system should have the ability to detect unknown threats and attacks. The ESS mechanism based on the DHR architecture can transform any unknown attacks against the executors into attacks that can be perceived at the structural level through the relative judgment of the output vector. Based on such information as target system's operation logs and the snapshots of abnormal scenes, and using big data analysis and AI technology, it is possible to discover unknown vulnerabilities and backdoors and analyze related attack methods. In addition, this principle requires the system to be able to prevent the execution of unknown attacks as much as possible. The endogenous mechanism of DHR can continuously change the host execution environment by dynamically and randomly performing operations such as reconfiguration, reorganization, virtualization, cleansing, and repair of heterogeneous executors, thus externally displaying a state of environmental uncertainty. This makes it difficult for attackers to achieve the planned attack effect based on intelligence information obtained in advance or on-site, formulated attack strategies, and selected technical means. Secondly, this principle requires the system to maintain sufficient performance and service quality in the face of unknown attacks from unknown threats. Based on multi-mode ruling and heterogeneous redundant executors, the DHR endogenous mechanism is able to thwart any differential-mode cyber-attacks based on unknown vulnerabilities and backdoors in the system. Even if an attacker exploits homologous vulnerabilities in redundant executors to achieve common-mode escape, the adaptive negative feedback control mechanism is still able to iteratively change the operation scenario so that the attack cannot continue to act on the system, thus minimizing the impact of unknown attacks on the system's functionality and performance. Finally, this

principle requires systems to be able to adapt adaptively to unknown threats. For any multi-mode output vector inconsistency caused by unknown attacks, the DHR will change the current service scenario or executor construction scenario through policy scheduling and multi-dimensional dynamic reconfiguration negative feedback mechanisms until the multi-mode output vector inconsistency disappears.

5.4.3 The Principle for System Architecture Application

The core idea of ESS-empowered cyber resilience is “architectural empowerment,” i.e., applying the DHR architecture to the development and construction of various network equipment and services. The implementation of the DHR architecture requires the support of diverse execution environments. Theoretically, if the difference between two executors is large enough, it will be extremely difficult for any independent attack to take effect on two heterogeneous executors at the same time. Therefore, the more diverse and differentiated the executors in the system architecture, the stronger the overall heterogeneity and redundancy, the more comprehensive and efficient the defense against known/unknown security threats, and the more prominent the empowerment effect on cyber resilience.

The DHR architecture can be applied to all levels of the system, ranging from basic hardware to top-level applications. Especially in the case of top-level applications, due to differences in the type, operating mechanism, etc. of the devices or applications involved in the application targets, there are also certain differences in the implementation modes of the architecture.

1. Basic Hardware Platforms

For the basic hardware facilities which various network equipment and application systems rely on, COTS-class processors with different CPU instruction sets can be used, including but not limited to x86, ARM series processors, and domestic Sunway processors.

2. Operating Systems

For the operating systems which various network devices and application systems rely on, operating systems with different architectures can be used, including but not limited to the internationally mainstream Windows series, various Linux versions, and the domestically produced Kirin series, Deepin series, and Unitech series.

3. Virtualization

For the virtualized environment in which various types of network equipment and application systems run, different virtualization technologies can be used, including but not limited to KVM, Docker, Xen, VMware, etc.

4. Application Layer Software

Routing and switching software: For the routing and switching protocol software on which the operation of routers and switches relies, virtualized routers launched by Huawei, H3C, Cisco, and other companies can be used as

heterogeneous routing calculation executors, and open-source routing software such as FRRouting and VyOS can also be used as heterogeneous routing calculation executors.

Web server software: For the web server software on which the web application system relies, different web server products can be used as heterogeneous executors, including but not limited to international mainstream Apache, Nginx, Lighttpd, and domestically produced Tongtech, Kingdee, Primeton, etc. or COTS-level homologous artificial heterogeneous products based on mimic compilation technology, such as multiple versions software obtained by mimic compilation for Nginx.

Domain name resolution service software: For the network domain name protocol resolution function realized by the domain name resolution system, non-homologous heterogeneous domain name resolution software products are used to implement the mimic domain name server executor, including but not limited to multiple versions of self-developed software, as well as adapted open-source software, such as Bind, Unbound, PowerDNS, etc. Based on randomized implementation and diversified compilation techniques, the source or binary code of the above software is heterogenized to obtain homologous and heterogeneous software versions, which are used as a complementary implementation of the mimic domain name server executors.

Data persistence: For the data persistence storage layer on which the operation of various application systems relies, mimic technical means such as database command heterogeneity and data operation command merging can be used.

5.4.4 The Principle of Security Technology Collaboration

5.4.4.1 Security Technology Knowledge Ontology

Along with the rapid development of emerging network technologies such as 5G and IoT and the intense evolution of network attack and defense confrontation, defensive cyber security technologies continue to develop in breadth and depth, emerging cyber security technologies continue to emerge, traditional cyber security technologies continue to evolve and upgrade, the knowledge in the field of cyber security technology is increasingly complex, and the demand of relevant scholars and engineers for knowledge ontology in this field is becoming stronger.

To formulate a unified terminology for the formalization of cybersecurity technology domain and clarify the relationship between these terms on different levels, so as to form a common understanding in this field and thereby promote the exchange, sharing, and reuse of domain knowledge, academia and the industry have actively explored the knowledge ontology (hereinafter referred to as the security ontology) in the field of cybersecurity technologies.

In 2007, the first relatively complete security ontology was constructed by Almut [14] et al. The core concepts of this ontology include threats, assets,

Fig. 5.17 D3FEND knowledge graph



countermeasures, and vulnerabilities. Specifically, the ontology consists of 88 threat classes, 79 asset classes, 133 countermeasure classes, and 34 relationships between classes. It also provides methods for extending security ontology classes, ontology reasoning, and querying.

In 2009, Professor Wang Ju'an [15] and others integrated databases such as CVE, CVSS, CWE, CPE, and CAPEC to construct a formal knowledge model capable of capturing the relationship between vulnerabilities, products, and security technologies and assisting users in decision-making through a clear description of threat patterns and vulnerability management.

In June 2021, MITER released the D3FEND [16], a knowledge graph of cybersecurity countermeasures, which is the latest security ontology research result so far. Complementing ATT&CK, a knowledge base of cyber-attack tactics and techniques, this knowledge graph clarifies the key concepts of cyber defense technologies and their terminology and provides a comprehensive and clear categorization and compilation of the current mainstream defensive cybersecurity technologies.

As shown in Fig. 5.17, D3FEND organizes cyber defense technologies into a three-layer architecture: the defense tactics, basic technology, and defense technology.

As shown in Fig. 5.18, current mainstream cyber defense technologies can be abstractly summarized into five defense tactics: hardening, detection, isolation, deception, and eviction. Defense tactics can be further divided into basic technologies that focus on different network elements. For example, hardening tactics can be divided, according to the objective of hardening, into four basic technologies: application hardening, credential hardening, message hardening, and platform hardening. The defense technology is the specific cyber security technology under its own defense tactics and basic technology. For example, the basic technology of message hardening includes three specific defense technologies: message authentication, message encryption, and transmission agent identity authentication. Of course, there are also many different implementations of defense technologies. For example, message encryption can be implemented using a variety of encryption algorithms and protocols. Common encryption algorithms include DES, AES, and SM2.

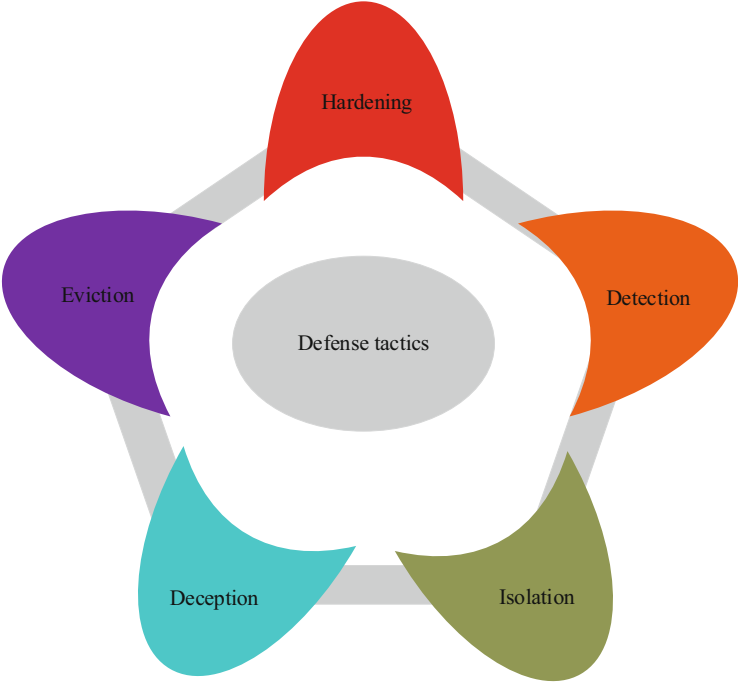


Fig. 5.18 D3FEND defense tactics

5.4.4.2 Collaborative Application of Security Technologies in the DHR Architecture

The principle of security technology collaboration refers to the principle of designing a collaboration framework or mechanism in order to achieve specific defense objectives and applying different defense tactics, basic technologies, defense technologies, and different engineering implementations of the same defense technologies to build a network defense system. Security technology collaboration consists of four levels: the level of defense tactics collaboration, the level of basic technology collaboration, the level of defense technology collaboration, and the level of defense technology engineering implementation collaboration. Different levels of security technology collaboration can be carried out under the DHR architecture, thereby achieving network resilience under generalized uncertain disturbances through the diversity, randomness, dynamics, and heterogeneous redundancy of security technologies.

Collaboration at the level of defense tactics refers to the collaboration between security technologies that serve different defense tactics. The defense objectives, defense scenarios, and defense mechanisms of the five defense tactics of hardening, detection, isolation, deception, and eviction are different. According to the DHR architecture, the defense tactics determined by the dynamic scheduling strategy

constitute a heterogeneous set of executors under the defense strategy. The input agent distributes input according to specific security technologies. Heterogeneous tactical executives perform security functions independently or interactively in parallel and deliver the outputs to the output agent for ruling. The ruling mechanism can adopt majority voting, multi-mode ruling, or iterative ruling based on specific security scenarios. Tactical-level security technology collaboration under a heterogeneous redundant architecture can, on the one hand, enhance the tactical variety, randomness, and dynamics of the cyber defense system, thus avoiding the blind spots and vulnerabilities brought by a single tactic and exponentially increasing the difficulty and cost of the attackers to break through the network defense system, and on the other hand, it can also help achieve the objectives of defense tactics through the collaboration of heterogeneous tactical executors by information sharing and other forms of collaboration, thereby achieving overall cyber security and resilience gains. For example, the dynamic scheduling strategy determines that the firewall and the intrusion detection system constitute a heterogeneous set of executors, thereby forming a tactical-level collaborative defense in which isolation and detection tactics are coordinated. The firewall is deployed between the internal and external networks to inspect the network traffic flowing through it according to the preset access control policy, intercept data packets that do not comply with the security policy, and prevent illegal access and intrusion. The intrusion detection system monitors network or system resources, looks for violations of security policies or signs of attacks, and gives alarms in real time. In an interactive and collaborative scenario, when the intrusion detection system detects an attack, it can notify the firewall to update the access control policy and prohibit the attacker's IP address or port, thereby making the firewall's access control policy more complete and effectively isolating the attacker from the internal network. The output agent will judge the outputs of the firewall and the intrusion detection system to assess network conditions and make a response accordingly.

Basic technology-level collaboration refers to the collaboration between different basic technologies that serve the same tactical objective. Different basic technologies under the same defense tactic focus on different network elements to achieve a unified tactical objective. From a tactical perspective, the basic technology-level heterogeneous executor set dynamically and randomly performs unified tactical functions using different network elements as entry points. Through the ruling mechanism, the output agent determines the results of the execution of defense tactics with more diverse and variable defense elements. The diversity, randomness, and dynamics of elements greatly increase the difficulty for attackers to bypass the defensive tactics. For example, in order to achieve the tactical objective of detecting cyber-attacks, current mainstream intrusion detection systems often comprehensively apply basic technologies such as file analysis, network traffic analysis, system platform monitoring, process analysis, and user behavior analysis to detect and analyze more elements in the network so as to improve the probability and efficiency of cyber-attack discovery and better accomplish the tactical objective of intrusion detection.

Defense technology-level collaboration refers to the collaboration between different defense technologies that serve the same tactic and belong to the same basic technology. Different defense technologies belonging to the same basic technology often implement the same defense tactic for the same target object from different technical perspectives. Defense technology-level collaboration achieves security gains through a variety of technologies. Under the DHR architecture, by dynamically and randomly adopting different defense technologies, the weaknesses of a single technology can be effectively compensated for, the implementation effect of defense tactics on specific defense elements can be enhanced, and cyber resilience can be improved. Taking message hardening under hardening tactics as an example, message authentication, message encryption, and transmission agent identity authentication are three specific defense technologies that belong to message hardening. All of them take network message as the target object and aim at enhancing its security, but they adopt different cryptographic algorithms and techniques from the perspective of integrity, confidentiality, and non-repudiation, respectively. Simultaneous use of message authentication, message encryption, and transmission agent identity authentication can more comprehensively prevent attacks on network transmission data and strengthen the security network messages in all aspects.

Collaboration at the level of defense technology engineering implementation refers to the collaboration between different engineering implementations of the same defense technology. The above three levels of security technology collaboration are more about achieving security gains from the perspective of diversity, while the defense technology implementation-level collaboration is to enhance cyber resilience through heterogeneous redundancy. Although different implementations of the same defense technology have exactly the same security functions, they have different software and hardware components. The resulting heterogeneous redundant structure can improve the system's resilience against random differential-mode failures, common-mode failures, as well as internal and external attacks through monitoring and cross-analysis mechanisms. Taking the honeypot technology as an example, different technical solutions can be used to realize a honeypot for the same device. Different technical solutions have different implementations of the protocol stack characteristics, port characteristics, operating system characteristics, and behavioral characteristics of the device. By deploying heterogeneous redundant honeypots, the tactical objectives of deceiving attackers and capturing attack behaviors can still be achieved in scenarios where attackers have successfully identified and bypassed some of the honeypots, thereby realizing cyber security and resilience gains.

5.4.5 The Principle of Strategic Ruling Feedback

Different from moving target defense, CMD organically integrates a variety of defense elements including dynamics, diversity, and randomness into a negative feedback control architecture. Therefore, cost-effective reliability and attack

resistance can be obtained by guiding the CMD implementation mechanism with a systematic control approach and systematic construction principles.

It is not difficult to find that the strategic ruling mechanism constitutes the DHR architecture with a generalized uncertain disturbance sensing function, which is the key link in CMD. It can detect abnormal or unusual behaviors of the executors and thus adjudicate feedback and policy scheduling to stop possible attacks. By making strategic rulings on multi-modal output vectors of given semantics, syntax, and even pragmatics on a standardized or normalizable mimic interface, any non-cooperative attack or random failure state reflected on the mimic interface can be effectively perceived. The arbitrator sends relevant status information to the feedback controller, which can form operation instructions for the input/output agent and the reconfigurable executors based on the given strategy or machine learning results. Introducing a strategy similar to mimic disguise into the strategy arbitrator can not only avoid the superposition state of majority-based ruling but also achieve richer functionality. For example, in order to improve the defense level, iterative ruling methods such as consistency comparison, weight judgment, and strategy parameters can be used, and the number objects and executors participating in the ruling can also be changed dynamically or randomly. When the ruling results indicate that all output vectors are different, and not in a seemingly “unmanageable” situation, spatio-temporal iterative ruling can be made by using the weights of confidence and historical performance of executors. The ruling based on the definition domain can be implemented when there are cases of semantically identical but syntactically different output vectors in multi-modal output vectors. When there is an undefined domain or optional domain or uncertain domain value or threshold allowable error in the output vector, the mask ruling method can be applied. In more complex cases, methods such as regular expressions can be used for matching ruling. In order to simplify the ruling complexity and accelerate the ruling speed, various preprocessing can be done on the executor output vector and its related data or status information (e.g., to reduce the time of comparing the IP grouping load fields or the output vector contents, the check digit or hash value for ruling can be formed in advance). In addition, by counting the abnormal output states of the multi-modal arbitrator and recording and analyzing the corresponding problem scenarios, the security situation of the target system’s software and hardware resources can be determined, and the effect of resisting non-coordinated attacks can be measured. It should be emphasized that mimic ruling can significantly increase the difficulty of side-channel attacks that use the arbitrator sharing mechanism to implement tunneling. Therefore, in the sensing function of strategic ruling, the redundant executor output voting technology is one of the core technical contents of fault-tolerant systems. Majority voting or consensus voting algorithms are usually used in isomorphic redundant systems, while heterogeneous redundancy supports cyber resilience and robustness because there may be differences in allowed executor output results. So far, there have been many studies on ruling algorithms. According to their execution methods and output strategies, the existing ruling algorithms are of the following types:

5.4.5.1 Ruling Based on Reliability Methods

Different voting algorithms are used in the field of reliability to obtain relatively accurate data as system outputs from redundant outputs of multiple versions of programs or less reliable hardware modules and thus have a wider range of applications.

Standard Ruling Algorithm

If one of the output results of each redundant executor is generated (selected or fused) as a system output, it can be categorized as a standard ruling algorithm, which mainly consists of two ways: result selection and result fusion:

1. Result selection. The data consistency comparison level includes traditional string comparison, bitwise comparison, data consistency ruling based on random sampling, and semantic similarity comparison. The multiple-choice result selection level includes but is not limited to consensus algorithm, majority ruling, plurality voting, median voting, etc. Based on majority ruling, a filtered ruling algorithm emerged, which incorporates acceptance tests into majority ruling. Such algorithms use various selection strategies to choose one path from the executor outputs as the system output.
2. Result fusion. The common one is the average value voting algorithm, whose output is the average of the redundant inputs. The weighted average value voting algorithm is an extension of the average value voting algorithm. This algorithm further calculates the weighted average of various redundant inputs with weights that can be pre-set or dynamically adjusted.

Hybrid Ruling Algorithm

Majority ruling or plurality ruling algorithms tend to output identical but incorrect redundant inputs as correct results. For example, the majority voting algorithm is obviously not applicable in scenarios where the semantics is the same but the syntax is different. Especially in some complex information systems, there are many functional expressions related to state, time, and environmental factors, so that even the output information of the same software/hardware entities may be inconsistent, making majority-based ruling almost impossible to be implemented. In this case, even if the majority-based outcome is a majority state, we cannot unreservedly believe in its given meaning. In order to improve the trustworthiness of the arbitrator, hybrid ruling algorithms often add some additional information to the standard ruling algorithm, such as the historical performance of each executor, vulnerability scores, and the confidence parameter statistics of the historical performance of each executor or scenario, thus dynamically reflecting the trustworthiness of the executors. These algorithms are summarized as follows:

1. Ruling based on probability and historical information. This type of algorithms uses information about the probability of correctness of the output of each executor, as well as the number of times the executor produces a correct result and other historical information to improve the reliability of the ruling output. Common algorithms of this type include maximum approximation voting, adaptive majority voting based on historical information, weighted voting based on historical information, etc.
2. Ruling based on the predictive technology. In some systems, there is some correlation between the output produced by the redundant executors in one cycle and the output produced in the next cycle. This correlation information can be used for predictive ruling algorithms. When the arbitrator senses any inconsistency in the executor outputs, the ruling algorithm can predict an output value based on historical information and compare it with each redundant output. Any redundant output can be used as the final output of voting as long as the difference between it and the predicted output is less than a predetermined prediction threshold, which is related to a specific application scenario. Commonly used predictive voting algorithms include the linear predictive voting algorithm, which uses the previous two voting outputs to predict the next result; the prediction-correction algorithm, which consumes more computing time but is more accurate; three-domain predictive voting, which is based on historical fault records to avoid selecting outputs from unreliable results when the outputs of the executors are inconsistent.

5.4.5.2 Decoding Mechanism Based on Error Correction Coding

Reasonable use of the error correction code mechanism in the ruling algorithm can obtain more accurate output results; improve the function of fault shielding, detection, and isolation; and reduce the complexity of communication. Attacks against target objects are usually not random events that occur independently. The attack effects are often deterministic due to the static nature, certainty, similarity, and “one-way transparency” of the target object and mathematically expressed as Boolean quantities rather than probabilistic values. This makes quantitative analysis of attack resistance difficult to characterize with the help of stochastic process tools. However, for DHR target systems with multimodal voting/coding mechanisms, firstly, the “True Relatively Axiom” axiom can transform deterministic or uncertain attacks against executor vulnerabilities, backdoors, and virus Trojans into probabilistic events in a certain distributional form at the system effect level. Secondly, by dynamically adjusting the adjudication policy/coding algorithms and changing the implementation algorithms of the executors or constructing the degree of dissimilarity, it is possible to change the probability magnitude and distribution pattern of the attack effect at the system level. Secondly, by dynamically adjusting the ruling strategies/decoding algorithms and changing the implementation algorithms or structural dissimilarity of the executors, it is possible to change the probability magnitude and distribution pattern of the attack effects at the system level. In

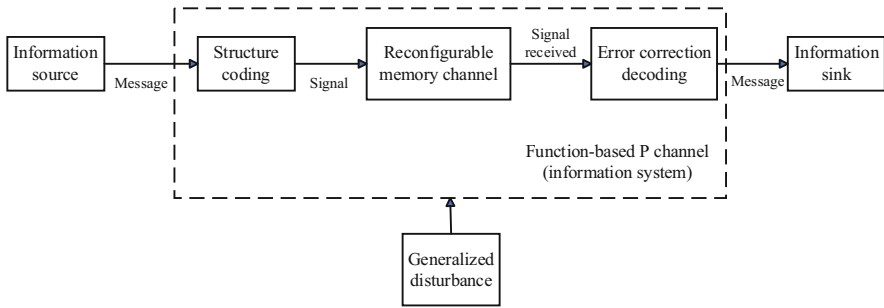


Fig. 5.19 Schematic diagram of the security defense model for general information systems

other words, when the target system adopts the DHR architecture with a multimodal voting/coding mechanism, both the attack events against individual executors and the random failures generated by executor components can be normalized into probabilistic reliability problems at the system level.

The fundamental problem of CESS can be described as how to correctly process and transmit information over a reconfigurable memory channel in the presence of non-random noise. The author believes that errors in information processing and transmission, reliability, and communication noise caused by network attacks can all be resolved through error correction coding. From this point of view, in the DHR architecture, encoding realizes policy distribution, while decoding is responsible for strategic ruling. In a general sense, various computers and information and communication devices in cyberspace can be regarded as some expression of a Turing machine. They can accept, store, and run a program that can be described by a Turing machine, in order to implement the described algorithm. Special attention should be paid to the fact that unlike the assumption of memoryless channels in the classic Shannon communication model, the DHR architecture can be abstracted as a reconfigurable memory channel with processing capabilities. Unlike the random noise assumed by Shannon, network attacks are obviously non-random and can be abstracted as generalized uncertain disturbances that include random communication noise, random physical failures, man-made attack noise, etc. Figure 5.19 is a schematic diagram of the security defense model for general information systems.

There are some mature decoding algorithms, such as maximum likelihood-based “hard” decoding, error control codes, and so on.

5.4.5.3 Distributed Consensus Mechanism

The ruling algorithms can be divided into two types: the centralized and the distributed, according to the way they are implemented. Both the above reliability-based method and the ruling method based on the decoding mechanism in error-correction coding can be used to implement centralized ruling as well as simple distributed ruling algorithms. The above-mentioned reliability-based methods and

the ruling methods based on the decoding mechanism in error-correcting coding can be used for centralized ruling as well as simple distributed ruling algorithms. However, these ruling algorithms are not specifically designed for distributed systems. Although they can support simple distributed architectures and tolerate crash faults such as partial node downtime, they cannot tolerate some nodes performing Byzantine attacks. Therefore, among the current new defense technologies, the blockchain is based on the Byzantine fault-tolerant technology to solve the problem of transmitting reliable information in unreliable networks. Due to the characteristics of the decentralized trust mechanism, increased cost of data tampering, and resistance to distributed denial of service, it has high research and application value in the field of network security. Based on the distributed system architecture, the blockchain technology needs to use the consensus mechanism to define fault-tolerant protocols in unreliable networks, reach a safe and reliable status consensus, and ensure that data forms a correct and consistent consensus across the entire network. The blockchain consensus mechanism is the core mechanism of the blockchain architecture, which fundamentally determines the security, availability, and performance of the blockchain, allowing distributed nodes to agree on the validity and consistency of block data, even if there are malicious Byzantine nodes among the nodes. Therefore, the consensus algorithm of the blockchain can be used as a representative of ruling algorithms in distributed systems.

According to the type of faults tolerated by the consensus mechanism, the blockchain consensus mechanism can be classified into the Crash Fault Tolerance (CFT) consensus mechanism and the Byzantine Fault Tolerance (BFT) consensus mechanism. The former considers the situation where a node fails but there is no response, while the latter considers Byzantine faults in which a node will forge or tamper with information to respond maliciously. The CFT consensus mechanism can only guarantee the reliability of the entire blockchain system in the case of crash faults such as any node going down in the blockchain system, but when any node in the system does something against the consensus such as tampering with the data, the algorithm will not be able to guarantee the reliability of the system. For this reason, the CFT consensus mechanism is mainly used in private chains. The current mainstream CFT consensus algorithms mainly include the Paxos algorithm and its derivative Raft consensus algorithm. The BFT consensus mechanism can guarantee the reliability of the blockchain system when any types of faults occur in the nodes in the blockchain system, as long as the nodes with crash faults/Byzantine faults are less than a certain percentage. Therefore, most blockchain systems (especially public chain systems) use the BFT consensus algorithm.

The basic process of the blockchain consensus is transaction packaging, block construction, bookkeeper selection, block verification, and blockchain update, among which the two core steps are bookkeeper selection and block verification. By bookkeeper selection strategy, the blockchain consensus mechanisms can be classified into five types: proof, election, random, consortium, and hybrid.

Proof-Based Consensus Mechanism

The proof-based consensus mechanism, also known as “the Proof of the X” consensus mechanism, requires miner nodes to prove that they have a particular capability in each round of consensus, which is usually accomplished by competitively completing a certain task. The task needs to be difficult to solve but easy to verify, and the miner node that wins the competition will become the bookkeeper. For example, the Proof of the Work (PoW) consensus mechanism is based on the miner’s computing power to complete the random number search task; the Proof of the Stake (PoS) consensus mechanism is based on the miner’s equity to complete the random number search task; the Proof of the Space (Proof of Space, PoSp) consensus mechanism is based on the size of the miner’s storage space to complete the random number search task; the Proof of the Elapsed Time (PoET) and Proof of Luck (PoL) consensus mechanisms are based on the miner’s trusted execution environment (a secure enclave created by Intel SGX technology) to generate random numbers, and the miner who generates the smallest quantity of random numbers will become the bookkeeper. Proof-based consensus mechanisms usually do not need to know the number of nodes participating in the election of bookkeepers and therefore can be applied to public chains that allow nodes to join and leave freely. However, the proof-based consensus usually consumes certain resources (such as computing power in PoW, storage space in PoSp) or requires specific environments or hardware (such as trusted hardware SGX in PoET and PoL). Proof of Stake can reduce the huge energy consumption caused by Proof of Work, but Proof of Stake also faces new attack threats, such as nothing-at-stake attack, grinding attack, and long-range attack, and stake bleeding attack.

Election-Based Consensus Mechanism

In each round of consensus, miner nodes elect the bookkeeping node for the current round by “voting,” and the first miner node to obtain more than half of the votes will obtain the right of bookkeeping. Distributed consensus algorithms in traditional distributed systems are mostly election-based consensus mechanisms, such as Paxos and Raft of the CFT type. Because these algorithms are crash fault-tolerant, they are difficult to be applied to blockchain scenarios, especially public chain scenarios. Most of the election-based consensus mechanisms used in blockchain scenarios support Byzantine fault tolerance, such as PBFT, SBFT, and MinBFT. The election-based consensus mechanism can achieve high transaction throughput, timeliness, and consistency, but it needs to be informed in advance of all miner nodes participating in the election or voting. Message complexity is proportional to the square of the number of nodes, and the master node tends to be the bottleneck of the system. Therefore, it is mostly used in permissioned chains, such as private chains or consortium chains.

Random Consensus Mechanism

Miner nodes use specific software functions or hardware to randomly determine the bookkeeping nodes for each round. For the sake of security, random consensus mechanisms often use cryptographic tools to ensure that the randomization process cannot be predicted or controlled by malicious nodes. There are usually two ways to implement random consensus mechanisms. One is to use a trusted execution environment based on Intel SGX to ensure the security of the consensus mechanism, such as PoET and PoL consensus mechanisms. The other way is to protect the security of the consensus mechanism through cryptographic tools, such as Verifiable Random Function (VRF) and BA* Protocol (Binary Byzantine Agreement Protocol). Examples of such consensus mechanisms are Algorand and Tangaroa consensus mechanisms. The advantages and disadvantages of this type of consensus mechanisms are described in the proof-based consensus mechanism and the hybrid consensus mechanism.

Consortium Consensus Mechanism

In consortium consensus mechanisms such as DPoS (Delegated Proof of Stake), Ripple, Elastico, PoA (Proof of Authority), Pala, and Thunderella, miner nodes first elect a group of representative nodes in a particular way, and then these representative nodes take turns to obtain the bookkeeping right or determine the right through election. One type of consortium consensus mechanisms combines the classic distributed consensus algorithm with the non-permissioned consensus to achieve representative node election through non-permissioned consensus (e.g., PoW, PoS, etc.), and then the representative nodes execute the classical distributed consensus algorithm. This method has the characteristics of strong consensus, the probability of blockchain bifurcation is small, and transactions can be confirmed quickly. However, this type of consortium consensus mechanism faces problems such as protocol initiation, committee reconfiguration, and new node joining. Another type of consortium consensus mechanism is that all nodes must undergo identity registration before participating in the consensus. This type of consortium consensus mechanism is also known as the permissioned consensus mechanism. The permissioned consensus mechanism is mainly applicable to consortiums formed between enterprises/organizations and can achieve higher transaction throughput rates.

Hybrid Consensus Mechanism

Hybrid consensus mechanisms refer to the miner node adopting a mixture of multiple consensus algorithms to select bookkeeping nodes, such as the PoA (PoW + PoS) consensus, Tendermint (PoS + PBFT), Tangaroa (Raft+PBFT), and Algorand (PoS + BFT) hybrid consensus. Hybrid consensus protocols usually combine classical distributed consensus algorithms with proof-based consensus

mechanisms, so that they can have strong consensus and adapt to the open public chain environment.

It should be pointed out that regardless of the ruling algorithm and execution method, it is necessary to analyze the confidence parameters of the historical performance of each executor or scenario. Invoking these parameters in the iterative ruling process is equivalent to constructing a negative feedback mechanism in the spatio-temporal dimension, so that a single majority-based ruling adds iterative significance that is strongly correlated with historical performance. In addition, in view of the complex adversarial requirements of the attack-defense game, engineering implementation often introduces weighted parameters such as current security situation, system resource status, frequency of attacks, and historical performance of the executors to participate in the judgment on the basis of the ruling, or provides alternative and diverse voting strategies for different situations, or even directly utilizes the early warning or alarm information of the detection device to make the selection. This approach is called multimode ruling (MR). Among them, (1) multimode ruling not only has a variety of ruling algorithms but also has a spatio-temporal iterative convergence mechanism that calls the corresponding algorithm; (2) the object of multimode ruling can be a file block, a data packet, or a hash value or checksum generated by the former or various data and status information that are strongly related to it.

In addition to mimic ruling, another important link is feedback control. The abnormal status output of the arbitrator will trigger a change in the degree of heterogeneity or variety of the defense scenarios within the current service set of the mimic bracket, thereby introducing dynamics. This process is accomplished by feedback control through two operation stages. One is replacing and cleansing the abnormal executor elements within the service set, or reconfiguring and reorganizing the software and hardware components or algorithms of the abnormal executor, or forcing the abnormal executor to change its operating state (e.g., to enter the restart/reconfiguration state, etc.), or reorganizing the executor elements within the service set. The other is observing whether the abnormal state of the arbitrator is eliminated or the anomaly frequency is lower than the given threshold after the above operation. If the requirements are not met, the operation of the first stage is repeated until the requirements are met; otherwise the feedback loop enters a temporary stable state to provide system robustness. Finally, feedback control can introduce AI algorithms to achieve accurate perception of executor status and network situation, implement precise cyber security protection, reduce processing time and cost, and improve processing efficiency. It can be seen that the target object's defense environment is dynamic, diverse, and random simultaneously, and the operation process adopts the method of adaptive iterative convergence. Therefore, the security effect of CMD can be designed, planned, tested, and measured.

On the whole, strategic ruling feedback control generally follows two principles:

1. Introducing randomness. Human attacks (based on vulnerabilities or backdoors) are non-random disturbances, so the resulting errors or failures are also subject to non-random distribution. The introduction of system dynamics through control

enables the stochastic transformation of human disturbances, i.e., the conversion of non-random disturbances or errors such as human attacks into random errors and the conversion of non-random attacks into some form of probabilistic events at the system level, a process that can be described with the help of a stochastic process tools. Both errors/failures resulting from cyber-attacks on individual executors and functional safety errors/failures resulting from the activation of reliability failures in the components of an actuator can be normalized at the system level as reliability problems of a probabilistic nature.

2. Improving control efficiency. First of all, from the perspective of robust control, most security events can be considered as model perturbations of generalized uncertain disturbances induced by ESS problems of the target object. Robust control mechanisms based on strategic ruling feedback are capable of controlling model perturbations within a given threshold when the model architecture is certain and the disturbance range is known. Therefore, introducing the robust control mechanism into the heterogeneous redundant structure can endow the latter with ESS properties based on uncertainty effects. This largely solves the problem of the lack of stability of classical heterogeneous redundant structures in response to trial-and-error or coordinated attacks. Secondly, through machine learning or AI algorithms, it is possible to accurately identify and perceive the abnormal status of the arbitrator, accurately implement the processing strategy, shorten the processing time, and improve the processing efficiency.

5.5 Summary of This Chapter

This chapter first analyzes the powerful enabling advantages of ESS for the development of cyber resilience from four aspects: the thinking perspective, theoretical basis, architectural guidance, and evaluation innovation. It points out that ESS empowerment can effectively address the major deficiencies of the previous cyber resilience thinking that merely emphasizes after-event recovery, the severe lack of dominant architectural guidance, and the major challenges of the lack of core competence assessment metrics. Secondly, it specifically analyzes the incremental capabilities that can be obtained through ESS empowerment in all dimensions of cyber resilience in terms of the four top-level goals of prevention, resistance, recovery, and adaptation, as well as the eight development objectives. Subsequently, based on the technology tree model, the root technology, branch technology, and leaf technology for the development of ESS-enabled cyber resilience engineering are given layer by layer using a hierarchical approach. Finally, the design principles for ESS-enabled cyber resilience are proposed and analyzed. It can be seen from this chapter that ESS can empower cyber resilience from multiple dimensions such as the target capability, technical approach, and design principles. In particular, the DHR architecture of ESS provides the overarching architecture for many technical approaches and design principles of existing network resilience and plays the role of “steel skeleton.” The endogenously safe and secure DHR architecture is a vivid

example of security-by-design on the manufacturing side. In particular, the author emphasizes that security-by-design does not mean blindly piling up various security measures in the system. Scientifically speaking, security-by-design should follow the principle of “structure determines nature,” be based on innovative system architecture design and be supported by collaborative application of security technologies under the traction of the system architecture. ESS-empowered cyber resilience can effectively respond to “unknown unknown” network attacks and guarantee the continuous and trustworthy operation of key services, thus ensuring the achievement of the objective of “mission assurance” of cyber resilience.

References

1. Ross R, Pillitteri V, Graubart R, et al. Developing cyber-resilient systems: a systems security engineering approach NIST Special Publication 800-160, vol. 2, Rev. 1. [EB/OL]. 2021. <https://doi.org/10.6028/NIST.SP.800-160v2r1>.
2. Wu Jiangxing. Research on cyber mimic defense [J]. J Cyber Secur. 2016;1(4):1–10.
3. Wu Jiangxing. Development paradigms of cyberspace endogenous safety and security [J]. Sci Sin Inf. 2022;52:189–204.
4. Purple Mountain Laboratory. Technical specification for generic log representation in the field of mimic defense in cyberspace [R]. Nanjing, 2022.
5. Zhou ZH. A brief introduction to weakly supervised learning [J]. Natl Sci Rev. 2018;5(1):44–53.
6. Knudsen EI. Supervised learning in the brain [J]. J Neurosci. 1994;14:3985–97.
7. Xie T, Grossman JC. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties [J]. Phys Rev Lett. 2018;120:145301.
8. Zheng X, Zheng P, Zhang R. Machine learning material properties from the periodic table using convolutional neural networks [J]. Chem Sci. 2018;9(44):8426–32.
9. Louis SY, Zhao Y, Nasiri A, et al. Graph convolutional neural networks with global attention for improved materials property prediction [J]. Phys Chem Chem Phys. 2020;22(32):18141–8.
10. Hochreiter S, Schmidhuber J. Long short-term memory [J]. Neural Comput. 1997;9(8):1735–80.
11. Xia Huili, Zhang Jin, Jiang Yiming, et al. A method for scheduling the executor of a mimic router and the mimic router [P]: China 111431946 A. 2020. 07.
12. Adams M, Bearly S, Bills D, et al. An introduction to designing cloud services for reliability [R]. Microsoft Corporation White Paper. 2012.
13. Gartner. Build adaptive security architecture into your organization [EB/OL]. 2017. <https://www.gartner.com/smarter-withgartner/build-adaptive-security-architecture-into-your-organization>
14. Herzog A, Shahmehri N, Duma C. An ontology of information security [J]. Int J Inf Secur Priv. 2007;1(4):1–23.
15. Wang JA, Guo M. OVM: an ontology for vulnerability management [C]. In: Proceedings of the 5th annual workshop on cyber security and information intelligence research: cyber security and information intelligence challenges and strategies, New York, 2009. p. 1–4.
16. Kaloroumakis PE, Smith MJ. Toward a knowledge graph of cybersecurity countermeasures [R]. The MITRE Corporation, McLean, Virginia, US, 2021.

Chapter 6

Metrics and Assessment of ESS-Empowered Cyber Resilience



6.1 A New Perspective on the Assessment of Cyber Resilience Empowered by the ESS Architecture

Existing cyber resilience assessment evaluates the cyber resilience capability of a system from the perspective of technical methods, and there are problems such as the lack of representative core indicators, the lack of comparability of evaluation results, and the difficulty in assessing emergent system capabilities. The DHR architecture of ESS acquires emergent ESS capability through system construction. In a sense, this is a vivid manifestation of the natural law that “structure determines nature” in the field of cyberspace security and safety. The above facts inspire us to focus on and evaluate system architecture capabilities in the cyber resilience assessment process, rather than limiting ourselves to evaluating various fragmented technical approaches. Inspired by the DHR architecture, this chapter proposes a system architecture-based cyber resilience assessment perspective, in which assessing the core system architecture capabilities is the key to cyber resilience assessment.

6.1.1 Current Perspectives on Cyber Resilience Assessment

The present cyber resilience engineering [1] can be represented by the “two-pyramid” structure shown in Fig. 6.1. The “pyramid” on the left represents the goal domain, which mainly answers the question of “what is cyber resilience” and consists of four layers: goals, objectives, sub-objectives, and capabilities. Goals include prevention, resistance, recovery, and adaptation, expressing the top-level connotation of cyber resilience. Goals are decomposed and refined into objectives, which are detailed in Chap. 4 and will not be repeated here. Goals and objectives present a many-to-many mapping relationship. Objectives are further decomposed into sub-objectives, the two showing a one-to-many mapping relationship. The

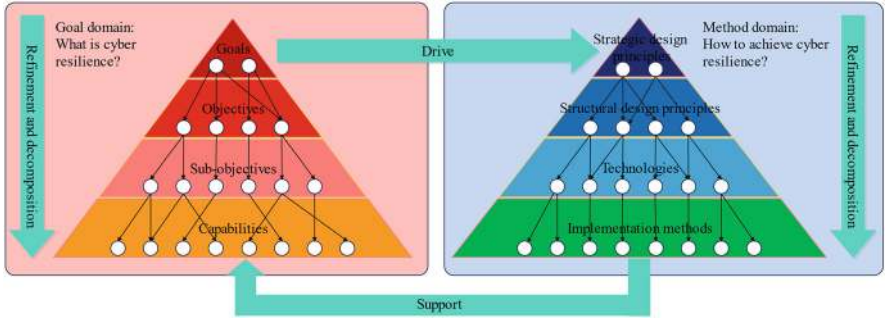


Fig. 6.1 The “two-pyramid” structure of present cyber resilience engineering

realization of sub-objectives relies on the support of specific capabilities, and there is a many-to-many mapping relationship between sub-objectives and capabilities. The “pyramid” on the right side of Fig. 6.1 represents the method domain, which mainly answers the question of “how to achieve cyber resilience.” From the top to the bottom, the method domain is divided into four levels: strategic design principles, structural design principles, technologies, and implementation methods. For detailed explanations of each level, see Chap. 4 of this book. There is a many-to-many mapping relationship between strategy design principles and structural design principles, and between structural design principles and technologies, and there is a one-to-many mapping relationship between technologies and implementation methods. The acquisition of capabilities needs to be supported by relevant implementation methods, and there is also a many-to-many mapping relationship between capabilities and implementation methods.

The working principle of the “two-pyramid” structure of cyber resilience engineering is to firstly refine the domain of objectives and determine the required capabilities; then, according to the mapping relationship between the capabilities and the implementation methods, inversely deduce the technical methods and design principles to be applied. Different technical methods may conflict and cannot be applied at the same time. In this case, it is necessary to determine the design principles that need to be applied first according to the priority of the goals and objectives, as well as the mapping relationship between goals, objectives, and design principles, and then clarify the technologies that should be applied on a priority basis.

The current cyber resilience assessment adopts an assessment perspective based on technical methods to evaluate the degree of realization of cyber resilience goals and objectives [2, 3]. The assessment process framework is shown in Fig. 6.2, and the assessment steps follow the numerical sequence labeled in the figure. The first step is to prioritize each goal of cyber resilience based on the organization’s overall risk management strategy, prioritize each objective, sub-objective, and capability, and determine the relevance of each technology and method according to the priority of each goal as well as the mapping relationships of the elements in the domains of cyber resilience goals and methods as shown in Fig. 6.1. The second step is to assess

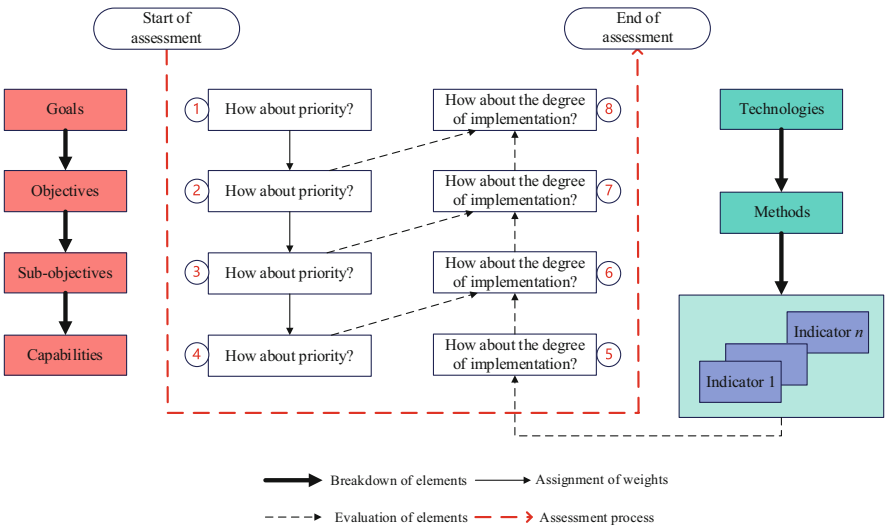


Fig. 6.2 Framework of current cyber resilience assessment process

the degree of realization of each specific capability. Current cyber resilience assessment methods define a set of evaluation metrics for each capability from the perspective of technical methods that support capability realization [2, 3], and these metrics can be used to evaluate the degree of realization of specific capabilities. The third step is to assess the degree of achievement of each sub-objective based on the results of assessment of the realization degree of capabilities and the priority of capabilities. The final step is to assess the degree of realization of the goals and objectives in sequence based on the degree of realization of the sub-objectives, using a bottom-up approach similar to the above-mentioned one, as shown in Fig. 6.2. It can be seen that the existing cyber resilience assessment starts from the bottom of the goal domain and method domain, adopts a bottom-up sequence, and evaluates the cyber resilience of a system from the perspective of technical methods by analyzing the degree of realization of each goal of cyber resilience.

6.1.2 A Cyber Resilience Assessment Perspective Based on the System Architecture

There are several shortcomings in the existing cyber resilience assessment perspective based on technical methods. First, the evaluation indicators oriented to technical methods are cumbersome and trivial, lacking representative core indicators. The existing cyber resilience engineering framework provides 14 technologies and 50 technical implementation methods [1]. For the implementation methods of each technology, the current cyber resilience assessment framework provides more than

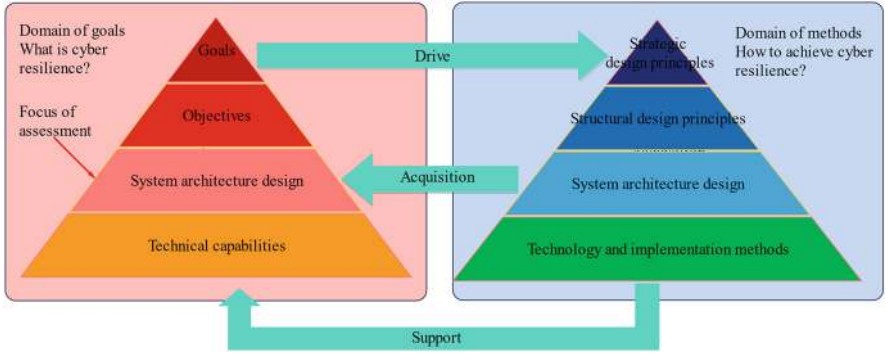


Fig. 6.3 The “Two-pyramid” structure of cyber resilience engineering based on system architecture

500 evaluation indicators [3]. However, in the specific implementation process of cyber resilience assessment, it is difficult for the evaluator to select a core indicator set that can represent the key capabilities of the system’s cyber resilience, and the operability of the assessment framework is poor. Secondly, the indicator weights are highly subjective and the evaluation results lack comparability. During the application process of the existing cyber resilience assessment framework, it is necessary to determine the priority weight of each goal, objective, sub-objective, capability, technology, method, and indicator. Since different organizations have different risk management strategies, the weighting system is set with significant differences, resulting in a lack of horizontal comparability of the assessment results. Finally, it is difficult to assess emergent system capabilities. From the perspective of the system theory, the whole is greater than the sum of its parts. Higher-level emergent capabilities of a system often cannot be pieced together by separately assessing the capabilities of lower-level components of the system. The assessment of the overall cyber resilience of a system can only be accomplished by viewing the system as a whole and considering the collaborative or synthetic relationship between different components of the system.

Inspired by the DHR architecture’s ability to obtain emergent capabilities through system construction, this book proposes a system architecture-based cyber resilience assessment perspective, which focuses on evaluating system architecture capabilities to solve the problems in the existing cyber resilience assessment. The “two-pyramid” model of cyber resilience engineering based on the system architecture is shown in Fig. 6.3, whose core point is that the realization of cyber resilience goals and objectives requires the support of system architecture capabilities, and the latter must be obtained through system architecture design, as shown in Fig. 6.3.

System architecture-based cyber resilience assessment focuses on system architecture capabilities and evaluates system architecture cyber resilience capabilities by defining the key properties, core capabilities, and main indicators of the cyber resilience system architecture (see Sect. 6.2 for details). First of all, system architecture capabilities can be used as the core indicator to measure system cyber

resilience, which solves the problem of the existing cyber resilience assessment that “the technical method-oriented evaluation indicators are complicated and trivial, and lack representative core indicators.” Secondly, the system architecture capability is an objective technical capability that is independent of organizational risk management strategies. The assessment results can be compared horizontally, which solves the problem of “strong subjectivity of indicator weights and lack of comparability of assessment results” in the existing cyber resilience assessments. Finally, the system architecture capabilities include the emergent capabilities of the system, which solves the problem of “difficulty in assessing emergent system capabilities” in the existing cyber resilience assessments.

It should be particularly pointed out that the system architecture-based assessment perspective is inspired by the phenomenon of emergent capabilities of the DHR architecture, and the assessment perspective itself is a direct reflection of the cyber resilience empowered by the DHR architecture of ESS.

6.2 System Architecture-Based Cyber Resilience Assessment System

6.2.1 Assessment Framework

An excellent system architecture ensures that the network system can perceive threat risks in due time, robustly resist threat attacks, promptly respond to and resolve system disturbances, restore its functions within an acceptable time, and intelligently adjust and adapt according to changes in threats. Inspired by cyber resilience strategic design principles, structural design principles, and the DHR architecture of ESS, Table 6.1 gives a cyber resilience assessment framework based on the system architecture. The framework covers threat situation perception capability, threat prevention and resistance capability, threat damage recovery capability, and threat change adaptation capability, each of which includes several assessment points and key assessment items. Among them, assessment items such as unknown threat perception, threat accumulation elimination, unknown attack resistance, and generalized functional safety (GFS) are new mandatory assessment items inspired by the DHR architecture of ESS.

The specific explanations of each key assessment item are detailed in Table 6.2 through Table 6.5, including the general meaning of the key capabilities of each system architecture as well as the description of the unique capabilities provided by the DHR architecture. Threat situation perception capability refers to the system’s ability to perceive, analyze, and identify various adverse conditions, especially potential man-made attacks. Threat situation perception capability includes two key points: “threat data exploitation” and “threat event perception.” The former refers to the ability to record and analyze various threat data, while the latter refers to the ability to perceive various anomalies or attack events. In particular, the

Table 6.1 Cyber resilience assessment framework based on a system architecture

Content of assessment	Assessment points	Key assessment items
Threat situation perception capability	Threat data exploitation	Threat data logging
		Threat backtracking analysis
	Threat event perception	Known threat perception
		System anomaly perception
Threat prevention and resistance capability	Attack risk reduction	Unknown threat perception
		Threat accumulation elimination
		Attack surface shrinking
	Threat damage defense	System camouflage
		Known attack defense
Threat damage recovery capability	Resource damage limitation	Unknown attack defense
		Generalized functional safety
	Damaged resource recovery	Modular system construction
		System component isolation
		Abnormal component replacement
Threat change adaptation capability	Intelligent system reconfiguration	Use of heterogeneous redundant components
		Environmental change perception
		Targeted component replacement
		Resources upgradable

perception of unknown threats is the core assessment content to ensure that the system has cyber resilience, as shown in Table 6.2.

Threat prevention and resistance capability refer to a system’s ability to prevent and resist human attacks and random failures. It includes two key points: “attack risk reduction” and “threat damage resistance.” “Attack risk reduction” refers to minimizing the possibility of the system being attacked through methods such as shrinking, camouflaging, jumping, and rolling back; “threat damage resistance” refers to the system being able to withstand attacks or failures when they actually occur, and not to be “broken in a single blow,” and to have good network resilience. The above capabilities are further subdivided into multiple key assessment items, as shown in Table 6.3.

Threat damage recovery capability refers to the system’s ability to contain the scope and extent of damage when it occurs, by, for example, limiting the damage to specific components or subsystems, and the ability to identify and replace damaged resources. Threat damage recovery capability includes two key points: “resource damage limitation” and “damaged resource recovery.” The former refers to the ability to limit damage to a specific subsystem or component, and the latter refers to the ability to quickly identify and repair or replace damaged resources. See Table 6.4 for detailed explanations of each key assessment item.

Threat change adaptation capability refers to the system’s ability to perceive changes in the external threat environment and make adaptive adjustments to the

Table 6.2 Detailed explanation of each key assessment item of threat situation perception

Key assessment item	Explanation
Threat data logging	• General explanation The system records and saves system inputs, resource status, user operation commands, abnormal operation events, and other information during operation to support threat backtracking analysis • Explanation of unique capabilities of the DHR architecture DHR architecture’s standardized logs, including executor scheduling records, ruling exception logs, etc. in addition, the DHR architecture can record snapshots of the current operating environment and related information when there is a ruling abnormality, and there is no problem of false alarms or false alarm information
Threat backtracking analysis	• General explanation Analyzing and processing the recorded historical threat data to reproduce the process of the attack, locate the source of the attack, and identify the attacker • Explanation of unique capabilities of the DHR architecture The DHR architecture can use operating environment snapshots and related operating records to trace back the causes and mechanisms of differential-mode disturbances, discover system instability and failure factors as well as security defects or unknown virus Trojans, perform collaborative analysis on standardized logs output by multiple DHR devices or systems, and also can discover new resources and means of attacks such as “0-day” and APT attacks
Known threat perception	• General explanation The system’s ability to perceive cyber-attacks with known characteristics, methods, and objects • Explanation of unique capabilities of the DHR architecture Mechanistically, DHR does not rely on various security knowledge bases based on prior knowledge to perceive either known or unknown threats. Although there are various known vulnerabilities and backdoors, virus Trojans, and attack signature databases on the internet, due to various reasons, it is difficult for application systems to obtain patches or updates in real time, as a result of which there are “N-day” threats. In the DHR architecture, all differential-mode disturbances can be perceived in real time, there is no distinction between known and unknown threats, and the security of the system is weakly related to or irrelevant to the real-time nature of maintenance work. It should be pointed out that DHR does not exclude prior knowledge. The DHR architecture can naturally absorb and utilize traditional threat perception information based on prior knowledge, narrow the “blind zone” of strategic ruling, and improve the accuracy of the system’s threat perception
System anomaly perception	• General explanation The system’s ability to perceive various abnormal events such as abnormal service performance, abnormal resource usage, abnormal user operations, abnormal input and output, and abnormal physical environment • Explanation of unique capabilities of the DHR architecture Through the “structural encryption” effect (see Sect. 3.6 for details), GFS events such as known threat perturbations, unknown threat disturbances, and random failures are uniformly transformed into ruling problems expressed in the differential-mode form in the DVR domain

(continued)

Table 6.2 (continued)

Key assessment item	Explanation
Unknown threat perception	• General explanation The system’s ability to perceive cyber-attacks whose characteristics, methods, and objects are unknown • Explanation of unique capabilities of the DHR architecture The DHR architecture’s structural encryption mechanism allows differential-mode disturbances of any executor within the current operating environment to be detected in real-time, independent of the known or unknown nature of the threat. No other current system architecture’s ability to perceive unknown threats is consistently robust.

system to reduce the risk of being harmed according to the current environmental situation. The ability to adapt to threat changes is mainly reflected in the intelligent reconfiguration of the system. See Table 6.5 for detailed explanations of each key assessment item.

6.2.2 Metrics

In order to assess the system architecture-enabled cyber resilience capability, this subsection, based on the assessment framework given in Sect. 6.2.1, gives the representative metrics of threat situation perception capability, threat prevention and resistance capability, threat damage recovery capability, and threat change adaptation capability. When introducing each capability metric, the general metrics are given first, followed by the specific metrics of the DHR architecture. The specific metrics of the DHR architecture are mainly used to measure the unique capabilities of the DHR architecture’s cyber resilience and are the specific explanations of general metrics in the context of the DHR architecture. Generally speaking, the specific capability metrics of a certain architecture should be used to supplement, enhance, or refine the general architecture metrics, rather than weaken or simplify them.

The metrics for each key capability of threat situation perception are shown in Table 6.6. The metrics for each key assessment item of threat prevention and resistance are shown in Table 6.7. The metrics for each key assessment item of threat damage recovery are shown in Table 6.8. The metrics of each key assessment item of threat change adaptation are shown in Table 6.9. In addition to the metrics listed in Table 6.9, threat change adaptation also includes environmental change perception capability, which can be manifested by the ability to perceive known threats, unknown threats, and system anomalies. The specific metrics corresponding to known threat perception, unknown threat perception, and system anomaly perception are detailed in Table 6.6 and will not be repeated here.

Table 6.3 Detailed explanation of each key assessment item of threat prevention and resistance capability

Key assessment item	Explanation
Threat accumulation elimination	• General explanation The system has the ability to prevent attackers from gaining a long-term and stable foothold within the system. For example, the moving target defense (MTD) architecture can eliminate threat accumulation to a certain extent through continuous system or component hopping. • Explanation of unique capabilities of the DHR architecture The DHR architecture does not pursue the real-time exhaustion or elimination of all ESS problems in the architecture. Instead, it adopts SR-FC and intelligent scheduling mechanisms and the structural encryption method of “one differential mode, one reconfiguration” to eliminate the influence of threatening factors causing differential-mode perturbation in the current operation environment in real time
Attack surface shrinking	• General explanation The system has the ability to shrink the attack surface, with an effect of “$1 + 1 < 1$.” Achieving cyber resilience inevitably requires diversity and redundancy, but simply applying redundancy and diversity-related technologies will lead to an expansion of the system attack surface [1]. Therefore, reducing the attack surface through system architecture design is one of the key capabilities to achieve the objectives of cyber resilience • Explanation of unique capabilities of the DHR architecture Theoretically, the DHR’s structural encryption mechanism shrinks the attack surface of the system to the “mimic bracket” component. Compared with the complex security situation within the executors or the current operating environment, the mimic bracket component is functionally simple and can be easily verified using formal methods. The “double-blind effect” produced by SR-FC dynamic structural encryption can completely offset any “one-way advantage” of the attacker, and effectively avoid the problem that “diversity and redundancy will lead to an expansion of the system’s attack surface.”
System camouflage	• General explanation The externally presented characteristics of the system are deceptive and do not directly reflect the internal characteristics of the system • Explanation of unique capabilities of the DHR architecture MDF and “observer effect” are unique to the DHR architecture. The cryptographic barrier formed by structural encryption puts the attacker in the dilemma where “there is no better method than trial-and-error or blind attack.” the “observer effect” makes trial-and-error or blind attack mechanically impossible to achieve the purpose of “entropy reduction.”
Known attack defense	• General explanation The system’s ability to resist cyber-attacks with known characteristics, methods, and objects • Explanation of unique capabilities of the DHR architecture The ability and effectiveness of the DHR architecture to defend against attacks is mechanistically independent of the known or unknown nature of the attack, and does not rely on (but does not exclude) any intrusion prevention techniques based on prior knowledge

(continued)

Table 6.3 (continued)

Key assessment item	Explanation
Unknown attack defense	• General explanation The system's ability to resist cyber-attacks whose characteristics, methods, and objects are unknown • Explanation of unique capabilities of the DHR architecture To date, this capability is unique to the DHR architecture. Some system architectures such as dissimilar redundancy structure (DRS) do not have stability robustness although they can resist unknown attacks within a certain timeframe
Generalized functional safety	• General explanation The system adopts a unified architecture and mechanism to deal with (known and unknown) network attacks and random failures. This architectural capability is particularly critical for cyber-physical systems • Explanation of unique capabilities of DHR architecture GFS capabilities are specific to the DHR architecture. Due to its static nature and deterministic mechanism, dissimilar redundancy structure (DRS) is not resistant to common-mode attacks or trial-and-error attacks, and does not have stability and quality robustness

6.2.3 System Architecture Scoring

Evaluators can score the system architecture cyber resilience capabilities based on the system architecture cyber resilience assessment framework and metrics. The assessment of each specific metric can be obtained through various assessment methods described in Sect. 6.3 (including static assessment, adversarial assessment, disruptive assessment, etc.). After completing the scoring of each specific metric, the system architecture cyber resilience score can be comprehensively calculated as a quantitative measure of the system architecture cyber resilience capability.

The system architecture cyber resilience score adopts a step-by-step weighted score calculation method. First, each assessment item in the system architecture cyber resilience evaluation framework is scored on a scale ranging from 0 to 5, as shown in Table 6.10. The scoring of an assessment item depends on the assessment results of its relevant quantitative metrics. The metric scores included in each capability in the system architecture cyber resilience evaluation framework are weighted and summarized to obtain the score for that capability. Table 6.11 gives the scoring weight of each evaluation point and key capability in the system architecture cyber resilience evaluation framework.

Based on the score of each key capability under each evaluation point, the scores of each evaluation point are weighted by Eq. (6.1), and the scores range from 0 to 50.

$$S_i = 50 \times \left(\sum_j W_{ij} R_{ij} \right) / \left(\sum_j 5 W_{ij} \right) \quad (6.1)$$

In the Eq. (6.1), i is the number of evaluation points; j is the number of key capabilities within evaluation point i ; S_i is the score of evaluation point i ; W_{ij} is the

Table 6.4 Detailed explanation of each key assessment item of threat damage recovery capability

Key assessment item	Explanation
Modular system construction	• General explanation The system adopts a low-coupling and high-cohesion modular structure, and the modules adopt standardized interfaces
System component isolation	• General explanation The subsystems and functional modules in the system are isolated from each other, and the communication between components adopts a whitelist method with minimum authorization, making it difficult for attackers to penetrate into other components through compromised components • Explanation of unique capabilities of DHR architecture The DHR architecture prohibits direct communication between executors in the current operating environment, and communication within the system must pass through a trusted intermediary (mimic bracket component), which provides a good architectural paradigm for system component isolation. ESS white-box testing, which fully exposes a small number of executors to the attacker, is to observe whether the attacker can move laterally within the current operating environment and take control of the entire system under these conditions
Abnormal component replacement	• General explanation Replacement of abnormal components in the system • Explanation of unique capabilities of DHR architecture It can realize the cleansing, recovery, reconfiguration, and reorganization of abnormal executors without affecting service provision, or even replace the current operating environment without being sensed by services
Use of heterogeneous redundant components	• General explanation Functionally equivalent heterogeneous redundant components are used. Each heterogeneous component follows the same external interface and behavioral specifications to facilitate replacement • Explanation of unique capabilities of DHR architecture According to the DVR complete intersection principle, the DHR architecture essentially belongs to the complete intersection of dynamics, variety (diversity), and redundancy. It has been theoretically demonstrated that increased redundancy and heterogeneity can lead to exponential gains in system security

weight of key capability j within evaluation point i ; R_{ij} is the score of key capability j within evaluation point i .

The evaluation points of the system architecture’s cyber resilience capability comprehensively reflect the baseline cyber resilience capability that the system possesses. If the score of a certain evaluation point of the system architecture is too low, it means that even if the system applies more resilience technologies in other aspects, it will still be difficult to fundamentally improve the system’s level of cyber resilience assurance for basic services. Figure 6.4 illustrates the bottom line of system architecture’s cyber resilience capability. After calculating the scores of

Table 6.5 Detailed explanation of each key assessment item of threat change adaptation capability

Key assessment item	Explanation
Environmental change perception	• General explanation The system can sense changes in the external environment • Explanation of unique capabilities of the DHR architecture The DHR architecture enables better perception of changes in the threat environment through strategic ruling, standardized logging, and collaborative analysis of relevant information
Targeted component replacement	• General explanation The system can intelligently adjust each subsystem and module in response to changes in external threats, reducing the risk of being attacked and improving the system's defense capability • Explanation of unique capabilities of the DHR architecture The DHR mechanism requires that differential-mode disturbance events in the current operating environment should be processed, and the current operating environment structure or executor status should be changed through iterative ruling and feedback scheduling to eliminate the impact of current network attacks
Resources upgradable	• General explanation Modules and components within the system can be upgraded and replaced • Explanation of unique capabilities of the DHR architecture The software/hardware resources in the DHR system can be reset, reconfigured, and reconstructed or the software/hardware versions can be upgraded through background operations without interrupting the current service or mission

each evaluation point, if the score of a certain point is lower than the bottom line, the system architecture is not cyber-resilient, and the overall cyber resilience score should be zero. If the scores of all evaluation points are higher than the bottom line, then based on the scores of each key capability, the score is calculated according to Eq. (6.2) to obtain the overall cyber resilience score of the system architecture, with the score ranging from 0 to 100.

$$S = 100 \times \left(\sum_i W_i S_i \right) / \left(\sum_i 50 W_i \right) \quad (6.2)$$

Wherein, S is the overall score of system architecture's cyber resilience; i is the number of evaluation points; W_i is the weight of evaluation point i ; S_i is the score of evaluation point i .

Table 6.6 Metrics for each key assessment item of threat situation perception

Key assessment item	General metrics	Specific metrics of DHR architecture
Threat data logging	• Whether to maintain system input and output logs • Whether to maintain system resource status logs • Whether to maintain system user operation logs • Whether to maintain system abnormal event logs • Percentage of modules in the system with logging capabilities • Percentage of logs with standardized specifications	• Whether to output DHR architecture standardized logs • Average time required to generate a snapshot of the differential-mode runtime environment • The percentage of current executors contained in the runtime environment snapshot
Threat backtracking analysis	• Percentage of abnormal events whose causes are understood through backtracking analysis • Percentage of attack events whose sources are located through backtracking analysis • Average time spent on threat backtracking analysis	• Through collaborative analysis of multi-source DHR architecture standardized logs and related information, it should be possible to 100% identify system reliability problems or cyber-attack problems.
Known threat awareness	• The number of threat intelligence sources used by the system • Frequency of updates to the system's threat intelligence • Number of threat types monitored by the system • Average time from attack input to perceived threat	• Whether the system's threat perception capability is independent of the quantity and quality of any threat intelligence source
System anomaly perception	• Number of types of abnormal events that can be sensed • Average time taken from occurrence to perception of an abnormal event	• Whether unknown attacks can be converted into differential-mode disturbances • Whether random faults can be converted into differential-mode perturbations
Unknown threat perception	• Average time from when an unknown attack occurs to when the system perceives it	• Average time from the occurrence of an unknown attack to the occurrence of strategic ruling anomaly • Whether it is possible to detect perceivable common-mode attacks or common-cause failures • Average time from the occurrence of imperceptible common-mode escape to the lifting of that state

Table 6.7 Metrics for each key assessment item of threat prevention and resistance

Key assessment item	General metrics	Specific metrics of the DHR architecture
Threat accumulation elimination	• Percentage of critical resources that can be dynamically rolled back • Average time to roll back from untrusted resources to trusted resources online • Percentage of critical resources that can be dynamically replaced. • Average time required to replace untrusted resources with protected resources • Average time of service disruption during resource rollback or replacement • Maximum or average service life of resources • Whether sensitive data remains after resources expire	• Average time taken from determining an abnormal executor to scheduling the abnormal executor offline • Average time required for abnormal executor cleansing • Average time required to generate a new executor from a heterogeneous resource pool • Average time required to schedule a new executor online • Average time required to synchronize the status of the executor newly scheduled online • Average time in which services are affected during the executor scheduling process
Attack surface shrinking	• Shrinkage ratio of system attack surface in terms of lines of code • Percentage of the critical system data encrypted	• The ratio of the number of lines of codes of the mimic bracket component to the number of lines of code of each executor • Number of structural codes
System camouflage	• Percentage of network resources capable of camouflage • Frequency of resource camouflage changes	• Degree of diversity and redundancy of available software and hardware resources • Frequency of executor scheduling or frequency of occurrence of differential-mode states within the operating environment
Known attack defense	• Percentage of types of known attacks that can be successfully defended against • Increase in system resource usage during attack defense • The degree of impact on system operations during the attack defense process	• Percentage of the types of known attacks discovered without relying on prior knowledge • No differential-mode disturbance can affect system service provision
Unknown attack defense	• Degree of impact on system resources when unknown attacks occur • The extent to which the system's service performance is affected in the event of an unknown attack • Whether the system can recover from unknown attacks • Average time from the occurrence of an unknown attack to system recovery	• 100% protection against any unknown differential-mode attack • Whether perceivable common-mode escape can be resisted • Average time from the occurrence of a common-mode attack to the elimination of the common-mode escape state • Provision of system services will not

(continued)

Table 6.7 (continued)

Key assessment item	General metrics	Specific metrics of the DHR architecture
		be compromised while resisting any differential-mode attacks
Generalized functional safety	• The extent to which system resources are affected when an attack or failure occurs • The extent to which the system’s service performance is affected in the event of an attack or failure • Whether the system can recover from an attack or failure • Average time from the time when an attack or failure occurs to the time when the system recovers	• The average time from the occurrence of a random-failure to the occurrence of strategic ruling anomaly • Average time from when an executor reaches the upper limit of ruling anomaly to when it is scheduled offline

6.3 Assessment Method

6.3.1 Overview

Cyber resilience assessment can be carried out at multiple levels such as the component, system, service, organization, region, and country. This book is concerned with system-level cyber resilience assessment, focusing on cyber resilience capabilities at the system architecture level. A variety of methods can be used to assess a system’s cyber resilience capabilities. According to the order of impact on actual systems and services from low to high, cyber resilience assessment methods can be divided into three categories: static assessment, adversarial assessment, and disruptive assessment.

The static assessment hardly interferes with system and service operation, and its implementation can be either qualitative analysis or quantitative assessment based on expert scoring. Adversarial assessment adopts the approach of red-blue confrontation, in which the attacker launches a real attack on the system, and the system’s cyber resilience capability is tested and evaluated under this condition. Adversarial assessment usually has a negative impact on system operation and service performance. The principle of disruptive assessment is to artificially inject faults or compromises into the system. The cyber resilience capability of a system is assessed by observing the operation of the system under the condition in which some of its components or security mechanisms fail. Disruptive assessment generally has a direct impact on system operation and service performance, and the degree of impact depends on the actual cyber resilience capabilities of the system. Existing disruptive assessment methods are mainly chaos engineering methods, which mean to inject random faults or failures into the system.

ESS white-box testing provides a new disruptive assessment method. It is not only capable of injecting random faults or failures into the system, but also capable

Table 6.8 Metrics for each key assessment item of threat damage recovery

Key assessment item	General metrics	Specific metrics of the DHR architecture
Modular system construction	• Percentage of components with standardized interfaces • Components that have strong coupling relationships with other components • Proportion	• Number of types of heterogeneous components in the resource pool • Average number of components contained in the executor
System component isolation	• Percentage of system components with restricted external communication • Percentage of system components with restricted internal communication • Whether restricted component isolation is implemented at the hardware level	• Number of mimic bracket components that communicate with the executors • Whether hardware isolation is implemented between executors • Whether hardware isolation is realized between executors and mimic bracket components
Abnormal component replacement	• Percentage of components that can be dynamically replaced • Average time for component replacement • The degree of impact on system service during component replacement	• Average time taken from determining an abnormal executor to scheduling the abnormal executor offline • Average time required to cleanse and recover the abnormal executor • Average time required to generate a new executor from a heterogeneous resource pool • Average time required to schedule a new executor online • Average time required to synchronize the status of the executor newly scheduled online • Degree of impact on service during executor scheduling process
Use of heterogeneous redundant components	• Percentage of components with multiple heterogeneous implementations • Number of heterogeneous operating systems used by the system • Number of heterogeneous hardware resources used by the system	• Number of types of heterogeneous software and hardware components in the resource pool • Number of various types of redundant components in the resource pool • Number of heterogeneous executors in the current running environment

of injecting security compromises into the system to simulate a scenario in which some components of the system are hacked, and assessing the resilience capability of the system under such conditions. Figure 6.5 summarizes the main elements, advantages, and disadvantages of the above three types of methods.

Analyzed from the two dimensions of black-box/white-box assessment and dynamic/static assessment, the positions of the above three types of assessment methods in the method domain are shown in Fig. 6.6. Both adversarial assessment

Table 6.9 Metrics for each key assessment item of threat change adaptation

Key assessment item	General metrics	Specific metrics of the DHR architecture
Targeted component replacement	• Percentage of components that can be effectively replaced in the event of environmental change • Average time to identify and replace risky or compromised components in the event of environmental changes	• Whether strategic ruling and feedback scheduling refer to the executor’s version maturity and historical trustworthiness • Whether strategic ruling takes into account the correlation between different executors, as well as security alerts and reminders • Whether strategic ruling takes into account the frequency of the executor’s abnormal ruling • Average time from when an executor reaches the upper limit of ruling anomaly to when it is scheduled offline
Resources upgradable	• Percentage of resources that can be automatically upgraded • Upgrade frequency of various types of automatically upgradeable resources • Average time required to complete resource upgrades • Degree of impact on service during the process of resource upgrade	• Percentage of components in the system that require real-time upgrades

Table 6.10 Scoring rules for assessment items

Score	Description
5	All assessed values for the corresponding quantitative metrics are at or above the target level
4	All assessed values for the corresponding quantitative metrics are within the acceptable range of the target level
3	Most of the assessed values for the corresponding quantitative metrics are within the acceptable range of the target level
2	Few of the assessed values for the corresponding quantitative metrics are within the acceptable range of the target level
1	All assessed values for the corresponding quantitative metrics are outside the acceptable range of the target level
0	This assessment item is not suitable for the target system architecture

and disruptive assessment belong to dynamic assessment, whose object is the to-be-assessed system running in the experimental environment or production environment. During the static assessment process, it is necessary to know the operating background, threat background, architectural characteristics, technical methods, design principles, and other information of the system. The above information is

Table 6.11 Evaluation points and key capability weights

Evaluation content	Evaluation points	Key capabilities	Scoring weights of evaluation points	Scoring weights of key capabilities
Threat situation perception capability	Threat data exploitation	Threat data logging	50	25
		Threat backtracking analysis		25
	Threat event perception	Known threat perception	50	10
		System anomaly perception		10
		Unknown threat perception		30
Threat prevention and resistance capability	Attack risk reduction	Threat accumulation elimination	50	20
		Attack surface shrinking		15
		System camouflage		15
	Threat damage defense	Known attack defense	50	10
		Unknown attack defense		20
		Generalized functional safety		20
Threat damage recovery capability	Resource damage limitation	Modular system construction	50	25
		System component isolation		25
	Damaged resource recovery	Abnormal component replacement	50	20
		Use of heterogeneous redundant components		30
Threat change adaptation capability	Intelligent system reconfiguration	Environmental change perception	50	20
		Targeted component replacement		20
		Resources upgradable		10

usually provided in the form of design specifications, log analysis reports, test reports, etc., and it is generally not necessary to require the system to be open to the evaluator in a white-box manner. Considering that the static assessment process requires a certain understanding of the system’s structural characteristics, technical methods, design principles, etc., and that in some demand scenarios, code auditing and other aspects may be involved, static assessment should include black-box and white-box approaches, as shown in Fig. 6.6. Adversarial assessments are generally

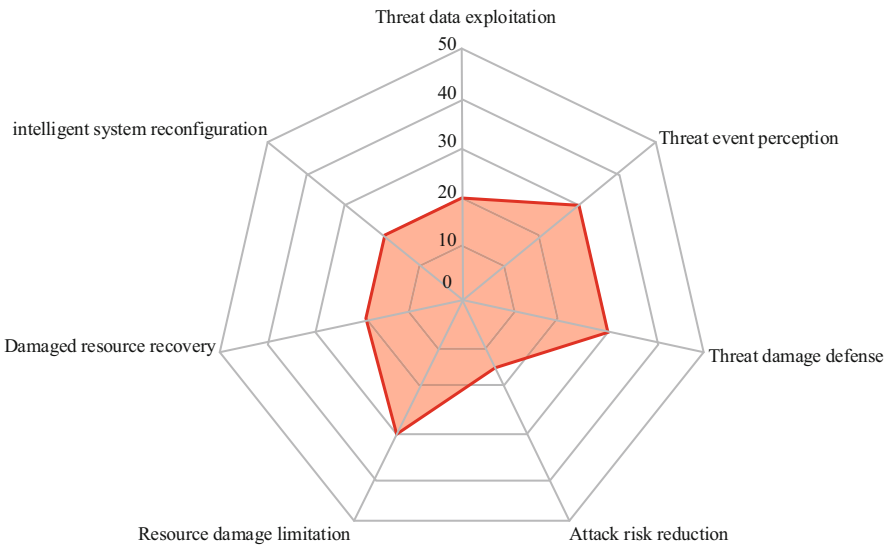


Fig. 6.4 System architecture cyber resilience baseline

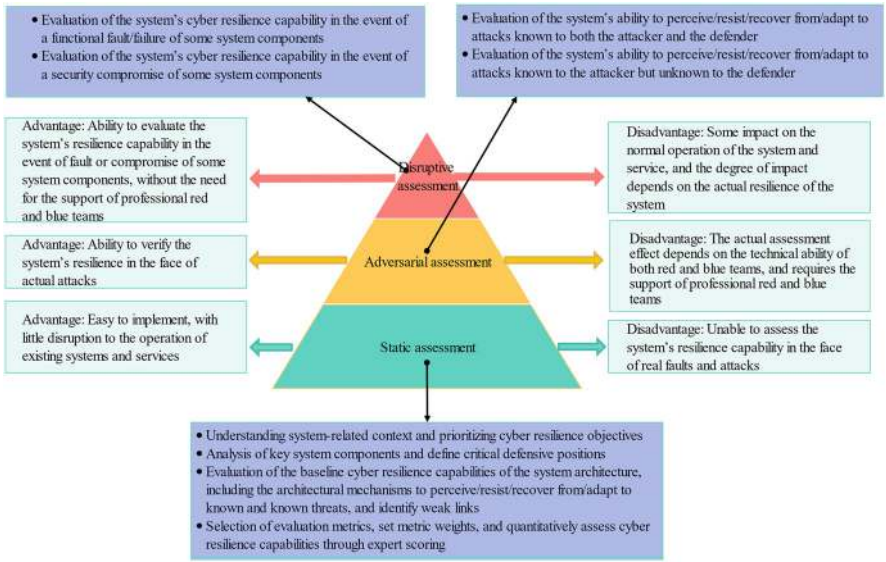


Fig. 6.5 Contents, advantages, and disadvantages of three types of cyber resilience assessment methods

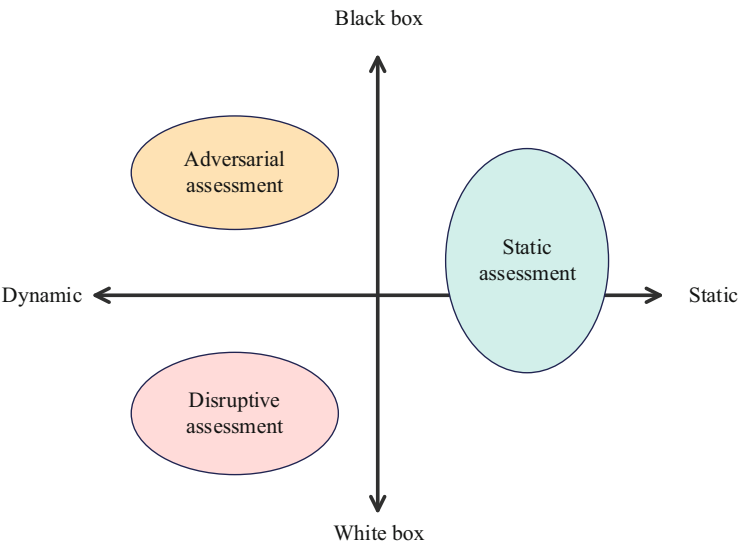


Fig. 6.6 The positions of various cyber resilience assessment methods in the method domain

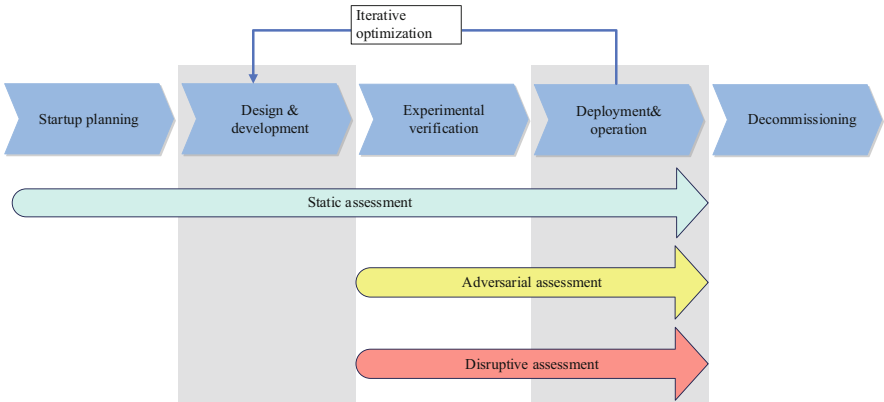


Fig. 6.7 Application of various assessment methods in different stages of the system life cycle

conducted in a black-box manner to simulate a scenario in which an attacker launches an attack on the system and verify the resilience of the system in the face of an attack. Disruptive assessments are generally conducted in a white-box manner, as they need to inject functional faults or security compromise into the system, and need to open part of the system components to the evaluator.

According to the Hall’s system engineering theory model [4], the system life cycle includes phases of startup planning, design and development, experimental verification, deployment and operation, and decommissioning. Different assessment methods are applicable to different stages of the system life cycle [4], as shown in

Fig. 6.7. In practical applications, one or several assessment methods can be selected based on a comprehensive measurement according to the life cycle stage of the system, the goals to be achieved by the assessment, the resources available, risk tolerance, and other factors.

6.3.2 Static Assessment

The system architecture's cyber resilience assessment through static assessment should follow the top-down principle. Firstly, a high-level background analysis is conducted to determine the operational background of the system and clarify the priority of the resilience objectives. Secondly, the technical architecture of the system is analyzed to determine the basic cyber resilience capabilities. Finally, based on the analysis results, combined with the experience of the experts, the relevant indexes are scored and suggestions for improvement are given. The third step is to score the relevant metrics and give improvement suggestions based on the analysis results and expert experience.

The static assessment is a rough assessment of the system's cyber resilience capability from a holistic perspective, the purpose of which is to outline the general profile of the system's cyber resilience capability. This process is also called cyber resilience analysis [1]. Cyber resilience analysis at the system architecture level mainly includes four steps: system background analysis, system technical architecture analysis, system threat and risk analysis, and provision of qualitative analysis conclusions and suggestions. The questions that should be focused on and answered during the static assessment process are shown in Table 6.12.

1. System Background Analysis

The process of system background analysis is shown in Fig. 6.8, which mainly includes four steps: analysis of the system's programmatic background; analysis of the system's architectural background; analysis of the system's operational background; and analysis of the system's threat background. The purpose of programmatic background analysis is to clarify how the system is acquired, developed, adjusted, and reconfigured; to sort out the participants of the system and understand their risk management strategies and risk points of concern, including prioritizing various types of objectives such as physical security, information security, reliability, system resilience, cyber resilience, and so on. The purpose of architectural background analysis is to clarify the type of the system, such as the CPS system, IoT system, enterprise IT facilities, etc.; to determine the external interface and external dependencies of the system; and to clarify the key components and technologies used in the system. The purpose of operational background analysis is to clarify the use of the system, including the basic functions of the system, user profile, key services, whether it contains high-value targets, effectiveness, and performance evaluation metrics, etc.; to determine the system's management support, upgrade, and maintenance; to clarify the

Table 6.12 Questions that should be focused on and answered during the static assessment process

Serial number	Question	Steps of question analysis
1	What are the key services carried by the system?	System background analysis
2	What high-value assets exist in the system?	
3	What are the sources of threats and characteristics of adversaries?	
4	How to set the weight of each cyber resilience objective and sub-objective?	
5	What cyber resilience design principles and technical methods have been applied to the system?	System technical architecture analysis
6	Can the cyber resilience design principles and technical methods adopted by the system satisfactorily support the realization of cyber resilience objectives and sub-objectives, especially those with larger weights?	
7	What is the assessment result of the system architecture capability metrics that can be statically analyzed?	
8	Are critical service subsystems capable of resisting “unknown unknown” cyber-attacks?	
9	What is the attack surface of the system?	System threat and risk analysis
10	What key resources and vulnerable links exist in the system?	
11	How complete is the system’s protection against the tactics, techniques, and procedures (TTPs) of known attacks?	

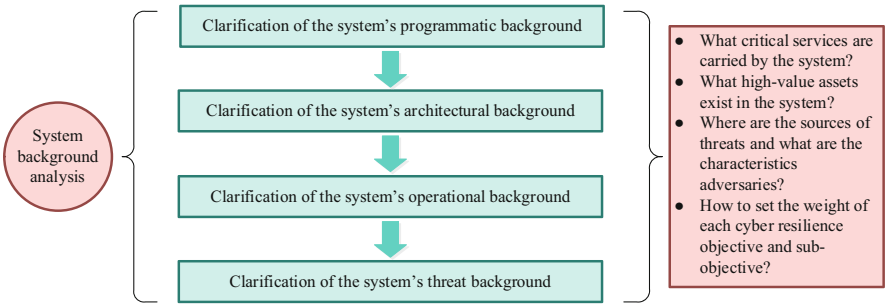


Fig. 6.8 System-related background analysis process

information interaction between the system and the external system and their dependency relationship; and to determine the expected changes in the system’s use and external dependency. The purpose of threat background analysis is to clarify the source of threats to the system, the threat events of concern, and threat scenarios and to analyze the adversary’s attack intentions, attack time, attack concentration, etc.

Through system background analysis, we can define the key services carried by the system, identify high-value assets existing in the system, sort out the sources of threats, understand the characteristics of adversaries, explain the cyber

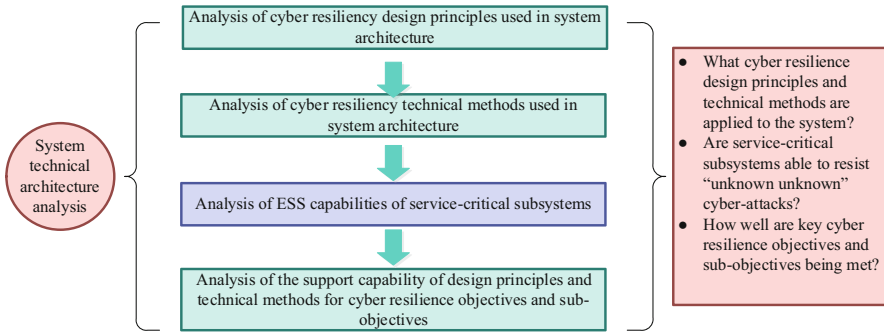


Fig. 6.9 Process of system resilience baseline analysis

resilience objectives and sub-objectives of the system in detail, and set the weight of each objective and sub-objective. Background analysis of the system is the premise and foundation of each subsequent analysis steps.

2. System Technical Architecture Analysis

The process of system technical architecture analysis is shown in Fig. 6.9. The first step is to analyze the cyber resilience design principles applied at the system architecture level, including strategic design principles and structural design principles. The second step is to analyze the cyber resilience techniques and methods applied at the system architecture level. The details about cyber resilience design principles and technical method can be found in Chap. 5. It is important to note the potential interactions between cyber resilience techniques. For example, the effectiveness of Technology A may depend on Technology B. Therefore, in the analysis process, various technical methods of cyber resilience should be explained in detail based on the specific application environment and background of the system, and their effectiveness in the real system should be evaluated based on the relationship between the technical methods, in order to obtain relatively objective analysis results. On the basis of analyzing and sorting out the design principles and technical methods adopted by the system, the ESS capabilities of service-critical subsystems are analyzed to assess whether the system has the ability to ensure the continuity and trustworthiness of key services under various adverse conditions. From the Incomplete Intersection Principle (IIP) proposed in Chap. 2, it can be seen that the necessary condition for service-critical subsystems to have ESS capabilities is that their system architecture must have complete dynamics (D), variety (V), and redundancy simultaneously. After completing the above three analysis tasks, the degrees of achievement of cyber resilience objectives, especially the key objectives with larger weights, are analyzed based on the relationship between the design principles and technical methods to support the cyber resilience objectives (see Appendix D.6 of Literature [1]).

By analyzing the technical architecture of the system, it is possible to clarify the cyber resilience design principles and technical methods adopted by the

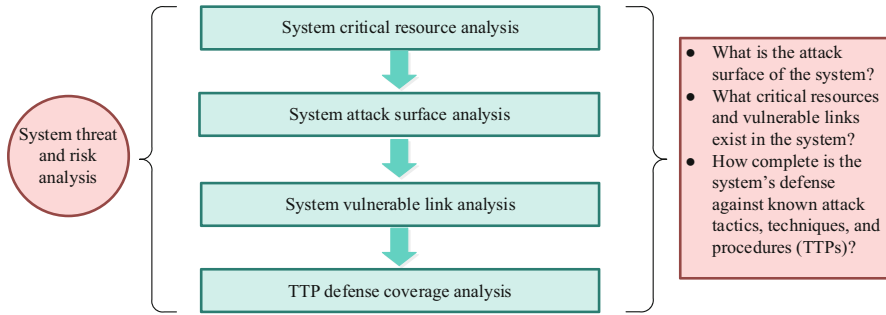


Fig. 6.10 System resilience capability analysis process in the operational background

system, analyze the degree of achievement of cyber resilience objectives, and determine whether the system has the “bottom-line capability” of cyber resilience to ensure the operation continuity and trust worthiness critical services.

3. System Threat and Risk Analysis

The process of system threat and risk analysis is shown in Fig. 6.10. First, from the perspective of key service functions, the key resources in the system are analyzed. The “key resources” here refer to the resources necessary to ensure the operation continuity and trustworthiness of critical services. Critical resource analysis can be performed through a variety of methods applied to contingency planning, resilience engineering, and mission assurance engineering, including criticality analysis, mission impact analysis, service impact analysis, Crown Jewels Analysis (CJA), and network mission impact analysis [1]. The purpose of system attack surface analysis is to determine the ways and links in which the system may be attacked. Attack surface analysis must have a relatively deep and complete understanding of the various backgrounds of the system. The purpose of analyzing the system’s vulnerable links is to identify potential vulnerabilities in the system, for example, system components that are at risk of single-point failure. The analysis of vulnerable links can be carried out using network analysis or graph analysis methods [1]. The purpose of TTP defense coverage analysis is to analyze the degree of coverage of the system’s defense capabilities against known types of attack tactics, techniques, and procedures (TTP) [5], and to assess the system’s ability to defend against APT attacks.

Through system threat and risk analysis, we can identify the attack surface of the system, sort out the vulnerable links of the system, discover potential risk points, and evaluate the system’s defense capabilities against APT attacks. It should be pointed out that the APT attacks represented by TTP are all known attacks, or in other words, they are “unknown known” network attacks. The system’s ability to perceive and defend against “unknown unknown” network attacks must rely on ESS capabilities.

4. Provision of Qualitative Analysis Conclusions and Suggestions

Through the above three-step analysis, the basic cyber resilience capabilities of the system can be determined and the main risks of the system can be

understood. We use the maturity level of five basic capabilities as an important indicator to qualitatively measure the system's cyber resilience: ① definition of service-critical subsystems and identification of critical defensive positions; ② the capability of service-critical subsystems to resist "unknown unknown" cyber-attacks; ③ the level of support of the system's cyber resilience design principles and technical methods for cyber resilience objectives/sub-objectives; ④ the level of elimination of the system's hidden attack surfaces and vulnerable links; ⑤ the system's capability to protect against known attacks. Each basic capability is mapped into three levels: "very mature," "mature," and "immature" according to its degree of perfection, as shown in Table 6.13. The maturity levels of the five basic capabilities allow a qualitative evaluation of the basic capabilities of the system's cyber resilience.

It should be emphasized that the five basic capabilities listed in Table 6.13 are not equally important for measuring system cyber resilience. For example, in order to achieve the ultimate goal of "mission assurance" of cyber resilience, service-critical subsystems must have the ability to withstand "unknown unknown" cyber-attacks, thereby ensuring the operation continuity and trustworthiness of critical services. If this capability is immature, even if the other four capabilities are mature, it is still difficult to conclude that the system's overall cyber resilience capability is mature.

Through static analysis, vulnerable links in the system's cyber resilience can be located and targeted suggestions can be given. For example, reducing the system's attack surface, eliminating vulnerable links, and enhancing the application of cyber resilience design principles and technical methods can better support the realization of high-weighted objectives and sub-objectives; revamping and updating the architecture of service-critical subsystems can enable them to resist "unknown unknowns" cyber-attacks.

6.3.3 *Adversarial Assessment*

Red-blue confrontation in the traditional cyber security field refers to carrying out actual red-blue confrontation drills based on real network environments to discover and improve cyber security defects at the technical and management levels. During the assessment process, one side plays the role of an attacker, aiming to obtain network asset access permissions, service data, and service control rights; the other side plays the role of a defender, aiming to discover and handle security incidents, analyze the results of attacks, and propose improvement measures. In China's military confrontation exercises, the Chinese side is represented by the Red Army and the enemy by the Blue Army. Following this custom, in China, the party that simulates hackers is usually called the blue team, and the party responsible for defending the network is called the red team. However, there are also cases where the attacking party is called the red team and the defending party is the blue team. To

Table 6.13 Basic cyber resilience capabilities focused on during the static assessment process of the system and their maturity definitions

Maturity	Capability				
	Definition of service-critical subsystems and identification of critical defensive positions	Capability of service-critical subsystems to resist “unknown unknown” cyber-attacks	Level of support of the system’s cyber resilience design principles and technical methods for cyber resilience objectives/sub-objectives	Levels of elimination of the system’s hidden attack surfaces and vulnerable links	System’s capability to protect against known attacks
Very mature	Able to clearly define service-critical subsystems and critical defensive positions, with service-critical subsystems and critical defensive positions meeting minimization conditions	Service-critical subsystems are fully capable of resisting “unknown unknown” cyber-attacks	Cyber resilience objectives/sub-objectives that the system focuses on are well supported, and high-weighted objectives/sub-objectives are better supported than other objectives/sub-objectives	The system’s attack surface is small, and there are no vulnerable links	Very high protection coverage against known TTPs, with complete protection of the system’s attack surface
Mature	Able to clearly define service-critical subsystems and critical defensive positions	Key components of service-critical subsystems have the ability to resist “unknown unknown” cyber-attacks	High-weighted cyber resilience objectives/sub-objectives that the system focuses on are well supported	The system’s attack surface is small but there are some vulnerable links	High protection coverage against known TTPs, with good protection of the system’s attack surface
Immature	Unable to clearly define service-critical subsystems and critical defensive positions	Service-critical subsystems do not have the ability to resist “unknown unknown” cyber-attacks	High-weighted cyber resilience objectives/sub-objectives that the system focuses on are not well supported and are difficult to achieve	The system has a large attack surface and obvious vulnerable links	Low protection coverage against known TTPs, with poor protection of the system’s attack surface

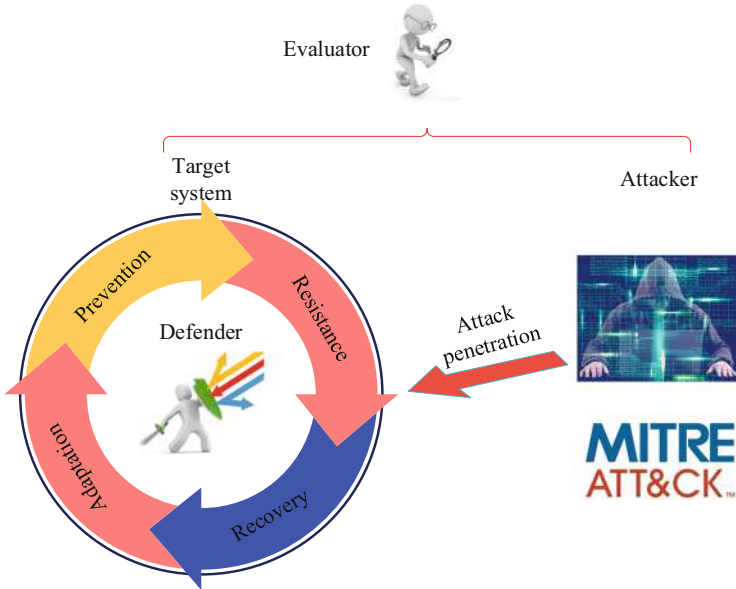


Fig. 6.11 The process of adversarial assessment

avoid ambiguity, this book uses “attacker” and “defender” to refer to the red side and blue side, respectively.

The adversarial assessment process is schematically shown in Fig. 6.11. During the assessment process, the system is a black box for the attacker. The attacker must try various potential attack paths to find a breakthrough. For example, the attacker can draw on the various TTPs listed in the open-source adversarial tactics, techniques, and common knowledge (ATT&CK) [5] to launch penetration attacks on the target system in an attempt to damage or even interrupt the operation of key services of the target system. The system is a white box for the defender. During confrontation, the defender takes targeted actions to block the attack, eliminate the impact, and ensure the operational continuity and trustworthiness of the system’s key services. In the adversarial assessment process, the evaluator is in a “God’s perspective,” and the details of the actions of both the attacker and the defender, as well as the operation of the system and services, are known to the evaluator. The evaluator evaluates the cyber resilience capabilities of the system by observing the operating status of the system and services during attacker-defender confrontation.

In the adversarial assessment process, the focus is on the system’s ability to perceive, resist, restrain, and adapt to known and unknown (but possibly known to the attacker) cyber-attacks, as well as the ability to safeguard the operational continuity and trustworthiness of critical services in the process. For network attacks that are known to both attackers and defenders, the system is mainly evaluated for the completeness of the defense coverage against various attack TTPs, as well as the extent to which critical service functions and performance are affected when an

Table 6.14 Basic cyber resilience capabilities of a system and their maturity definitions focused on during the adversarial assessment process

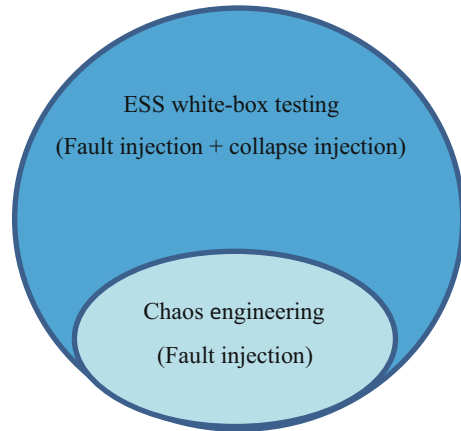
Maturity	Ability	
	System cyber resilience in the face of known attacks	System cyber resilience when facing attacks known to the attacker but unknown to the defender
Very mature	100% protection coverage against known TTPs, with no impact on critical service operations	The system has the ability to perceive and resist unknown attacks, and key service operations are not affected
Mature	Pretty good protection coverage against known TTPs, with a minor impact on critical service operations	The system has pretty good ability to restrain, recover from and adapt to unknown attacks, and critical service operations are not significantly affected
Immature	Low protection coverage against known TTPs, with obvious vulnerable links in the system and obvious impacts on critical service operations	Critical service operations of the system are significantly affected

attack occurs. By using network attacks known to the attacker but unknown to the defender, the system’s ability to perceive, resist, restrain, and adapt to unknown attacks can be assessed to a certain extent. It should be noted that in the face of professional defenders, implementation of network attacks known to the attacker but unknown to the defender essentially requires the attacker to master the high-risk 0-day vulnerabilities of the defender’s system, which is often difficult to achieve in real confrontation. For assessing the cyber resilience of the assessed system in the face of “unknown unknown” cyber-attacks, ESS white-box testing is an easy-to-implement, reproducible, and verifiable assessment methodology. A description of the ESS white-box testing methodology can be found in Sect. 6.3.4. The basic cyber resilience capabilities of the system and their maturity definitions, which are the focus of the adversarial assessment process, are shown in Table 6.14.

6.3.4 Disruptive Assessment

The basic method of disruptive assessment is to artificially introduce faults, failures, or security risks into the system and evaluate the robustness and security of the system under this premise. Chaos engineering methods currently used in robustness testing of large-scale distributed systems and ESS white-box testing methods are both disruptive assessment methods. Chaos testing is mainly to inject functional faults into the system to know the failure effect (Fault Injection), in order to verify the robustness of the system. Chaos testing focuses on injecting functional fault into the system to gain knowledge of the failures and verify the robustness of the system. ESS white-box testing not only includes the testing capabilities of current chaos engineering methods, but also injects security compromises into the system to simulate the scenario where some components of the system are compromised,

Fig. 6.12 Relationship between disruptive assessment methods



and evaluate the robustness and security of the system under this condition. From the perspective of methodological functionality, chaos engineering is a subset and a special case of ESS white-box testing, as shown in Fig. 6.12.

6.3.4.1 Chaos Engineering Assessment Methods

Chaos engineering is a complex technical means to improve the resilience of technical architectures. By deliberately creating faults, chaos engineering tests the behavior of the system under various faults and loads, identifies and repairs faults, and avoids serious consequences of the faults.

Chaos engineering implementation typically involves the following four steps:

The first step is to define and measure the “steady state” of the system. Metrics need to be precisely defined to show that the system is functioning the way it should. For example, in its chaos engineering practice, Netflix uses the rate at which customers click the play button on their video streaming device as a metric called “traffic per second.” In fact, service metrics are often more useful than technical metrics in chaos engineering because they are better suited to measuring user experience or operations.

The second step is to create a hypothesis. Since the attempt is to disrupt the steady state of the system during normal operation, the hypothesis should include the expected behavior of the system after the disruption is implemented.

The third step is fault implementation. Common methods of fault implementation include simulating data center faults, forcing system clocks to be out of sync, simulating I/O anomalies in driver codes, simulating delays between services, and randomly triggering functional anomalies. Implementing faults in complex systems can easily cause unexpected chain reactions, which is one of the results that chaos engineering looks for. Therefore, before implementing faults, it is important to first consider what could go wrong and to define the boundaries of the impact (also known as the “blast radius”) before proceeding with the simulation.

The fourth step is to prove or disprove the hypothesis. Steady-state metrics are compared with those collected after a disturbance is injected into the system. If a discrepancy is found in the measurements, then the chaos engineering experiment has been successful and the cause needs to be analyzed in depth and the system hardened so that similar events in the real world do not cause serious problems. If it is found that the system can maintain a stable state, then it indicates that the system's resilience is sufficient to cope with the implemented fault.

The basic principles for applying chaos engineering are as follows [6]:

Developing a hypothesis about steady-state behavior. Emphasis should be laid on the measurable outputs of the system rather than its properties. Measurements of these outputs over a short period of time constitute a proxy for the steady state of the system. The system's throughput, error rate, latency percentile, etc. may all be metrics of steady-state behavior. By focusing on the patterns of behavior in experiments, chaos engineering verifies whether the system is working properly, rather than trying to verify how it works.

Diversifying real-world events. Chaotic variables reflect real-world events, and we can prioritize these events by potential impact or estimated frequency. Events similar to hardware failures, such as server outages, software failures (such as incorrect responses), and non-fault events (such as traffic surges or scaling events) should be taken into account. Any event that can disrupt the steady state is a potential variable in a chaos experiment.

Conducting the experiment in a production environment. System behavior varies depending on environment and traffic patterns. Since changes in resource utilization can occur at any time, capturing actual traffic is the only reliable way to capture the request path. In order to ensure the authenticity of the system's execution method and the relevance of the currently deployed system, chaos engineering strongly recommends experimenting directly with production environment traffic.

Conducting automated experiments continuously. Manual experiments are labor-intensive and ultimately unsustainable. Therefore, we need to automate and continuously conduct experiments, and chaos engineering needs to build automated orchestration and analysis into the system.

Minimizing blast radius. Experimenting in production may result in undesirable customer complaints. While there must be a compensation for some short-term negative effects, it is the responsibility and obligation of the chaos engineer to ensure that these subsequent effects are minimized and taken into account.

Chaos engineering is a powerful practice that has transformed how software is designed and engineered on some of the largest service systems in the world. While other approaches address speed and resilience, chaos engineering specifically deals with system uncertainty in these distributed systems. However, existing chaos engineering methods and practices mainly introduce random failures, under which network robustness and resilience capabilities are observed. In contrast, ESS white-box testing not only introduces random failures, but also evaluates the security and resilience capabilities of the system under the condition in which some components of the system are successfully controlled by an attacker.

6.3.4.2 ESS White-Box Testing

The white-box testing method of ESS is a disruptive assessment method that is currently used to evaluate the functional safety and cyber security capabilities of ESS equipment and systems based on the DHR architecture. In ESS white-box testing, part of the reconfigurable execution environment in the ESS equipment and system is opened to the attacker. Under this premise, it is tested whether the attacker can carry out effective attacks on the entire equipment or system.

In the DHR architecture of ESS, unknown differential-mode vulnerabilities and backdoors or virus Trojans are allowed to exist in multiple heterogeneous reconfigurable execution environments. The security capabilities of the entire system are guaranteed by the DHR architecture of ESS. Figure 6.13 shows a schematic diagram of ESS white-box testing. The attacker has the administrator privilege of the reconfigurable execution environment 1 and can perform any operation on this reconfigurable execution environment, or shut down this reconfigurable execution environment (in this case, the white-box test degenerates into a chaos test, that is, a test of system resilience capability when some components fail). Under this premise, it is verified whether the attacker can rely on this reconfigurable execution environment to attack components such as input agent, strategic ruling, output agent (error correction output), or penetrate into other reconfigurable execution environments from the reconfigurable execution environment 1 to gain control of the entire device or system. The DHR architecture of ESS fully takes into account the security of the mimic bracket components, the security of internal communication between components, and the dynamic characteristics of the reconfigurable execution environment, making it impossible for an attacker in white-box mode to break through the constraints of the DHR architecture, gain a long-term foothold, and achieve a stable attack effect.

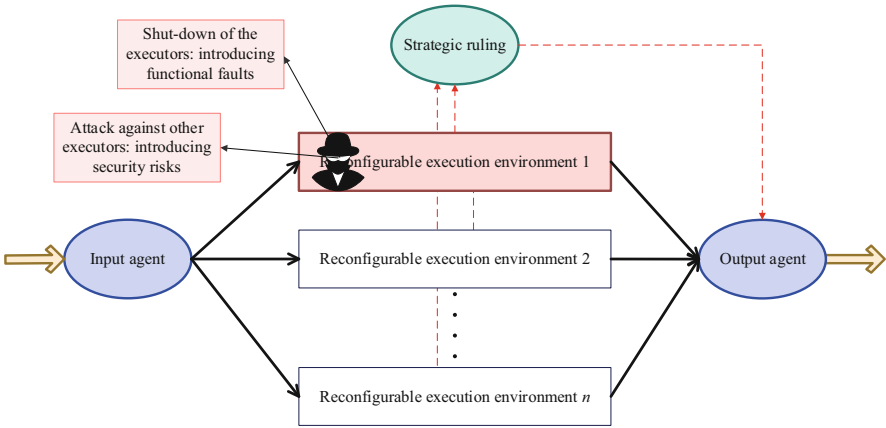


Fig. 6.13 Schematic diagram of ESS white-box testing

Table 6.15 Maturity of the ESS capabilities of a system

Maturity	Capabilities
Very mature	1. The time of common-mode escape is short, hardly affecting the security of network services 2. The ESS equipment and system can output standardized operation logs 3. The ESS equipment and system are capable of adaptive adjustment and evolution 4. The set critical defensive positions can fully ensure the continuous safe operation of key services; all the equipment and systems in critical defensive positions meet the requirements of Articles 1 to 3
Mature	1. In some cases, the time of common-mode escape is long, during which the security of network services may be affected 2. The ESS equipment and system can output standardized operation logs 3. The set critical defensive positions can fully support the continuous safe operation of key services; all the equipment and systems in critical defensive positions meet the requirements of Articles 1 to 2
Immature	1. The ESS equipment and systems have a higher probability of suffering common-mode attacks due to insufficient heterogeneity in the implementation of multiple executors 2. The time of common-mode escape is long, affecting the security of network services 3. There is no clear delineation of critical cyber defensive positions, and service-critical systems fail to implement adequate ESS protection

Depending on the number of reconfigurable execution environments open to the attacker, ESS white-box testing can be categorized into multiple modes such as the differential mode, *N-I* mode, *N* mode, etc. See Sect. 3.11 for a description of various modes of testing.

ESS white-box testing evaluates the system’s cyber resilience in the face of “unknown unknown” cyber-attacks. During the white-box testing process, problems that need to be focused on include:

1. Can the system continue to operate safely and steadily in the face of “unknown unknown” network attacks?
2. Are the functions and performance of key service functions of the system negatively affected?
3. Does the executor scheduling process of the ESS equipment and system affect service performance?
4. Can the ESS equipment and system output standardized logs to enhance the threat perception capability of the network?
5. How long does it take for the ESS equipment and system to sense and recover from common-mode escapes during common-mode testing?
6. Can the ESS equipment and system adjust the scheduling and ruling strategy according to the current attack mode and achieve adaptive evolution?

Table 6.15 gives a description of the maturity of the system’s ESS capabilities. The higher the maturity of ESS capabilities, the stronger the network’s ability to perceive and resist unknown threats, adaptively adjust and evolve, support continuous safe operation of critical services, and ensure cyber resilience.

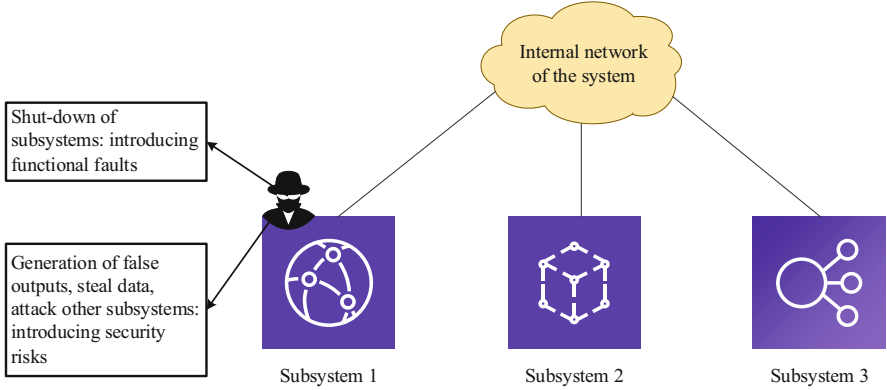


Fig. 6.14 Security testing diagram of an ESS white-box testing method applied to general systems

The ESS white-box testing methodology introduces the idea of disruptive testing into system security testing for the first time. When evaluating the resilience of a certain network system, the control of some of the components or nodes of the system can be completely handed over to the attacker, and the scenario in which the relevant components and nodes are compromised after being attacked can be simulated under this condition. Figure 6.14 shows a schematic diagram of applying the ESS white-box testing method to the security testing of a general system. The attacker has administrator privilege for Subsystem 1 and can perform any operation on this subsystem or shut down this subsystem (at this point, the ESS white-box test degenerates into a chaos test, which tests the system’s resilience capability when Subsystem 1 fails). After having the administrator privilege for subsystem 1, the attacker can try to attack and penetrate into other subsystems to gain control of the entire device or system. A system architecture with good resilience should be able to limit the scope of compromise and fully apply the system architecture design principles of dynamics and heterogeneity, so as to make it difficult for the attacker to gain a long-term foothold and achieve a stable attack effect. In essence, ESS white-box testing extends the traditional chaos engineering methodology to a more general situation. During the testing process, not only functional faults or failures, but also security collapses are artificially introduced. ESS white-box testing provides a verifiable GFS testing methodology for cyber resilience assessment. Especially for service-critical subsystems, it is necessary to conduct reproducible and verifiable assessment of their robustness and security through ESS white-box testing.

6.4 Relevant Regulations and Standards

On a global scale, the European Union has taken the lead in cyber resilience legislation. In September 2022, the Union published the draft of *Cyber Resilience Act* (CRA) [7], which introduced mandatory cybersecurity requirements in the whole life cycle of hardware and software products in the form of legislation. The Act sets four specific objectives, namely:

1. Ensuring that manufacturers can improve the security of digital products in their whole life cycle, including the design and development stages
2. Ensuring a consistent cybersecurity framework and promoting the compliance of hardware and software manufacturers
3. Improving the transparency of the security attributes of digital products
4. Enabling enterprises and consumers to use digital products safely

The scope of application of the Cyber Resilience Act includes all software and hardware products with direct/indirect/physical/logical network connections. The Act exempts some digital devices or products for which there have been regulations such as the European Union's Medical Device Regulation, or Regulation (EU) 2017/745, the In Vitro Diagnostic Medical Devices Regulation (IVDR), or Regulation (EU) 2017/746, the General Vehicle Safety Ordinance, or Regulation (EU) 2019/2144, and the Civil Aviation Ordinance, or Regulation (EU) 2018/1139. However, the Act does not exempt European digital identity wallets, electronic health record systems or products with high-risk AI systems (the Act has special provisions for high-risk AI systems, which are only applicable to high-risk AI systems defined in the draft of the EU Artificial Intelligence Act). The Act divides the products and covers into three categories, which, in descending order of security risk level, are Class II, Class I, and Unclassified (by default).

High-risk products belong to Class II, mainly including operating systems, virtual machine hypervisors, container runtime systems, public key infrastructures, digital certificate issuers, industrial firewalls, industrial intrusion detection/defense systems, general-purpose microprocessors, microprocessors for programmable logic controllers and security components, industrial routers, industrial modems, hardware security modules, etc. Class I products have a lower cybersecurity risk level than Category II products, but a higher risk level than unclassified (by default) products. Class I products include identity and access management software, browsers, password managers, malware detection software, products using virtual private networks, network management, configuration, monitoring and resource management tools, security information and event management systems, update and patch management tools, mobile devices and application management software, remote access software, and such products as firewalls, routers, modems, microprocessors, industrial automation and control systems, and industrial IoT that are not covered by Class II.

Unclassified (by default) products are digital products outside of Class I and Class II. Manufacturers of these products shall be responsible for ensuring and declaring

that their products meet all necessary safety/security requirements. They must provide a written declaration of conformity and affix a conformity mark to the product. Class I products must apply a unified standard or European cybersecurity certification scheme, or be subject to third-party conformity assessment. Class II products must be evaluated by a third party to demonstrate that they meet safety and security standards.

In July 2023, the Council of the European Union reached an agreement on the Cyber Resilience Act, which will take effect after it is formally approved by the European Parliament and the Council of the European Union. In addition to the Cyber Resilience Act, the European Union has also formulated a number of acts or directives related to cyber resilience, such as the Digital Operational Resilience Act (DORA) [8] for digital finance and the Critical Entities Resilience Directive (CER) [9] for key infrastructure. DORA was proposed in September 2020 and officially approved by the European Council and the European Parliament in November 2022. Financial entities and third-party ICT service providers must put it into implementation before January 17, 2025. DORA has formulated technical requirements for financial entities and ICT service providers in four areas: ICT risk management and governance, incident response and reporting, resilience testing, and third-party risk management. Among them, the “resilience testing” requirement provides that entities should regularly test their ICT systems to assess their protection strength and identify vulnerabilities. The results of these tests, as well as the plan for addressing the vulnerabilities identified, should be reported to the competent authorities for verification. Entities must conduct basic tests such as vulnerability assessment and scenario testing once a year. Financial entities that are considered to play a critical role in the financial system also need to conduct threat-led penetration testing once every 3 years. Key ICT providers of the entity must also participate in these penetration tests. CER was officially approved on January 16, 2023, and put into step-by-step implementation in 2024. The directive covers 11 key areas: energy, transportation, banking, financial market infrastructure, health, drinking water, wastewater, digital infrastructure, public management, space, and food. It requires all EU member States to determine key infrastructure, assess risks, and take measures to enhance cyber resilience (toughness), aiming at ensuring that key entities can prevent, resist, withstand, and recover from various destructive events.

The EU is still at the forefront of the world in formulating standards for cyber resilience. In 2015, the European Telecommunications Standards Institute (ETSI) formulated the group specification of “Network Functions Virtualisation (NFV) Resiliency,” [10] which set forth specific technical requirements for improving the resilience (toughness) of telecommunication networks based on network function virtualization technology. In 2021, the German Institute for Standardization formulated the standard “Stress-Testing Resilience of Critical Infrastructures Exposed To Cyber-Physical Threats,” [11] which defined the baseline of resilience and stress testing of industrial factories and premises, stipulated the testing principle and framework, and put forward the methods and indicators of such testing, as well as the procedure and process of using these methods and indicators. MITRE Corporation (a non-profit organization) of the United States released the Cyber Resiliency

Metrics Catalog [3] in 2018, which put forward more than 500 technical indicators for evaluating cyber resilience. Although the Metrics Catalog is rich in content, each indicator focuses on the evaluation of a specific technical ability, so the Catalog lacks an evaluation of the system architecture's overall ability, and there are problems in its operability in practical application and the consistency of evaluation results. China is also working on the formulation of the national standard "Network Resilience Evaluation Criteria," [12] which is expected to be published and implemented soon.

6.5 Summary of this Chapter

From the perspective of assessing the cyber resilience capability of the system architecture, this chapter provides cyber resilience assessment metrics at the system architecture level and explains the relevant cyber resilience assessment methods. The system architecture cyber resilience assessment framework focuses on the common problems of system cyber resilience and the assessment of core cyber resilience capabilities. The corresponding assessment results can be regarded as core metrics to measure the system cyber resilience capabilities and serve as core support for the "key vote" in practical applications. Cyber resilience assessment methods at the system architecture level include static assessment, adversarial assessment, and disruptive assessment. ESS white-box testing, as a new disruptive assessment method that provides capabilities not available in existing assessment methods, can verifiably measure and assess the GFS performance of a system. In practice, digital products passing ESS white-box testing can be labeled with a "gold label" of cybersecurity (similar to the security label of "U.S. Cyber Trust Mark" for IoT devices launched by the Federal Communications Commission in 2024) to help consumers know the security capabilities of digital products when purchasing them and promote the transfer of cybersecurity responsibility to the manufacturing side.

References

1. Ross R, Pillitteri V, Graubart R, et al. Developing cyber-resilient systems: a systems security engineering approach [R]. NIST special publication 800-160, Volume 2, Revision 1.2021.12.
2. Bodeau D, Graubart R, McQuaid R, et al. Cyber resiliency metrics, measures of effectiveness, and scoring: enabling systems engineers and program managers to select the most useful assessment methods [R]. MITRE technique report (MTR180314). 2018.
3. Bodeau D, Graubart R, McQuaid R, et al. Cyber resiliency metrics catalog [R]. MITRE technique report (MTR180450). 2018.
4. Hall A. A methodology for systems engineering [M]. New York: van Nostrand Publishing Company; 1966.
5. MITRE. ATT&CK [EB/OL]. 2022. <http://attack.mitre.org/>
6. Principles of chaos engineering [EB/OL]. 2019. <https://principlesofchaos.org/>
7. The European Cyber Resilience Act (CRA). <https://www.european-cyber-resilience-act.com/>

8. The Digital Operational Resilience Act (DORA). <https://www.digital-operational-resilience-act.com/>
9. The Critical Entities Resilience Directive (CER). <https://www.critical-entities-resilience-directive.com/>
10. ETSI GS NFV-REL 001 V1.1.1 (2015-01). Network functions virtualisation (NFV) resiliency requirements. https://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/001/01.01.01_60/gs_NFV-REL001v010101p.pdf
11. DIN SPEC 91461:2021. Stress-testing resilience of critical infrastructures exposed to cyber-physical threats. <https://webstore.ansi.org/standards/din/dinspec914612021>
12. National Technical Committee 260on Cybersecurity of Standardization Administration of China. Notice on inviting participating units in the formulation of information technology—network resilience assessment criteria. 2023 Feb 27. <https://www.tc260.org.cn/front/postDetail.html?id=20230227121156>

Chapter 7

Engineering Applications in Typical Fields of ESS Empowerment



7.1 ESS Engineering Construction Baseline

When constructing an ESS application, we need to establish an application baseline for evaluating or verifying the ESS characteristics of the empowerment structure and determining whether the ESS implementation of the target object needs improvement.

First, it is necessary to establish an initial baseline for the application of ESS structures—to determine the capabilities that the structurally empowered system should have. These capabilities can be functions and characteristics based on threat analysis (such as continuity of application operations, cybersecurity and network defensibility, maintainability and availability, performance management, etc.), or they can be derived from the ESS implementation goals, techniques, and methods in the system architecture design of the application (such as dynamics, heterogeneity, redundancy, and other features of the DHR architecture of ESS, input agents and error correction output mechanisms, strategic ruling mechanisms, scheduling mechanisms, feedback control mechanisms, sharding, randomization, diversification, virtualization, reconfigurable reorganization, etc.). In addition, these capabilities can also be the complexity of solving ESS problems in structural enablement, and the compatibility with other additive security technologies and methods. These capabilities may require decomposing the application into individual sub-applications and functional modules or analyzing and determining them at the architectural level.

Then, differences between the initial baseline and the actual application, as well as existing problems, need to be analyzed. Differences between application implementation capabilities and required capabilities, as well as individual problems with application implementation, may have been discovered and tracked during the design, implementation, and use phases of the application. This information can be obtained from application follow-up reports, penetration test reports, incident failure reports, and various assessment-related reports. Given that some of the differences

and problems may be common, the solutions used for constructing the application should be analyzed in a focused manner. Finally, assessment criteria need to be established, and an initial assessment of these capabilities, differences, and problems needs to be performed. The initial assessment can be an imprecise quantitative assessment. For example, the assessment results of various metrics can be roughly classified from high to low, or they can be simple qualitative judgments to determine whether or to what extent a certain feature of the ESS technology has been realized. The ESS characteristics of a constructed application can be assessed from multiple perspectives. For example:

From the perspective of basic functionality and performance, corresponding assessment metrics are formulated based on the capabilities required for application implementation of the empowered structure, such as the basic functions and protocols, the number of executors in the DHR architecture, the degree of heterogeneity and independence among them, the data processing and protocol types of input agents and error correction outputs, the arbitrator's ruling strategy and algorithm, the scheduler's scheduling strategy and algorithm, the execution efficiency and the response latency, etc.

From the perspective of security and risk management, the security and reliability assessment of the empowerment construction system is carried out based on existing attack models and risk assessment models. Examples include threat perception and response time, service function status in the face of threats or application pressure, etc.

From the perspective of ESS defense effects, the empowerment construction system is assessed through differential-mode test case injection experiments in ESS theory, temporal synergistic differential-mode test case experiments, $n-1$ perceivable common-mode test case injection experiments (n is the number of executors), n -mode (common-mode) test case injection experiments, and injection experiments of feedback control loops.

In addition, the initial assessment may involve assessing potential costs and program risks (such as potential cost risks, schedule impacts, and performance impacts) associated with investments in subsequent life cycle phases (such as operations and maintenance). The focus and priority of the empowered structure can be used to determine assessment methods (e.g., qualitative assessment scales, validity metrics, scoring systems, etc.) and related candidate metrics.

7.2 ESS-Empowered Network Communications

7.2.1 *Endogenously Safe and Secure Routing and Switching Equipment*

7.2.1.1 Threat Analysis

As the core equipment of the cyberspace information infrastructure, routers and switches are involved in the core layer, convergence layer, and access layer of the Internet. However, routing and switching equipment itself has some inherent problems, making it difficult to defend against malicious cyberattacks in a timely and effective manner [1], especially in the following four aspects:

1. The huge amount of codes in the design and implementation process results in numerous potential ESS problems.
2. It is generally difficult to deploy additional security protection and threat detection means such as firewalls and antivirus software for dedicated closed systems with high load-carrying requirements, and thus most routing and switching devices do not or cannot set up defenses against malicious attacks. Although routing and switching devices in some application scenarios have built-in conventional detection and defense modules, such as limiting the number of logins, regularly updating passwords, etc., their functions are relatively limited and rely heavily on prior knowledge (library), making it very difficult to defend against cyberattacks targeted at unknown vulnerabilities and backdoors in the devices.
3. Current routing and switching devices do not have cyber resilience. Due to the lack of an ESS mechanism featuring complete intersection of dynamics (D), variety (V), and redundancy (R), it is impossible to proactively suppress or eliminate network attack disturbances before, during, and after the event, resulting in the long-term existence of some “elder” vulnerabilities, backdoors, or even virus Trojans and leaving the system at the risk of being attacked throughout its life cycle.
4. Current routing and switching devices do not have trusted service capabilities. Once the routing forwarding table is hijacked or deliberately tampered with, the routing forwarding service will lose trustworthiness.

In view of the important position of routing and switching equipment in the network system, its unknown vulnerabilities and backdoors, once exploited by attackers, will produce functional safety, network security, and information security problems that are intertwined. This will not only compromise the functional safety (paralysis or downtime) of the routers and switches themselves but will also have a serious negative impact on the trustworthiness of their services. For example, if an attacker takes control of routers and switches and has the ability to tamper with routing forwarding tables at will, they can launch large-scale man-in-the-middle attacks and even steal or tamper with encrypted data, thereby jeopardizing the privacy, integrity, and availability of sensitive data. Therefore, cyber-resilient digital

facilities empowered by ESS are of great significance to enhancing the security capabilities of infrastructure “base” and ensuring the achievement of the objective of “co-management and co-governance” of cyberspace from the side of design and manufacturing of cyber-physical systems.

7.2.1.2 Design Thinking

Routing and switching equipment can be functionally divided into three planes: the data forwarding plane, the routing control plane, and the configuration management plane. The function of the data forwarding plane is to find corresponding routing table entries for data packets entering the system and forward the data packets out according to the table lookup results. The routing control plane implements route calculation by running various routing protocols (such as RIP, OSPF, BGP, etc.) and transmits the generated routing table entries to the data forwarding plane for use. The configuration management plane mainly provides users with various human-computer interfaces (such as CLI, SNMP or Web-based network management, etc.). A brief model of the logical functions of the routing and switching equipment or system is shown in Fig. 7.1.

Statistics from China National Vulnerability Database (CNVD) show that the vast majority of medium- and high-risk vulnerabilities come from management and routing protocols and vulnerabilities in the data plane has not yet been disclosed. This is related to the differences in functional processes of different planes of routing and switching equipment. The data plane is usually implemented using dedicated chips or hardware, making it difficult to attack. The control plane benefits from the standardization of routing protocols and the normalization of interaction messages as

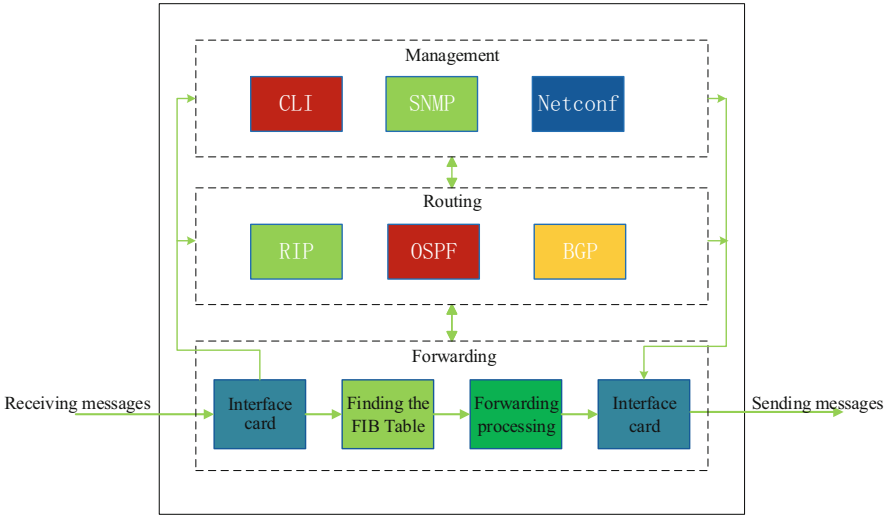


Fig. 7.1 A brief model of the logical functions of routing and switching equipment and systems

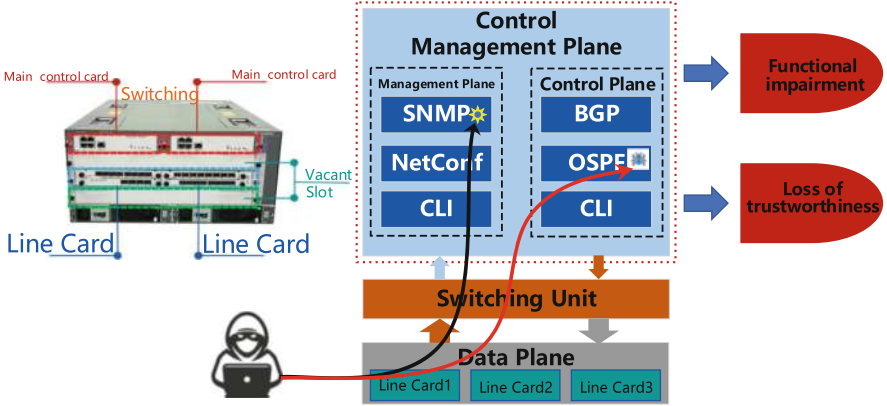


Fig. 7.2 Vulnerabilities of routing and switching equipment

well as the completeness of testing, with relatively few software defects. The management plane often has more software defects and is more vulnerable to attacks due to the specificity of the protocol and the convenience of remote management. As shown in Fig. 7.2, current routing and switching equipment lacks ESS genes, resulting in a lack of cyber resilience in the face of man-made attacks. Any exploitation of the associated vulnerabilities will result in impaired device functionality and loss of service trustworthiness.

In view of the vulnerability and architectural characteristics of routing and switching equipment, the routing control plane and configuration management plane of the equipment are regarded as critical defensive positions, and a mimic interface based on the DHR architecture is set. The mimic interface of the routing control plane is set in the operation of updating routing table entry information, and the mimic interface of the configuration management plane is set during the issuance of system management configuration commands.

According to the protection objectives determined by the mimic interface, the functional structure model of routing and switching equipment is combined with the DHR architecture of ESS, and the ESS architecture model of routing and switching equipment based on DHR is designed, as shown in Fig. 7.3. The model is based on the DHR architecture, which introduces multiple entities or virtual management executors in the configuration management plane and multiple entities or virtual routing calculation executors in the routing control plane, thereby constructing physical or logical heterogeneous redundant processing units in each plane. At the same time, an input agent is introduced for message distribution, and strategic ruling is introduced to realize error correction output. The results of strategic ruling are fed to the dynamic scheduling module through the feedback controller to realize dynamic reconfiguration, scheduling cleansing, and function recovery of each executor. Through the iterative changes in structural coding based on differential-mode sensing, it is possible to resist network attacks and solve the problems of functional deterioration and failure and loss of trusted services caused by unknown

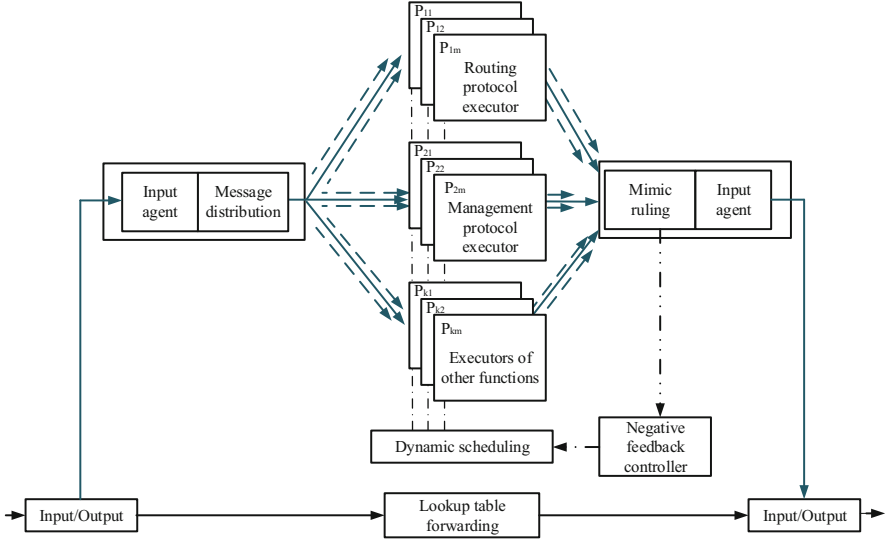


Fig. 7.3 An architectural model of endogenously safe and secure routing and switching

vulnerabilities and backdoors. Relying on feedback control, it is possible to achieve “one differential-mode sensing, one structural encryption.” Rapidly updating the structural codes (including cleansing, restoring, reconfiguring, resetting, changing the state of the operating environment, etc.) helps to achieve the cyber resilience objective of “mission assurance and service trustworthiness.”

7.2.1.3 Architecture

The architecture of endogenously safe and secure routing and switching equipment is shown in Fig. 7.4. Routing and switching equipment adopts the design concept of separating forwarding from control. While forwarding data according to the routing table, the forwarding plane uploads management protocol messages and routing protocol messages whose destination address is the local address to the protocol agent unit, which distributes the messages to each functionally equivalent heterogeneous executor. The independently generated routing update commands and configuration commands are sent to the ruling module, which performs differential-mode threat sensing and realizes error masking in the command error correction output. The scheduling and decision-making unit changes the apparent structure of the current operating environment based on the results of ruling and isolates and repairs damaged or corrupted executors, thereby ensuring a high level of security and defense while guaranteeing the normal realization of routing and switching functions.

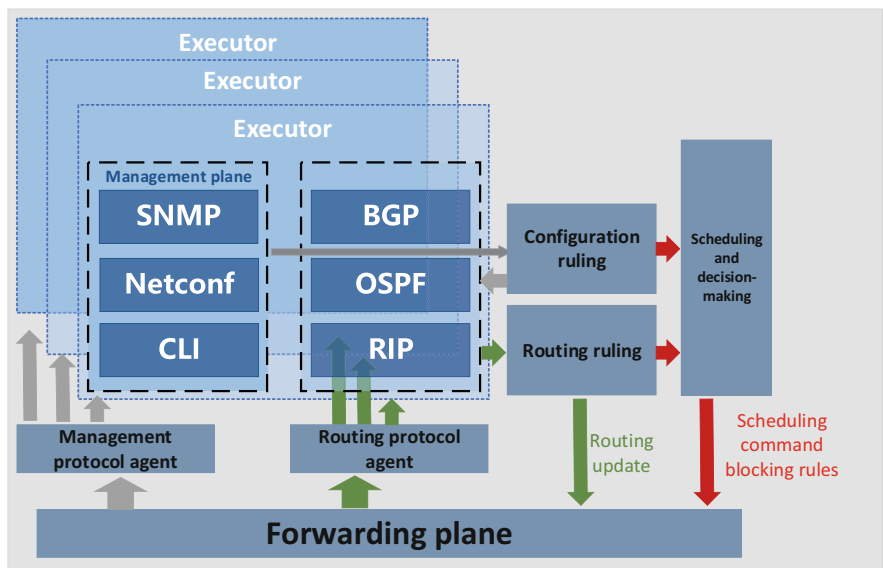


Fig. 7.4 Architecture of endogenously safe and secure routing and switching equipment



Fig. 7.5 Schematic diagram of heterogeneous executor pool

7.2.1.4 Construction of Heterogeneous Executors

The heterogeneous executor pool (shown in Fig. 7.5) contains multiple heterogeneous executors. These executors can provide equivalent functions such as routing

calculation and configuration command parsing, but they are different in hardware, operating system, compilation environment, protocol, and management, so as to minimize the likelihood that multiple heterogeneous executors have the same vulnerabilities and backdoors and the possibility of common-mode failures or common-mode escapes.

As shown in Fig. 7.5, heterogeneous executors can be constructed based on a variety of different CPU architectures (such as x86, ARM, MIPS, etc.) and operating systems (such as different distribution versions such as Linux, Windows, or operating systems developed by various manufacturers). The heterogeneous operating environment of executors can be realized at the level of instruction set, operating system, and middleware through different combinatorial models or through a combination of virtuality and reality. Physical executors can use physical hosts; or systems that have been virtualized can be run on a virtualized platform. Factors such as risk threats, costs, and efficiency should be comprehensively considered when determining the construction method of the heterogeneous executor operating environment.

The application protocol of executors uses different versions of open-source routing software (such as FRRouting, Quagga, VyOS, etc.), which are compiled diversely and deployed in various heterogeneous executor environments to achieve heterogeneity at the level of application software. At the same time, virtual router software launched by different manufacturers (such as H3C's VSR1000, Cisco's XRv, etc.) can also be deployed as executors to run in the executor pool. However, due to the different styles of management commands, closed source codes, etc., corresponding command line translation modules need to be developed for such heterogeneous executors, so that the configuration commands entered by the user can take effect in these executors.

7.2.1.5 Neighbor-Insensitive Protocol Agent Mechanism

The protocol agent unit is the entrance and exit for messages entering and exiting the system. This unit is responsible for distributing routing protocol messages to each executor and filtering the protocol messages sent outward by the executor for forwarding or error correction according to certain policies.

After the messages are input to the forwarding plane, the routing protocol messages destined for the local interface will be diverted to the routing protocol agent unit according to the configured diversion policy, the management protocol messages will be diverted to the management protocol agent unit, and the remaining messages will be forwarded according to the routing table maintained by the forwarding plane. The main objective of the protocol agent unit is to achieve parallel operation and single presentation of multiple routing executors based on the establishment of routing sessions with neighbors and route learning.

Two-way agents are used to complete the state synchronization of the finite-state machine (FSM) between multiple executors and neighbor sessions, the mechanism and principle of which are shown in Fig. 7.6. Internally, message replication is used

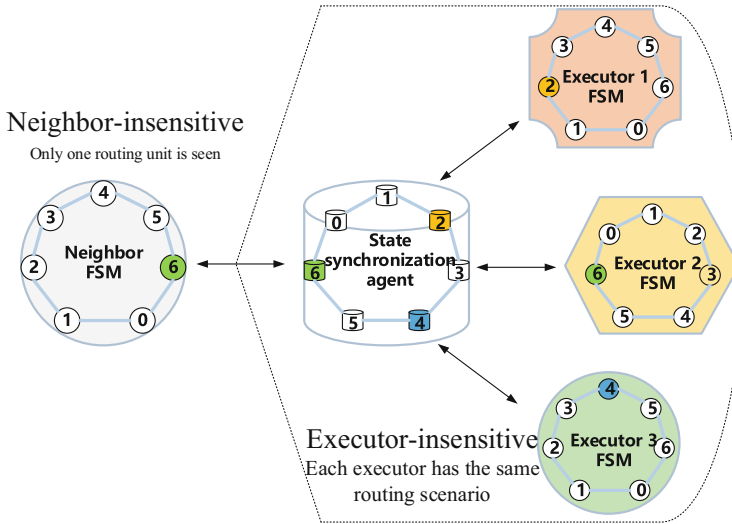


Fig. 7.6 Protocol state machine synchronization mechanism

to establish a session between the neighbor and each executor; externally, error correction output is used to achieve hidden presentation of multiple executors to the outside world. A caching mechanism is used to achieve route synchronization after executor switching and restart.

7.2.1.6 Differentiated Routing Ruling Mode Oriented to Application Requirements

The ruling unit and feedback scheduling unit are key components to maximize the DVR intersecting environment. The synergy between these two components makes it more difficult for attacks against the system to form a “coherent common-mode escape within the DHR environment under noncooperative conditions.”

The ruling unit is responsible for sensing routing update events or configuration issuance events of multiple executors and performing multimode ruling/error correction output. This unit performs relativity ruling on the routing table entry information delivered by each routing executor to the data forwarding plane according to certain principles. For executors that produce inconsistent results, the unit encapsulates the relevant information and summarizes it to the feedback control unit for the ruling unit to generate a ruling strategy.

The validity time of routing is an important performance indicator. The system’s ruling point should be set at the channel through which the control plane outputs routing table entries to the data plane. The way of routing ruling determines the validity time of routing table entries. According to the actual needs of different scenarios on the validity time of routing, two routing table ruling modes are designed

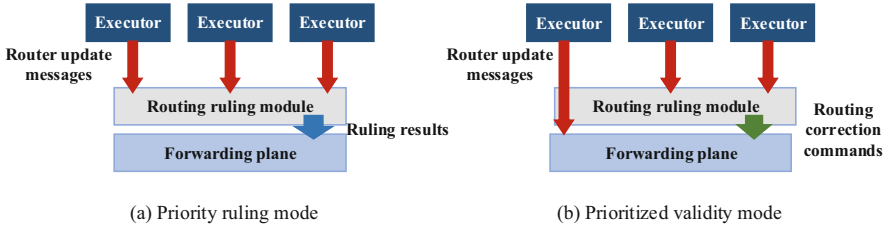


Fig. 7.7 Two modes of routing ruling

(as shown in Fig. 7.7). One is the priority ruling mode, in which routing ruling will be initiated first when routing is updated and routing table entries can take effect in the forwarding plane only after passing the arbitration. The other is the prioritized validity mode, which means that the routing update of a particular executor will take effect immediately in the forwarding plane before the ruling module initiates the ruling. If the ruling module adjudicates that a certain routing table entry that has just taken effect is abnormal, it will perform error correction for that table entry. The priority ruling mode is suitable for scenarios where there is a higher requirement for service trustworthiness, while the prioritized validity mode is suitable for scenarios where there is a higher requirement for route validity time.

For the priority ruling mode, the primary design goal is to shorten the route validity time as much as possible while ensuring no false alarms. Since there is a time gap when each executor notifies routing update messages to the routing ruling module, a waiting time window must be set to make a ruling over the received routing update information when the window expires. If the waiting time window is set unreasonably, it may cause problems such as false alarms or a ruling taking too long to take effect. To solve this problem, the routing ruling module adds a dynamic queue waiting timer to the waiting time window, so that the length of the waiting time window can be dynamically adjusted according to the distribution of the arrival time of routing update messages, which will reduce false alarms while shortening the validity time of the ruling. For the prioritized validity mode, the problem mainly lies in how to correct the misreading of the routing table as quickly as possible after the ruling module detects that the routing update is an abnormal behavior. The update behavior of the routing table is divided into adding, updating, and deleting routing table entries. For each of these three behaviors, different correction and recovery plans need to be formulated, respectively, so that the routing table can be quickly restored to the pre-attack state after being tampered with by the attacker.

7.2.1.7 Adaptive Recovery Mechanism Based on Feedback Control

The main function of the feedback control scheduling unit is to manage the heterogeneous executor pool and the operation of the executors in the pool. In accordance with the scheduling strategy specified by the ruling unit, it schedules multiple heterogeneous functional executors to realize the isolation, reconfiguration,

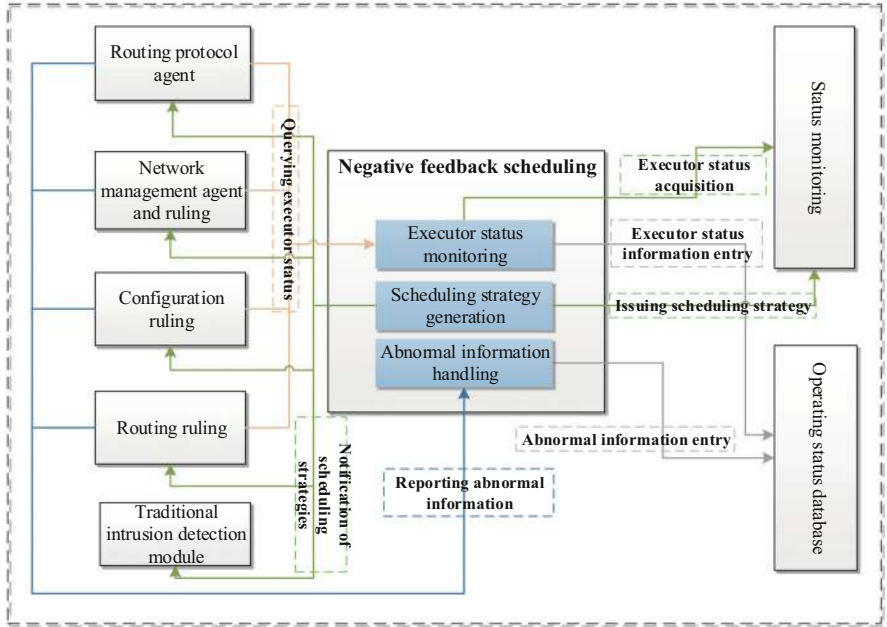


Fig. 7.8 Main functional components of the feedback scheduling unit

cleansing, and recovery of functional executors, increase the difficulty of scanning and discovery by attackers, minimize the visibility of unknown vulnerabilities and backdoors, isolate the abnormal functional executors, eliminate differential-mode perception expressions within the current operating environment, and ensure cyber resilience capability. Its main functional components are shown in Fig. 7.8.

For a differential-mode attack, the feedback control scheduling unit only needs to replace the executor with abnormal output in the operating environment with an executor selected from the background resource pool. However, if the substitute executor has exactly the same security flaws and weaknesses that can be exploited by the attacker as the one replaced, then there is still a theoretical possibility of common-mode escape [2]. Although the probability of this situation is very low, for the routing ruling mechanism that is solely based on majority-based ruling, it is difficult to detect attacks exploiting common-mode vulnerabilities and backdoors and locate problematic executors, because there may be “perceivable common-mode escapes” within the “blind zone of relativity ruling.” Therefore, it is necessary to combine routing rulings, the historical operation information (weight) of the executors, and even related alarm prompts with the cleansing scheduling/reconfiguration reset mechanism of the executors or the current operation scenario, so as to solve the problem of defending against cross-platform common-mode vulnerability and backdoor attacks and thus have the cyber resilience capability of “stifling” the consequences of the attacks that will continue to take effect in case of “common-mode escapes.”

In order to cope with the security threats caused by common-mode vulnerabilities and backdoors in the executor, the feedback scheduling unit designs an executor iterative cleansing mechanism to detect and remove the consequences of such network attacks. The executor iterative cleansing mechanism utilizes the executor scheduling mechanism and the strategic ruling mechanism. The core idea is to determine whether multiple heterogeneous executors may have been penetrated by attacks based on common-mode vulnerabilities and backdoors by checking whether the routing state of the executors changes after they have been cleansed and realize diagnosis and self-healing through multiple rounds of scheduling and strategic ruling. The iterative cleansing of an executor has two triggering conditions: ① the scheduling mechanism randomly cleanses a running executor from time to time, and if the state of the arbitrator changes after the cleansed executor comes back online, iterative cleansing will be triggered immediately. ② When the routing ruling mechanism detects that the route distribution behavior of an executor is inconsistent with that of most of the other executors, it will consider its routing state abnormal, and the executor will be taken offline for cleansing. After the executor is cleansed and brought back online, the iterative cleansing mechanism will initiate a query to the executor to check whether it still has the “abnormal routing state” identified in the last ruling. If not, it means that the cleansing has achieved the desired effect, and no subsequent operations will be performed. If yes, then iterative cleansing will be triggered.

The iterative cleansing process starts by randomly selecting an executor from the executors whose routing states were judged to be normal in the last ruling, forcing it to be cleansed, and checking whether the state of the arbitrator has changed after the executor has been cleansed. If a differential-mode state occurs, a second round of cleansing is initiated; otherwise, the iterative cleansing process is terminated, and the executor previously judged as abnormal is scheduled for replacement. The second round of cleansing will cleanse the remaining executors and bring them back online, and then the final ruling will be made. If the ruling result is inconsistent with that of the first round of ruling, it is assumed that an attack exploiting common-mode vulnerabilities has occurred to penetrate multiple executors, and the previous ruling result will be corrected for incorrect table entries according to the new ruling results. This mechanism can gradually and automatically restore the executors remotely controlled by the attacker to the normal state even if the attacker has penetrated most of the executors by exploiting common-mode vulnerabilities (which may only be true in theory). Common-mode escape cannot be a reproducible and stable event even if it occurs.

7.2.2 *Endogenously Safe and Secure Network Control System*

7.2.2.1 Threat Analysis

The development of cloud computing, data center, and network function virtualization technologies has gradually opened up previously closed networks. Innovative network architecture technologies such as software-defined networks (SDN) and multimodal network environments continue to emerge, further promoting in-depth changes in network technologies. With the development of white box technologies, the management and control functions that originally resided in network equipment such as routers and switches have been gradually stripped away and migrated upward to independent network control systems, making the latter increasingly important in the entire network architecture as the “new core” and “nerve center” of the network and a bridge connecting the underlying switching equipment and upper-layer applications. Downward, it controls the network element devices and realizes the abstraction and encapsulation of network-layer functions; upward, it provides network functions to meet the differentiated needs of upper-layer applications for the network.

However, with the globalized development of information technologies, the supply chain of various software and hardware systems in the network is easily “poisoned and polluted,” which highlights the problems of random failures, vulnerabilities, backdoors, and other uncertain disturbances in the network, directly threatens the normal operation of network control systems, and has a very bad impact on the robustness of their function, performance, and quality. Once a network control system fails due to random faults and human attacks, it will seriously threaten all levels of the network and have the deepest and widest impact on network functions.

The security problems of the network control system, the key protection object of the current network, are mainly reflected in the following five aspects: ① the new network’s decoupling data and control functions introduce a new attack surface in the network control system, such as the network control system’s southbound interface protocol and northbound application programming interface. ② One advantage of the new network is that the network control system software can be installed on an operating system based on commercial off-the-shelf (COTS) hardware (e.g., Windows or Linux operating system). However, this exposes the network control system to the same risks as the operating system, such as recurring buffer overflow attacks [3, 4]. ③ The centralized control architecture has become the Sword of Damocles for the network control system, and APT attacks can gain control of the entire network simply by infecting the network control system. ④ Numerical control separation enables frequent communication between switching equipment and network control system, which will greatly reduce network performance after a denial-of-service attack or a distributed denial-of-service attack [5]. ⑤ In addition to system vulnerabilities and backdoors, new attack surfaces, and disappearing hardware defense lines, control system software/hardware

misconfigurations and random faults/failures are also cyber resilience challenges faced by network control systems.

To sum up, it can be concluded that network control systems, as the key to support the application of new network technologies, are currently favored by the industry and academia and also present a large number of open-source/closed-source devices developed by different manufacturers in different languages. However, this flourishing development trend cannot conceal the security crisis with numerous vulnerabilities. Regarding the software/hardware security threats and damages faced by network control systems, whether from the perspective of philosophical principles or software/hardware engineering technologies, it is objectively impossible to design a software system without vulnerabilities/backdoors or a hardware device without errors/defects. If the security and robustness of the network control system cannot be guaranteed, it will be impossible to ensure the security/resilience of the network and the trustworthiness of network services.

7.2.2.2 Architecture

At the functional level, the network control system is used to run complex control protocols such as topology discovery, route calculation, and status monitoring, generate control table entries for use by the forwarding plane, and accept configuration management information from users and management applications. Network attacks based on unknown vulnerabilities and backdoors can be defended against by incorporating network control systems into the protection scope of the DHR architecture and constructing network controllers with an ESS mechanism.

Input messages from network elements or administrators are distributed in parallel to multiple online, heterogeneous controller executors for processing through corresponding control or management protocol agents according to subdivided message protocol types. The control and configuration information output by the latter after independent computation is fed back to the arbitrator (either centralized or decentralized); the arbitrator synthesizes the output results of each controller executor, the historical status of each executor, and the degree of heterogeneity between the executors to generate the ruling results, which are then sent down to the forwarding surface to take effect through the error correction output link. The arbitrator feeds the results of each ruling to the control scheduler as one of the bases for executing the scheduling strategy. If the arbitrator considers that the output result of an executor is suspicious, it will schedule it offline for cleansing, recovery, reorganization, or reconfiguration and at the same time schedule other executors in the ready state to work online.

According to the DHR architecture design of the network control system, ESS transformation can be carried out on all kinds of mainstream SDN controllers. The architecture of an endogenously safe and secure network control system is shown in Fig. 7.9. By constructing a dynamic, heterogeneous, and redundant controller executor, the southbound protocol agent and the management protocol agent are introduced to replicate and distribute protocol messages to achieve communications

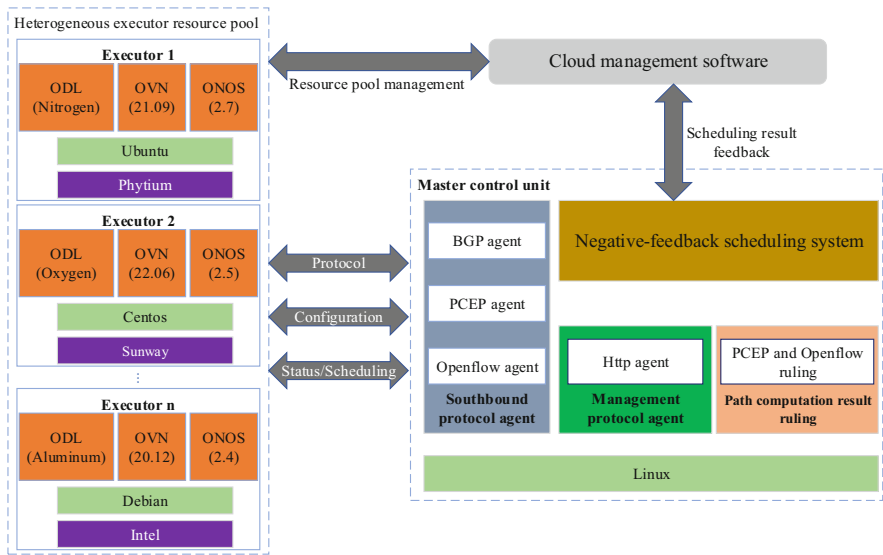


Fig. 7.9 Architecture of endogenously safe and secure controllers

between executors and communications with the outside world. Ruling points and error correction output links are set up on the path distribution channels of protocols such as PCEP (Path Computation Element Communication Protocol) and OpenFlow to make rulings on routing control commands of different heterogeneous controllers. The feedback scheduling system is responsible for receiving and processing the information reported by the path computation results and rulings, realizing unified management and processing scheduling of system’s abnormal information, and synchronizing the operating status of executors scheduled online.

7.2.2.3 Construction of Heterogeneous Executors

A variety or a combination of methods can be used to construct the controller’s heterogeneous executors. These methods include the following:

1. Diversified compilation realizes code obfuscation, equivalence transformation, randomization, and stack-related diversification technologies at three levels: the source code, the intermediate code, and the target code, ultimately achieving heterogeneity of compilation results.
2. Controller version heterogeneity. Different versions of the controller software system are used. In general, non-isomorphic versions of controller software do not have the same defects, and the probability of common-mode vulnerabilities and backdoors simultaneously existing in multiple non-isomorphic heterogeneous controller versions is very low. For example, OpenDaylight (ODL) controllers have several mature versions, such as Nitrogen, Oxygen, etc., and a new

version of Open Virtual Network (OVN) controller is released every year. Leveraging the different versions is an effective way to implement heterogeneous executors.

3. Heterogeneity of operating systems on which executors run. The controller software runs on an operating system. The vulnerabilities presented by executors using different operating systems (such as Ubuntu and Centos) are different. Different operating systems also have certain differences in the results of the controller software scheduling and execution.
4. Heterogeneous hardware platform on which executors run. The software-based control system should be distributed to run on multiple heterogeneous hardware platforms. The use of basic hardware with multi-command architectures such as Phytium, Sunway, Intel, and Hygon can not only prevent the loss of functionality caused by the simultaneous failure of isomorphic hardware platforms under common faults but also reduce the probability of common-mode vulnerabilities under the condition of human attacks.
5. Traditional security technology measures are differentially deployed in a heterogeneous set of executors. According to the mathematical expression of DHR, $P = f(S) = f(S_1 \cup S_2 \cdots \cup S_k) = f(S_1 \cup L_1) \cup (S_2 \cup L_2) \cdots \cup (S_k \cup L_k)$, where $S_1 \neq S_2 \cdots \neq S_k$, $L_1 \neq L_2 \cdots \neq L_k$, but L_i can be empty; and $P = P_i = f(-) = P_i = f(S_i) = (P_1 \cup P_2 \cdots \cup P_k)$, where $P_i \neq f(L_i) \neq f(L_1 \cup L_2 \cdots \cup L_k)$. We can differentially deploy a collection of security technologies L that have nothing to do with intrinsic functions in the current operating environment. For example, configuring security software such as honeypots, sandboxes, firewalls, and intrusion detectors on different executors makes it impossible for an attacker to create the same attack chain in different executors and exploit different attack resources. This is a more economical approach to heterogeneity.

7.2.2.4 Protocol Agent

The controller protocol agent is responsible for processing protocol messages, realizing communication between each controller executor and the outside world, and realizing the basic functions of sending/receiving protocol messages. It stores and replicates the messages received from the outside and distributes them to the executors and supports dynamic rotation of executors. After the executor comes back online, it can actively connect to the executor for each existing protocol connection. According to application classification, controller protocol agents can be divided into southbound protocol agents and northbound protocol agents. The former are responsible for interfacing with the network elements in the data plane, and the latter are responsible for interfacing with the application software of the controller.

From the perspective of protocol characteristics, controller protocol agents can be categorized into stateless and stateful protocol agents. Each protocol message request handled by the stateless protocol agent has nothing to do with any previous request message, so there is no need to save the protocol and session state. On the

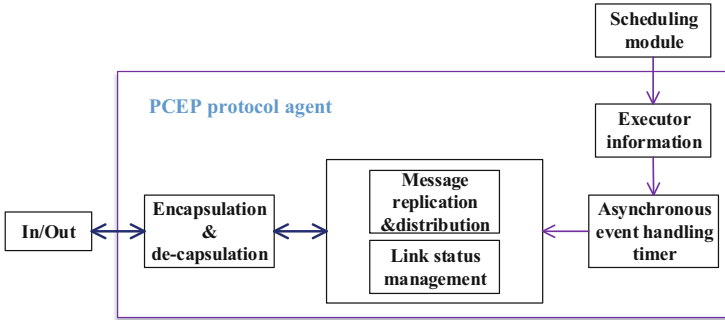
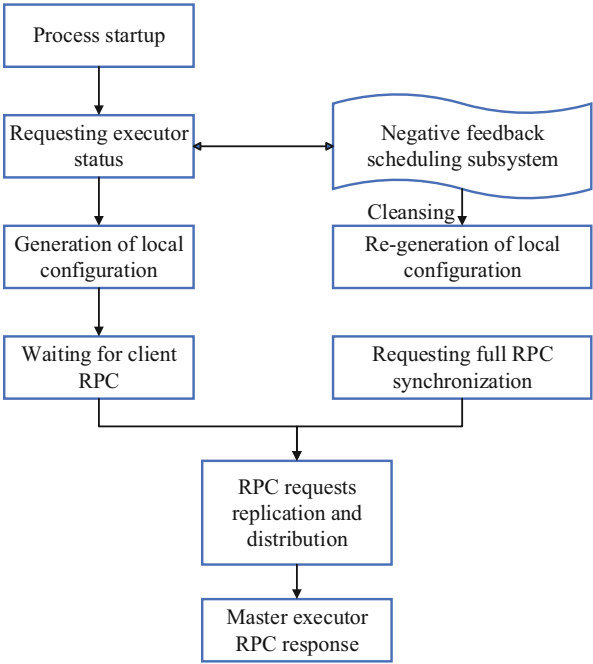


Fig. 7.10 Architecture of a PCEP protocol agent

contrary, the stateful protocol agent needs to maintain protocol and session-state information, and the corresponding protocol state synchronization needs to be performed when a new executor comes online. This section introduces the design and implementation ideas of protocol agents, with PCEP and OVSDb (Open vSwitch Database), two commonly used southbound protocols in controllers, as examples.

The path computation element protocol (PCEP) is the southbound protocol of the network controller. The controller calculates the path on the network for segmented routing so as to centralize computation paths. This requires a communication protocol between the network element and the controller. The controller as PCE (path computation element) is the server, the network element as PCC (path computation client) is the requesting client, and path computation is completed between PCE and PCC through PCEP protocol. The PCEP protocol agent is responsible for processing PCEP protocol messages and realizing the communication between each executor and its external neighbors. The agent architecture is shown in Fig. 7.10. First, the agent obtains information from the scheduling module about the status, interface mapping relationship, and MAC of the executor and dynamically receives the executor's online and offline information as well as information about changes in the executor's interface. It parses the messages received from the internal interconnection interface into normal messages and forwards them to the downstream process. When the service interface receives a message, it encapsulates the normal message into a specific message header and distributes it to each executor. The link state management module is responsible for the state of the TCP session between the executor and its neighbors and the cache management of the initialization messages. The message replication and distribution module is responsible for replicating messages and distributing them to each executor, as well as receiving messages sent by each executor. The asynchronous event processing timer is responsible for handling notification messages, periodic checks, and other matters, mainly including the retry operation when the monitoring interface fails, getting executors online and offline, and the removal of zombie connections.

Fig. 7.11 Operation process of an OVSDB protocol agent



The OVSDB management protocol is also a commonly used southbound interface protocol for controllers, which is used for programmable access and configuration management of virtual switches. This protocol defines a set of RPC (remote produce call) interfaces, which allow users to manage OVSDB through remote calls, mainly including the communication protocol JSON-RPC method and the supported OVSDB operations. The function of OVSDB management protocol agent is essentially to realize the agent of JSON-RPC that implements data transmission based on HTTP. The OVSDB management protocol implementation process is shown in Fig. 7.11. At startup, it requests the status information of the executor from the feedback scheduling subsystem, obtains the information of the executor that is currently online, and generates the local configuration information. Thereafter, it acts as an RPC server waiting for the client’s RPC connection. When the RPC client request is received at the service interface, the OVSDB management protocol simulates the client to send an RPC request to the executor, parses and extracts the message, encapsulates the message into a specific message header, sends it to each executor, and saves the state of the RPC session with the client as well as the status of the RPC session with each executor. The OVSDB management protocol agent only responds to the RPC reply message of a specific executor and ignores the response messages of other executors to RPC requests.

7.2.2.5 Strategic Ruling and Executor Scheduling

The strategic ruling of endogenously safe and secure network controllers adopts a trustworthiness-weighted protocol semantic-level discrimination method. The trustworthiness of the executor is determined based on a comprehensive evaluation of its operating environment, version, known defects, and its historical operating data. Since the outputs of different heterogeneous executors vary in terms of message formats, random labeling, etc., semantic-level discrimination is necessary. For different types of protocol messages, the endogenously safe and secure network controller makes comprehensive rulings and discovers the existence of abnormal executors through protocol parsing, semantic recovery, and semantic comparison, while taking into account of the trustworthiness of each executor.

For PCEP protocol ruling, the path ruling module regularly sends the computation path result obtained from the HTTP request message to each controller executor and parses the data to obtain the path control command. When the ruling timer expires, the computation path result data output from each executor is put into the ruling queue for comparison and ruling, and the computation path data is compared at the full semantic level. If the comparison result is normal, no processing is done; if the comparison result is abnormal, the information of the controller executor with abnormality is reported to the feedback scheduling subsystem.

For OVSDB protocol ruling, the endogenously safe and secure network controller introduces a southbound database arbitrator. After multiple heterogeneous executors convert and translate the same northbound input, the updated database of each executor is sent to the southbound database arbitrator for ruling. The southbound database arbitrator receives the synchronized southbound database files of each executor, parses the obtained southbound database files of each executor, and puts them into the queue waiting for ruling. The information waiting for ruling is sent to the voter for semantic-level voting to determine whether there are abnormal southbound database entries. After the ruling, if the information issued by a particular executor is found to be abnormal, the issued table entries will be corrected, and the normal information will be reissued through the said particular executor. At the same time, the abnormal information will be notified to the feedback control scheduler.

The feedback scheduling module of the endogenously safe and secure network controller implements heterogeneous executor scheduling based on the results of strategic rulings and externally input executor scheduling commands. The feedback scheduling module informs other modules of the system about the changes in the executor's online or offline status in a timely manner. Executor scheduling adopts various methods such as strategic ruling exception triggering, executor running status exception triggering, and random triggering. Execution scheduling triggered by strategic ruling exceptions can avoid (usually the most likely) differential-mode attacks. Scheduling offline and cleansing the executors with ruling exceptions can block attacks that have already occurred and realize the objective of "restrain" of cyber resilience. Executor scheduling triggered by abnormal executor running status can eliminate executor downtime caused by random faults during operation and

improve the robustness of the endogenously safe and secure network controller. Finally, random scheduling can effectively block common-mode attacks. Through random scheduling, common-mode attacks are converted into differential-mode attacks, thereby avoiding the risk of attackers exploiting common-mode vulnerabilities to achieve long-term malicious control in extreme cases.

7.2.2.6 Comparison with the Present Cluster Deployment Method

Existing controllers are mainly deployed in clusters to improve their reliability. The cluster runs multiple controller instances at the same time and each instance can be in three states: master, slave, and follower, where the master controller communicates with north-south devices outside the cluster. Each controller instance uses a consensus protocol (such as Raft) to achieve state synchronization. When the master controller fails due to a random fault, the slave controller will automatically take over its work to avoid the overall failure of the network caused by the failure of the master controller.

Although the existing cluster deployment approach can cope with controller failures caused by random faults and improve system reliability, it is essentially an isomorphic redundant approach. Isomorphic controller instances inevitably have common-mode vulnerabilities, backdoors, and even virus Trojans, so the existing cluster deployment approach cannot solve common ESS problems, and the security of controllers is still at great risk. Considering the importance of the controller to the network, if its security cannot be guaranteed, cyber resilience will be impossible. In contrast, endogenously safe and secure controllers employ multiple heterogeneous controller executors and combine them with strategic ruling (either centralized algorithms or distributed algorithms, such as Byzantine algorithms, etc.), dynamic scheduling, and feedback control mechanisms to obtain endogenously safe and secure cyber resilience endowed by the DHR architecture. From the perspective of implementation cost, current controller clusters need to run multiple controller instances in parallel, and the required computational resources are equivalent to the endogenously safe and secure network controllers. In other words, endogenously safe and secure controllers achieve the three goals of endogenously safe and secure network resilience: high reliability, high trustworthiness, and high availability, goals that are impossible to achieve with the present clustering approaches at basically the same cost.

7.3 ESS-Empowered Cloud Computing

7.3.1 *Cloud Computing and Cloud Native*

With the rapid development of information and communication technologies (ICT), individuals and institutions have an increasing demand for basic ICT resources such

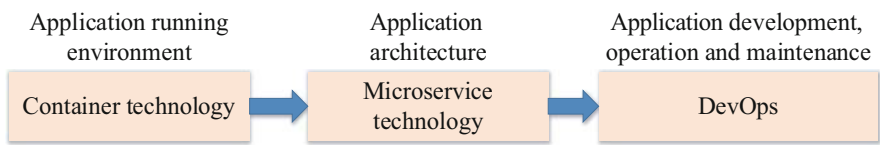


Fig. 7.12 Cloud-native key technologies

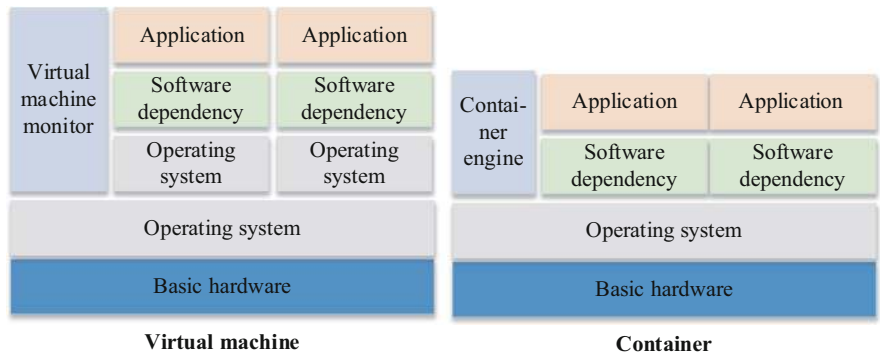


Fig. 7.13 Comparison of container and virtual machine isolation principles

as computing and storage. In response to the resource utilization dilemma under the traditional local computing model, a new resource supply model—cloud computing—has emerged. With the advantages of on-demand usage, ubiquitous access, resource pooling, and elastic scaling, cloud computing has subverted the traditional application development, deployment, and operation and maintenance models. Especially with the evolution of cloud nativity, application development is gradually shifting to be designed natively for the cloud. With the evolution of cloud-native transformation, application development has gradually transformed into native design for the cloud.

In the cloud-native stage, a unified cloud-native platform is used to carry out resource management, service development, service operation, and service maintenance, shielding the impact of heterogeneous software and hardware on application development, so that cloud-native applications can seamlessly migrate between heterogeneous hardware environments. The key technologies mainly include the container technology, the microservice technology, and the DevOps technology [6], as shown in Fig. 7.12.

Among them, the container technology provides a lightweight operating environment for cloud-native applications by sharing the host operating system [7], the principle of which is shown in Fig. 7.13. The implementation of the container technology mainly relies on the cgroup mechanism and the namespace mechanism in the Linux kernel. Cgroup is a resource management framework in the Linux kernel, which can set resource limits for processes or groups of processes in the system, such as central processing unit (CPU) capacity, memory, and input/output

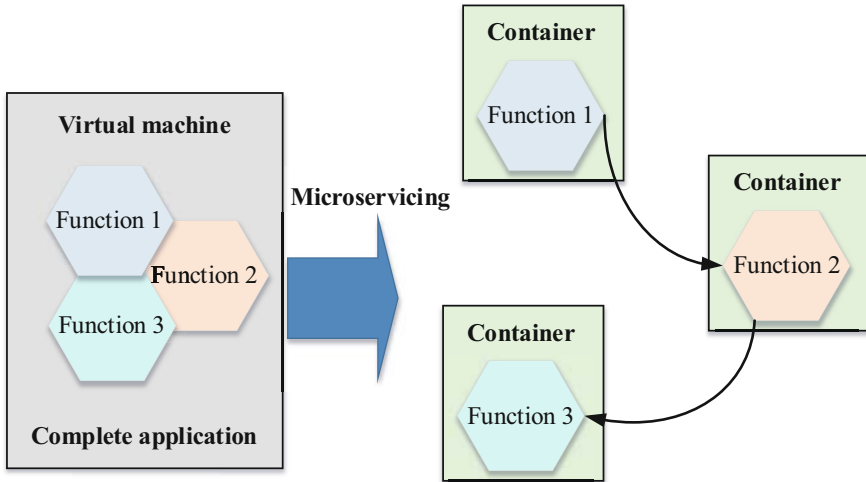


Fig. 7.14 Complete application microservice splitting

(IO) capacity and other resources. Based on the cgroup mechanism, resource limits can be set for different containers to ensure that there is no resource interference between containers on the same host. The namespace mechanism is an isolation mechanism provided by the Linux kernel, the program can only perceive programs in the same namespace as its own but cannot perceive programs outside the namespace. The Linux kernel mainly includes the following six namespace isolation functions: the host name, the message queue, the process number, the network device and protocol stack, the mounting point, and the user group. Based on the above six isolation mechanisms, the processes in the container will be as if they were in a separate system environment to achieve independence and isolation.

The microservice technology evolved from the service-oriented architecture (SOA) technology and is a new type of application architecture [8, 9]. The comparison between microservice architecture and the traditional application architecture is shown in Fig. 7.14. Traditional applications often adopt a monolithic architecture, where all functional modules in the application software are tightly coupled together and run in the same operating environment, such as a virtual machine. However, as the functions of application software continue to increase, the amount of codes in the application software becomes very large. A huge monolithic application makes its update and iteration very complicated. For example, due to the strong coupling between functions, a simple modification of one function may have a cascading impact on other functions. In response to the problems caused by the large scale of monolithic applications, the SOA technology proposes to split the different functional units of an application, but due to the lack of lightweight operating environments, it has not been widely used. When the container technology emerged, containers provided an ideal lightweight operating environment for split services, and the microservice architecture emerged. In the microservice architecture, multiple

loosely coupled microservices are orchestrated and combined to provide users with specific functions. Microservices are independent of each other and can be called by other microservices through application programming interface (API). Based on the microservice architecture, the development of different microservices can be carried out independently, reducing the difficulty of communication between teams. At the same time, it also greatly reduces the complexity of updating and iterating large-scale application software.

Based on the microservice application development model, the cloud-native platform provides a series of components for application development, such as application development frameworks, development-operation integration (DevOps) components, observability components, etc. The powerful cloud-native platform not only gives full play to the resilience and resource pooling advantages of cloud computing but also shifts the focus of concern from resource-centered to application-centered, which greatly facilitates the innovation of cloud applications.

7.3.2 Threat Analysis

Under the trend of cloud computing and cloud-native transformation, features such as containerization and microservicing have subverted the development, deployment, and operation and maintenance models of traditional applications. This disruptive architecture also brings new security challenges, which are mainly reflected in two aspects: containerized deployment and microservice splitting.

7.3.2.1 Containerized Deployment

Weak isolation of the operating environment is the primary security threat to containerized deployment. Lightweight isolation technologies such as containers provide an isolated operating environment for applications by sharing the host operating system. However, the approach of sharing the OS kernel also exposes containers to the threat of escape. Firstly, some dangerous configurations or mounting of the container engine may lead to container escape. For example, the process in the container is given excessive privileges, and some sensitive files of the host are mounted to the container. Secondly, the container engine invokes the isolation mechanism of the operating system to implement container isolation, such as the control group (cgroup) and namespace mechanisms in Linux [10]. If there are vulnerabilities and backdoors in the container engine, the attacker will be able to exploit them to achieve container escape. For example, the CVE-2019-5736 vulnerability can be exploited to attack the Runc module in the Linux container engine to achieve container escape. Finally, containers on the same host are essentially running on the same operating system. If there are vulnerabilities in the security mechanisms of the operating system (such as capability and seccomp mechanisms in Linux), attackers in the container can directly attack the operating system and

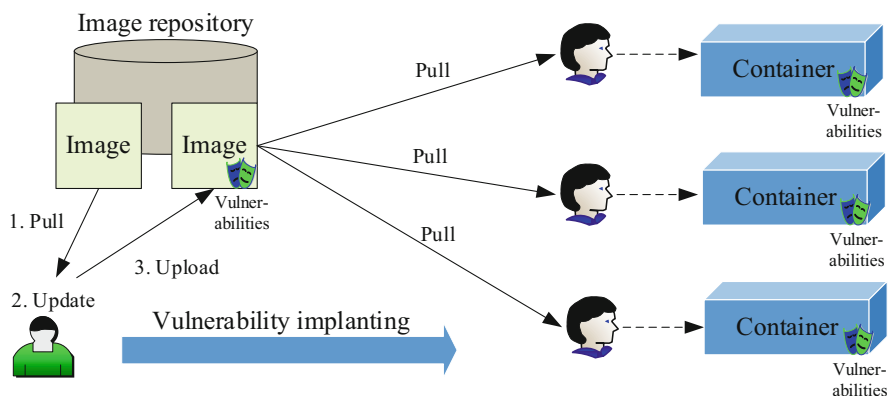


Fig. 7.15 An example of implanting vulnerabilities into container images

achieve container escape. For example, by bypassing the kernel address space layout randomization (KASLR) mechanism of the Linux kernel, an attacker can overwrite the address of a Linux kernel function pointer, launch a memory leak attack, and complete container escape. In order to avoid the threat of container escape caused by a shared operating system, secure containers such as the Kata container and Firecracker achieve virtual machine-level isolation effects through hardware virtualization [11]. However, these types of secure containers still face the threat of vulnerabilities and container escapes. For example, Yuval Avrahami shared the research results of escaping from the Kata container to the host machine through vulnerabilities at the 2020 Black Hat North America Conference [12].

The layered architecture and distribution mechanism of container images introduces additional security threats while facilitating the building of applications. Figure 7.15 shows an example of an attacker planting a vulnerability in an image. Users can build a new container image by using the existing image as the base image and superimposing the image layers they need on top of it. However, the layered architecture of container images also introduces new security problems, fundamentally because of the uncontrollability of the basic image on which the new image is built. When container images are packaged, the required images are often pulled from the public image repository as the base image. However, the security of the base image is not self-evident. On the one hand, there may be vulnerabilities and backdoors in the base image that are not repaired or blocked in time, and there may also be security risks caused by misconfiguration. On the other hand, attackers may also implant a backdoor into the image and upload it to the image repository for everyone to call as the basic image, leading to a serious “trapdoor flood.”

7.3.2.2 Microservice Splitting

The current characteristics of microservicing in cloud computing have given rise to application scenarios such as microservice and serverless, which also bring new security threats and damages.

Microservices are a typical application model in cloud-native scenarios. Different from traditional monolithic applications, microservice applications work together through multiple microservices to accomplish various functions of the application. Due to the fine-grained splitting of services and the complex interdependencies between services, the interactions between microservices show an explosive growth trend, which also exposes a large number of attack surfaces to attackers. The exponential growth of the attack surface makes the security control of microservices very difficult. At the same time, the complexity of interactions between microservices also brings a huge challenge to the privilege control of cloud networks. If the container network cannot implement fine-grained privilege control based on the interaction requirements between microservices, more security threats may be introduced. For example, microservices that do not need to be accessed by external networks are set with external network access privileges. Finally, microservice frameworks such as Spring Cloud and Dubbo provide common requirement modules for the development of microservices, which may also introduce common-mode vulnerabilities or backdoors. For example, log4j, a widely used logging framework in Spring Cloud, has a remote code execution vulnerability (CVE-2021-44,228), which has a huge impact on major industries around the world, and the harm is comparable to the “EternalBlue” vulnerability in 2017 [13].

“Serverless” is a cloud-native scenario where microservicing is developed to the extreme. Developers only need to write functions related to the application logic, without having to consider the deployment, scalability, and availability of the application [14]. In the “serverless” scenario, the function codes are pulled into the operating environment for compilation and execution, and multiple functions are orchestrated into a function workflow pattern to achieve specific functions. Similarly, the “serverless” service model greatly increases the attack surface, making it vulnerable to serious security threats in both the code compilation and running phases. At the same time, the management of the operating environment in “serverless” is more complex, and the reuse operation of the operating environment may facilitate the lateral movement of attackers. Finally, the “Serverless” Framework may also introduce location security vulnerabilities.

7.3.3 Design Idea

The current mainstream cybersecurity defense paradigm is difficult to completely solve the security threats faced by cloud computing and its native technology in the development, especially the common ESS problems. The CESS defense paradigm

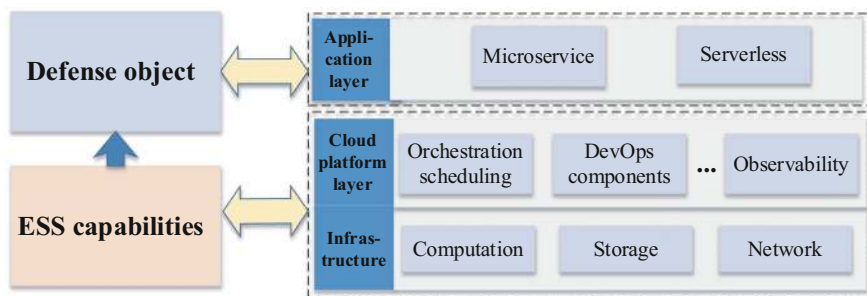


Fig. 7.16 Design idea of an endogenously safe and secure native cloud system

provides a new methodology for solving “unknown unknown” cyberattacks in cloud computing. Under the guidance of the CESS defense paradigm, it is necessary to combine the ESS technology and the cloud-native technology to create an endogenously safe and secure native cloud and provide a highly trustworthy, available, reliable, safe, and secure operating environment for cloud-native applications. The endogenously safe and secure native cloud should have the characteristics of cloud-native transformation, ESS-native transformation, and systematic protection. Its design idea is shown in Fig. 7.16.

First of all, the system design needs to follow the development trend of the cloud computing technology based on cloud-native transformation and provide on-demand basic capabilities for cloud-native applications in the form of servitization through the infrastructure layer, cloud platform layer, and application layer. Specifically, the infrastructure layer provides basic computing, storage, and network resources in cloud computing; the cloud platform layer uniformly manages the underlying resources and provides orchestration scheduling, DevOps, observability, and other functions for upper-layer applications; and the application layer realizes the normal services of cloud-native applications such as microservice and serverless based on the functions provided by the cloud platform layer.

Secondly, ESS capabilities should also be regarded as the core capabilities of the cloud platform, integrated in the cloud computing platform, and provided to applications in a service-oriented mode to achieve on-demand invocation and flexible allocation of ESS capabilities. In order to achieve the goal of ESS-native transformation, it is necessary to integrate the generic components of the strategic ruling feedback control of the DHR architecture, such as input agent, error correction output agent, strategic ruling arbitrator, feedback scheduler, etc., at the cloud platform layer. At the same time, it is necessary to follow the idea of cloud computing resource management to pool these generic components into ESS elements in the resource pool to enable on-demand invocation. In addition, the complete intersection of dynamics, heterogeneity, and redundancy required by ESS needs to be achieved at the infrastructure layer and cloud platform layer. For example, heterogeneity of the operating environment can be achieved by deploying heterogeneous physical servers; redundancy of resources can be achieved by

creating multiple replicas on the cloud platform; and the dynamics of application instances can be achieved through the cloud platform management interface.

7.3.4 Architecture Design

Following the above design idea, the endogenously safe and secure native cloud system architecture is shown in Fig. 7.17. The system can be logically divided into a heterogeneous infrastructure layer, a heterogeneous virtualization layer, and a cloud platform layer. Specifically, the heterogeneous infrastructure layer contains multiple servers configured with different CPUs and operating systems, which serve as computing nodes of the cloud computing cluster. The heterogeneous virtualization layer provides resource virtualization tools for upper-layer applications, including not only the general container virtualization tool Runc but also some cloud-native virtualization tools for special application scenarios, such as the secure container engine Kata and the lightweight virtual machine engine Firecracker. The cloud platform layer, based on the resources provided by the underlying layer, is responsible for the full life cycle management of applications from development, deployment, to operation.

In this logical architecture diagram, the environmental heterogeneity capabilities provided by the infrastructure layer and the virtualization layer serve as the basic environment support for the ESS architecture. The cloud platform layer, based on the basic software/hardware environment, realizes the capabilities required by ESS and

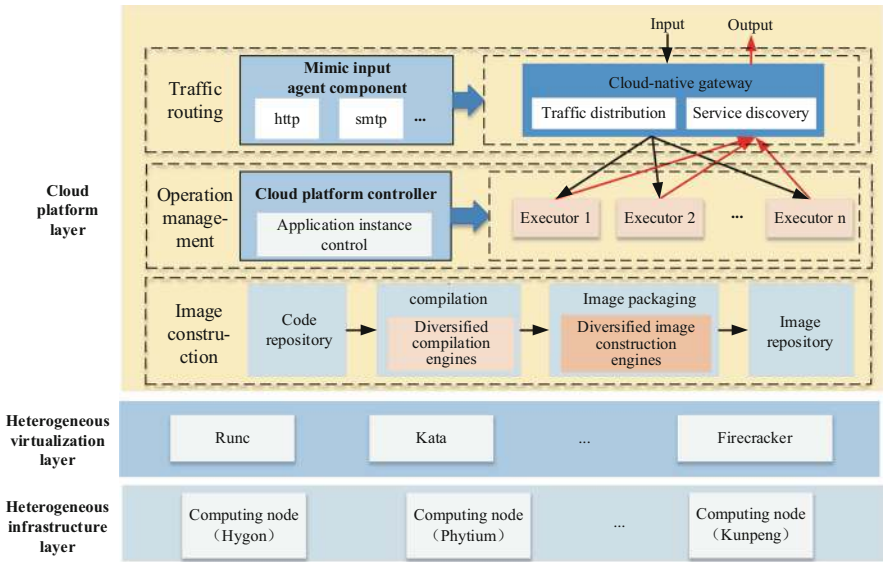


Fig. 7.17 System architecture of an endogenously safe and secure native cloud

is the core of achieving endogenously safe and secure cloud-native transformation. According to the characteristics of the cloud-native architecture and the design idea of ESS-native transformation, the cloud platform layer can be divided into an image construction unit, an operation management unit, and a traffic routing unit.

1. Image construction unit. The purpose of this unit is to provide application heterogeneity based on a continuous integration/continuous deployment (CI/CD) pipeline. In the traditional CI/CD pipeline, operations such as source code compilation, image construction, and application deployment are performed sequentially in an automated manner, which simplifies the manual operations required for application deployment and greatly improves the efficiency of application launch. In the above architecture, in order to achieve application heterogeneity, the heterogeneity engine can be integrated into the CI/CD pipeline to provide users with programmable interfaces and realize application heterogeneity on demand. The above architecture integrates diversified compilation engines and diversified image construction engines in the CI/CD pipeline. Among them, diversified compilation engines can use the software diversification technology during source code compilation to compile heterogeneous executable files from the same source code; diversified image construction engines use heterogeneous container base images and middleware to construct heterogeneous images during image packaging.
2. Operation management unit. This unit is responsible for the heterogeneity, dynamics, and redundancy management of application services during operation. In terms of heterogeneity, it is necessary to select an image for each replica to ensure the heterogeneity between replicas; in terms of dynamics, it is necessary to control the running time of each replica; in terms of redundancy, it is necessary to choose the right number of replicas with due consideration given to such factors as security, performance, and cost.
3. Traffic routing unit. This unit is the access portal of the system and is responsible for automatically sensing the status of the currently running application replica and forwarding external access requests to the application replica. In the face of different service types, gateways of corresponding protocol types need to be deployed online from the input agent component library of the ESS architecture. Different from traditional cloud gateways, this type of security gateway can not only achieve load balancing of access requests but also has an ESS mode, i.e., distributing requests into multiple replicas and making strategic rulings on the execution results of multiple replicas.

7.3.5 System Practice

Based on the architecture of the endogenously safe and secure native cloud system, this subsection introduces some key technical practices in the endogenously safe and secure native cloud system, mainly including cloud-native system network construction, endogenously safe and secure DevOps practices, endogenously safe and secure

native application adaptation framework, and endogenously safe and secure application behavioral status ruling.

7.3.5.1 Construction of the Cloud-Native System Network

The network of the endogenously safe and secure native cloud system is the base of the cloud computing cluster, and the specific network design is shown in Fig. 7.18. It involves four types of networks, three of which are physical networks, namely, the access network, the service network, and the management network, and one of which is a virtual network, namely, the container network.

The access network provides users outside the cloud platform with access to services within the cloud. The access network of the endogenously safe and secure native cloud system needs to meet the following requirements: ① it must be able to support large-scale concurrency; ② it must be able to ensure extremely high reliability; ③ it must be able to implement functions such as mimic distribution and ruling. To meet the above requirements, an m - n two-level agent model is designed in the access network implementation. In the outermost layer, m load balancers form a load balancer cluster; in the inner layer, n mimic input agents

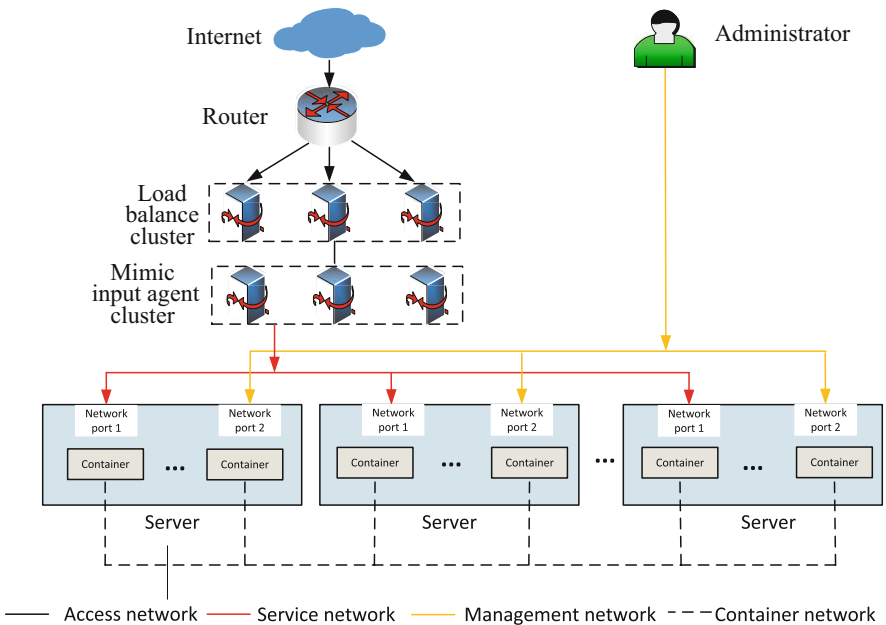


Fig. 7.18 Cloud-native system network construction

form a mimic input agent cluster. Among them, the load balancer plays the role of a four-layer agent, which is developed based on DPVS and achieves high-performance forwarding of data packets through the advantages of the Data Plane Development Kit (DPDK) CPU binding and data lock-free. Meanwhile, the router uses the equal-cost multi-path routing (ECMP) protocol to distribute data packet loads to multiple load balancers, avoiding the problem of single-point failures. The mimic input agent plays the role of both a seven-layer load balancer and a mimic distributor. Generally speaking, each load balancer runs multiple mimetic input agents on its backend, thus increasing the concurrent processing power of the system.

The service network and management network carry east-west traffic within the cluster, and the container network is a virtual network based on the service network and management network. The container network not only needs to ensure normal communication between microservices of ESS applications but also needs to ensure network isolation between executors. In practice, therefore, the container network is designed according to the principle of least privilege, allowing only necessary network communication in the east-west direction. In order to achieve this purpose, the eBPF technology is used to mount the eBPF program at the eXpress Data Path (XDP) mounting point, the traffic control (TC) mounting point, and the socket mounting point of the Linux network protocol stack, thereby realizing fine-grained isolation control of the container network.

7.3.5.2 Endogenously Safe and Secure DevOps Practice

DevOps provides best practices for the development of cloud-native applications, and the endogenously safe and secure DevOps adopts the idea of ESS to design the plan and pipeline ranging from source codes to application instances for endogenously secure native cloud applications, which mainly includes three phases: source code compilation, image packaging, and container deployment.

In the source code compilation and image packaging phases, software diversification techniques such as diversified compilation are integrated into the CI/CD pipeline, using the Tekton project as the framework of the CI/CD pipeline and integrating diversified compilation techniques based on the project. The CI/CD pipeline architecture is shown in Fig. 7.19. Based on this architecture, for each type of language, it is only necessary to prepare the compilation environment in the container environment in advance and then package it into a container image as a resource pool for diversified compilation. When performing diversified compilation tasks, the compilation environment can be selected directly according to the compilation requirements, and the source codes can be compiled. Similarly, heterogeneous or diversified middleware can also be selected during image construction to build heterogeneous images. After the task is completely executed, the task execution environment will be deleted. This CI/CD pipeline automates the generation of heterogeneous images and greatly enriches the heterogeneity of the application of an endogenously safe and secure native cloud.

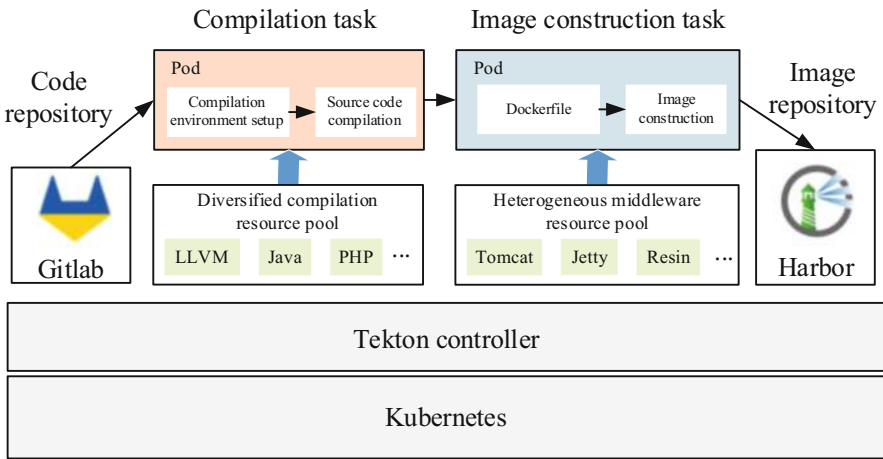


Fig. 7.19 CI/CD pipeline architecture based on Tekton

In the container deployment phase, the heterogeneity of the container operating environment is taken into account, with the goal of maximizing the heterogeneity between executors and determining the deployment location and scheduling strategy of the container. The heterogeneity of the container operating environment is mainly reflected in the following three aspects: ① heterogeneity of the hardware environment, such as hardware servers configured with different CPUs and operating systems; ② heterogeneity of container runtime, such as Runc, Kata, Firecracker, and other container engines with different implementation principles [15]; ③ differentiated configuration of traditional security technologies that are transparent to applications, such as sandboxes, honeypots, firewalls, intrusion detection, and other security software with different defense mechanisms.

7.3.5.3 Endogenously Safe and Secure Native Application Adaptation Framework

In order for a cloud-native application to have ESS capabilities, it is generally necessary to adapt the application to make it an endogenously safe and secure native application. A typical endogenously safe and secure native application architecture is shown in Fig. 7.20. For a cloud-native application with a microservice architecture, it is necessary to first determine the critical defensive positions in the microservices and protect key microservices. The microservices to be protected need to be transformed into a DHR architecture so as to have ESS capabilities. Specifically, when requesting input, the request needs to be distributed to multiple heterogeneous executors of the microservice through the mimic input agent, and rulings should be made on the response of executors. When the executors need to

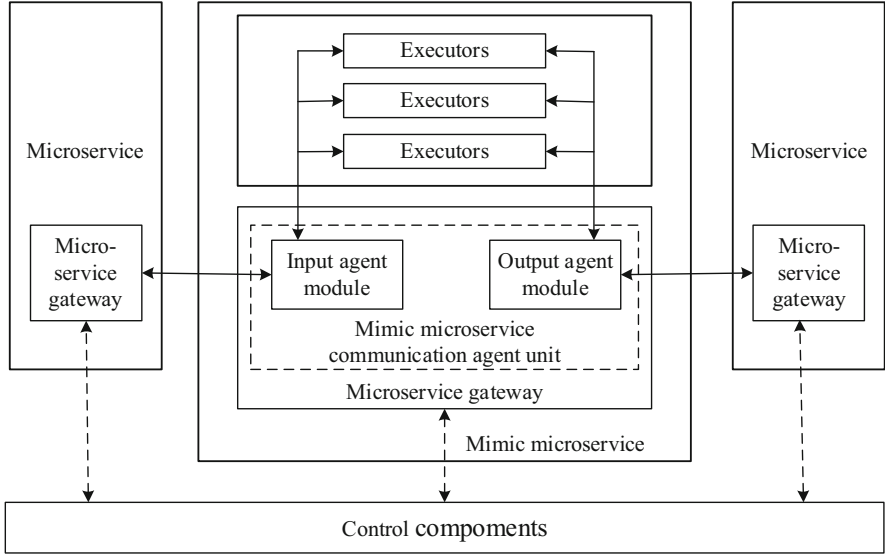


Fig. 7.20 Endogenously safe and secure native application

call other microservices, the newly initiated request is normalized and output via the mimic output agent.

In view of the steps required for the mimic transformation of cloud-native applications, this system adopts an endogenously safe and secure native application adaptation framework with the purpose of “low intrusion” or even “no intrusion” mimic transformation, thereby greatly reducing the workload and difficulty of mimic adaptation of cloud-native applications. This is mainly reflected in the following aspects:

1. The transmission of mimic tags and the observability of mimic applications are realized based on the OpenTracing technology. When the mimic input agent normalizes the newly initiated requests of the executor, if it cannot distinguish the batch of newly initiated requests, it will greatly affect the processing performance and the stability of the strategic ruling. To address this problem, this framework, based on the OpenTracing technology, carries mimic tags in the request interactions of all microservices of the application. Each batch of requests has the same mimic tag. The mimic tag can be used for tracking the request processing and performance of each microservice within the entire mimic application, on the one hand, and for strategic ruling of the mimic output agent, on the other hand.
2. Based on the Kubernetes webhook mechanism [16], the creation of mimic applications is made “nonintrusive.” This mechanism can intercept and modify the user’s application creation request so that the created application has a DHR architecture. By this approach, it is possible to automatically create cloud-native applications with ESS architecture without modifying the user’s deployment files for cloud-native orchestration.

3. Session sharing between heterogeneous executors is realized based on Redis. Current applications often use session to store the status of user access. In the case of multiple executors under the ESS architecture, Redis is used to realize session sharing among multiple executors.

7.3.5.4 Endogenously Safe and Secure Application Behavioral Status Ruling

On the basis of dataflow-based ruling, this system uses the behavioral status of the executor as a post-strategy ruling indicator to further enhance the security of endogenously safe and secure native cloud applications. Its logical architecture is shown in Fig. 7.21.

Due to characteristics such as microservicing, debugging, performance analysis, and fault localization of cloud-native applications are particularly complex. In response to this problem, the cloud-native observability technology is booming and is expected to realize the collection of running status and system call monitoring of each microservice in cloud-native. Based on the cloud-native observability technology and the empowerment of the ESS architecture, this system collects multidimensional information such as the executor's running status, system logs, indicator data, and system calls and models the behavioral status of the executor with this information to construct the state transfer model of the executor. Finally, strategic rulings are made on the behavioral status of the executor as a supplement to the data flow-based ruling to further enhance the security of the system.

7.4 The Internet of Vehicles System Empowered by ESS

7.4.1 *T-BOX Threat Analysis*

7.4.1.1 Basic Functions of T-BOX

In recent years, with the wide application of electronic information, AI and other technologies in the automotive field, intelligence, networking, and electrification have become the development trends of future intelligent networked vehicles, and cybersecurity has also become an important issue in the field of automotive safety. In 2015, two hackers, Charlie Miller and Chris Valasek, remotely took control of the steering, braking, and acceleration systems of the 2015 model of Jeep Cherokee vehicles, which ultimately led to the recall of 1.4 million of those vehicles, causing significant financial losses to the manufacturer. In March 2018, an Uber self-driving car ran into a pedestrian on a public road in Tempe, Arizona. The pedestrian died after being taken to the hospital. The investigation revealed that a series of poor design decisions led to the car's inability to properly handle and respond to the presence of the pedestrian. According to the 2020 Global Automotive Cybersecurity

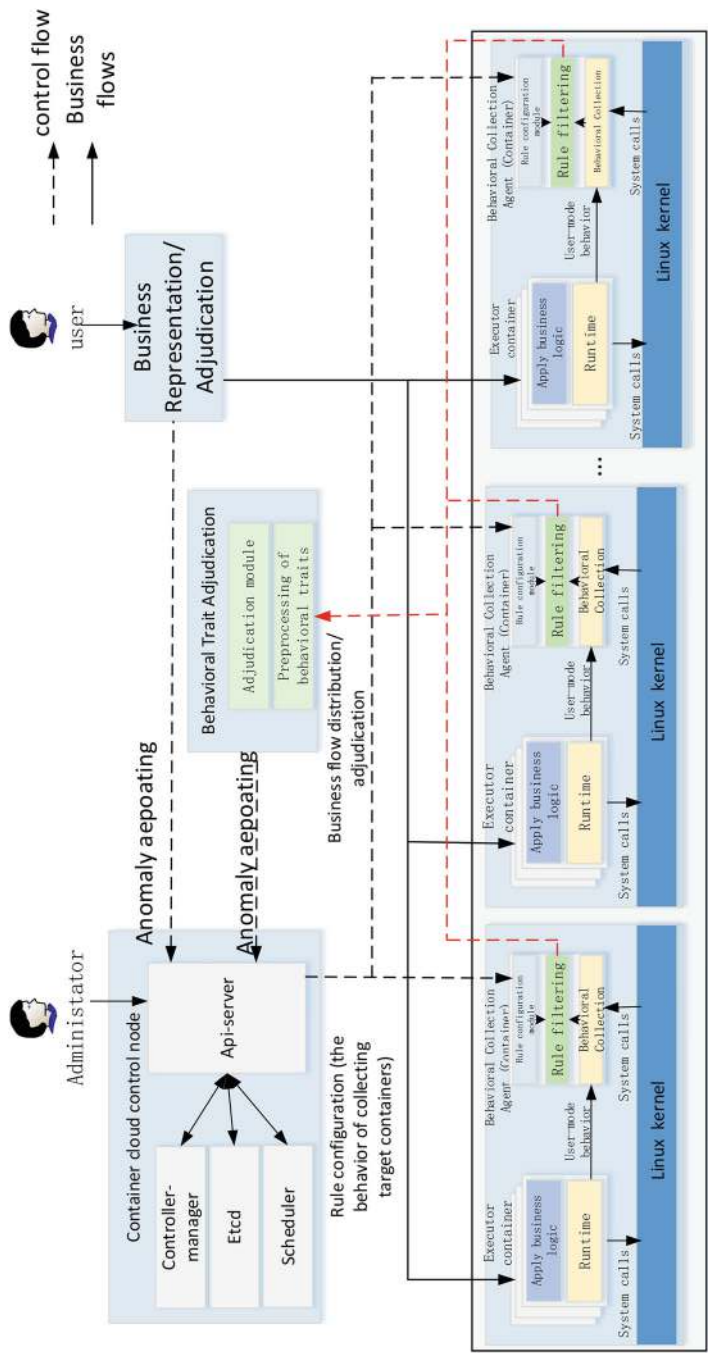


Fig. 7.21 Logical architecture of endogenous safe and secure application behavioral status ruling

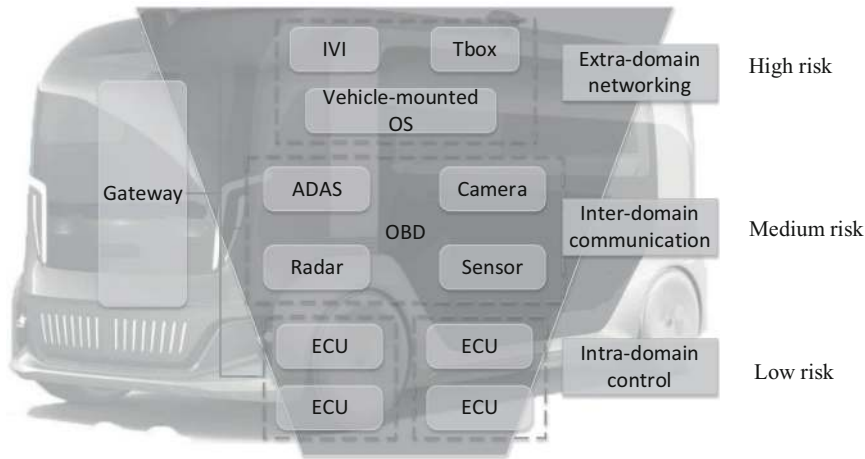


Fig. 7.22 In-vehicle cybersecurity threat level

Report released by Upstream Security, 330 million cars had been networked by the beginning of 2020, and a growing number of networked cars had increased the potential destructive power of cyberattacks targeted at them. A large-scale cyberattack against smart cars could paralyze the entire city and even cause catastrophic loss of life and property, as shown in the Hollywood blockbuster *Fast & Furious 8*. Since 2016, the number of smart car accidents has increased by 605%, more than doubling in 2019 alone. Therefore, safety has become one of the “choke points” in the process toward automotive intelligence and connectivity.

Analyzed from the perspective of cybersecurity, the vehicle-side threat model mainly involves the security risks in in-vehicle gateway, telematics BOX (T-BOX), in-vehicle infotainment (IVI), electronic control unit (ECU), onboard diagnostics (OBD), over-the-air (OTA), automotive operating system (AOS), and in-vehicle networks, as shown in Fig. 7.22 and Table 7.1.

Among them, T-BOX, also known as telematics BOX, is the nerve center of intelligent networked vehicles, responsible for vehicle-to-vehicle, vehicle-to-cloud, and vehicle-to-road communications, as well as remote monitoring, control, and diagnosis of vehicles. T-BOX can deeply read the vehicle’s CAN bus data and private protocols and upload the data to the cloud server through the wireless network. It is the “internal and external communication gateway” or choke point of intelligent networked vehicles and is highly susceptible to remote control, sensitive information leakage, and in-vehicle cyberattacks caused by system vulnerabilities, backdoors, and virus Trojans, as shown in Fig. 7.23.

Table 7.1 Analysis of in-vehicle cybersecurity protection levels

In-vehicle systems	Means of attack	Threat impact	Protection levels
T-BOX	Vulnerability exploitation, protocol cracking, hijacking, DOS attack, information leakage	Remote malicious control, sensitive information leakage, in-car network attack	High
IVI	Vulnerability exploitation, privilege escalation, hijacking, bypass, data theft	Remote malicious control, sensitive information leakage, in-car network attack	High
AOS	Vulnerability exploitation, privilege escalation, buffer overflow, DOS attack	Malicious acquisition of system privilege and arbitrary application installation	High
In-vehicle gateway	Deception, hijacking, DOS attacks	Malicious tampering with messages, denial of service	Middle
OTA	Man-in-the-middle attack, deception, tampering	Malicious tampering with upgrade packages, malicious system writing	Middle
OBD	Privilege escalation, deception, tampering	Malicious message execution, denial of service	Middle
ECU	Replay, denial of service	Functional tampering, denial of service	Low

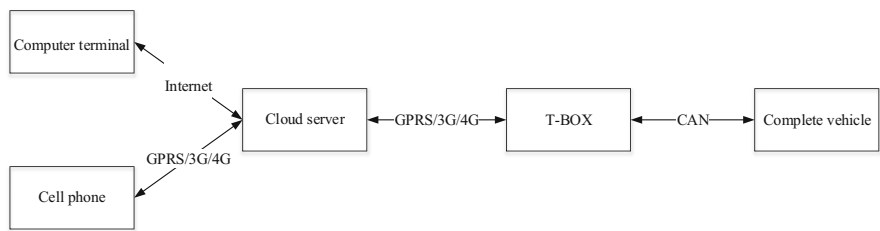


Fig. 7.23 Schematic diagram of T-BOX network connection

7.4.1.2 T-BOX Security Threats

T-BOX is mainly composed of a microcontroller unit (MCU) and a communication module. MCU is mainly responsible for the reception and processing of the vehicle’s CAN network data, information uploading, power management, data storage, fault diagnosis, and remote upgrading; the communication module is mainly responsible for the network connection and data transmission, providing Wi-Fi hotspots for users and Internet access for IVI. As shown in Table 7.2, T-BOX mainly faces threats such as protocol cracking, DOS attacks, information leakage, and session hijacking.

Table 7.2 List of major threats to T-BOX

Targets of attack	Means of attack	Cost of attack	Physical contact	Description
T-BOX	Protocol cracking	High	Yes	Cracking keys, decrypting communication protocols, replaying and tampering with instructions by means of reversing firmware
	DOS attack	Middle	No	T-BOX being unable to receive normal commands and respond to TSP services
	Information leakage	High	Yes	Attackers being able to find better attack paths by accessing internal data through the debugging interface
	Hijacking	Middle	No	Hijacking T-BOX sessions to take remote control of vehicle-body control domain and automotive powertrain domain through forged protocols

7.4.1.3 Causes for Threats

For a long time, the automotive industry has been focusing on the issue of functional safety and provides functional safety development guidance to vehicle and component manufacturers based on the ISO26262 standard. With the development of automotive digitization, intelligence, and networking, cybersecurity has gradually received more attention in recent years. For intelligent networked vehicles at present, the development of cybersecurity technologies lags far behind the development of functional safety technologies. In fact, however, key components of smart vehicles rely on both functional safety and cybersecurity in most cases. How to solve the intertwined problems of cybersecurity and functional safety at the system architecture level is a research gap at home and abroad.

Currently, intelligent vehicles are gradually evolving into a physical-information fusion system integrating perception, computation, network, and control, in which the in-vehicle network, as the “nerve conduction system” of the entire intelligent networked vehicle, is becoming one of the main targets of network attacks, in addition to facing traditional reliability challenges. Since the in-vehicle network faces a large network attack surface, numerous security protection links, and large differences in available resources, it is inevitable that a certain link in the product cannot realize sufficient security protection. For example, a typical vehicle network system is composed of key network components such as T-BOX, IVI, in-vehicle gateway, and ECU, and these networked components often have widely varying computing, storage, and bandwidth resources due to their own functional and design requirements. For many components with limited resources, the deployment of new network defense measures may face the dilemma that “a clever woman cannot cook a meal without rice.” Moreover, the in-vehicle network also faces complex cybersecurity requirements. On the one hand, it has heterogeneous communication formats such as serial ports, Ethernet, CAN ports, 4G/5G, etc. On the other hand, each heterogeneous sub-network faces different attack surfaces or levels of security

threats. Therefore, how to strategically deploy “critical defensive positions” according to the characteristics of the in-vehicle network has become crucial.

GFS problems are one of the biggest challenges facing the industrial development of intelligent networked automobiles. Undoubtedly, the unavoidable “unknown security defects” of software and hardware will be a major cause for quality problems in future intelligent networked automobiles. In the absence of effective solutions to the common ESS problems in the network, it is necessary to establish a mechanism for pre-discovery, repair, and real-time detection of and response to unknown defects. Currently, the automotive industry generally adopts the OTA (over-the-air download, or online upgrade) technology based on the “patch methodology” to occasionally fix software bugs and update software versions. However, vehicles are not like ordinary digital products after all. The latter can meet the basic safety and security requirements as long as their vulnerabilities or defects are fixed in a timely manner, but the former may cause irreparable losses of life and property due to a slight error. Moreover, there is no way to know whether new defects, vulnerabilities, or backdoors will be introduced after each time of OTA “patching.” In view of this, people cannot help but ask: will intelligent networked cars face the same dilemma as the traditional ICT industry that “only after a sheep is lost will the sheepfold be mended”?

The mission and responsibility of automobiles to carry life has not changed since the birth of the automotive industry. Therefore, under the trend of intelligence and networking, new “onboard” software and hardware must still adhere to serving and being responsible for life. With the continuous infiltration of factors of man-made cyberattacks into the field of traditional functional safety, the interweaving and superposition of unknown defects, vulnerabilities, and backdoors have made many assumptions about traditional functional safety difficult to remain valid. Therefore, according to the heterogeneous characteristics and differentiated security requirements of in-vehicle networks, it will undoubtedly be of great significance and value to explore and establish an integrated protection architecture that ensures the functional safety and network security of automobiles and effectively copes with network attacks based on unknown vulnerabilities and backdoors of software while providing highly reliable communications.

7.4.2 Design Ideas

7.4.2.1 Defense Model

How to ensure functional safety in the face of cyberattacks has become a fundamental and critical challenge to the global automotive industry. To address this challenge, the author’s research team, based on the original ESS theory and technology and working with leading passenger car and commercial vehicle manufacturers, has been exploring new theories and paths to crack the intertwined problems of functional safety and cybersecurity of intelligent networked vehicles in an effort

to contribute Chinese wisdom and China's solutions to the breaking of the major bottlenecks restricting the development of the industry.

In view of the unknown security threats and damages caused by unknown vulnerabilities and backdoors in the traditional T-BOX, the ESS-empowered T-BOX adopts an ESS mechanism based on a generalized robust control structure on the basis of traditional functions and innovatively applies the DHR architecture in the automotive network devices, which can sense and suppress uncertain threats or random system failures without relying on the prior knowledge about attackers or their behavioral characteristics and additional security technologies, so as to ensure vehicle driving safety and secure trustworthy interaction of information inside and outside the vehicle. As shown in Fig. 7.24, the T-BOX with ESS structure introduces the defense triad of dynamics, variety, and redundancy (DVR) into T-BOX system design, distributes the heterogeneous interface input data of T-BOX to multiple heterogeneous functionally equivalent executors, and identifies anomalous responses based on strategic ruling over associated statuses or output messages, so as to suppress random hardware failures and uncertain software errors caused by disturbances of natural factors and effectively control security threats and attacks that exploit the vulnerabilities, backdoors, and virus Trojans in the system.

7.4.2.2 Defense of Critical Positions

The in-vehicle network connection system T-BOX carries the functions of Bluetooth, Wi-Fi, cellular network, and other information interaction channels. It is the gateway for internal and external communication of intelligent networked cars. On the one hand, it is extremely vulnerable to remote attacks caused by vulnerabilities and backdoors in the system; on the other hand, in-vehicle information lacks effective supervision, and where sensitive in-vehicle information goes is not controlled. Since T-BOX is responsible for information transmission inside and outside the vehicle, it is necessary to analyze its critical defensive positions and focus on ESS planning. According to network communication functions, the in-vehicle network is divided into three domains: the boundary domain of networked terminals responsible for vehicle-cloud communication; the gateway domain responsible for infotainment, intelligent driving, human-machine interaction, OBD, OTA, and other information interactions; and the controller domain responsible for the CAN network.

1. The boundary domain of networked terminals is a key node for in-vehicle and out-of-vehicle exchanges. It generally adopts boundary firewall isolation technologies such as local service access control, data forwarding access control, domain name-based access control, application layer content-based access control, and security auditing.
2. The gateway domain is responsible for protocol conversion and information interaction of different ECU domains. It generally adopts the inter-domain firewall isolation technology with black and white lists and privilege control to

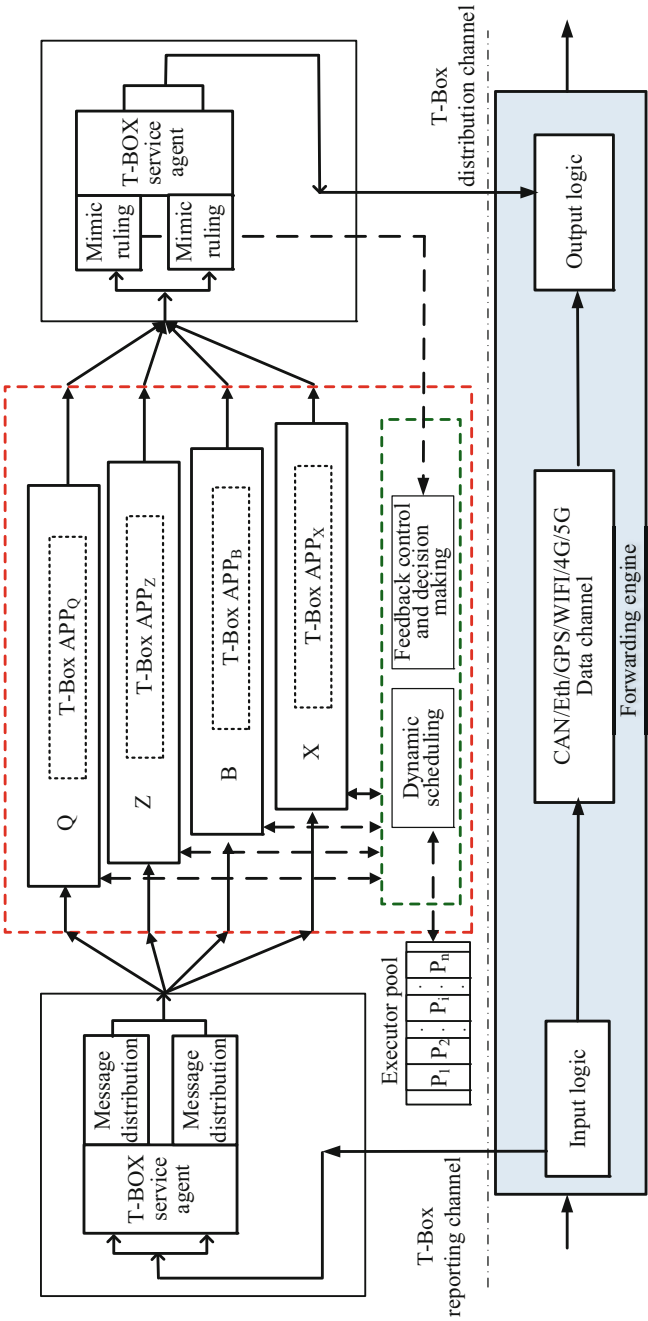


Fig. 7.24 Flowchart of the T-BOX's ESS structure

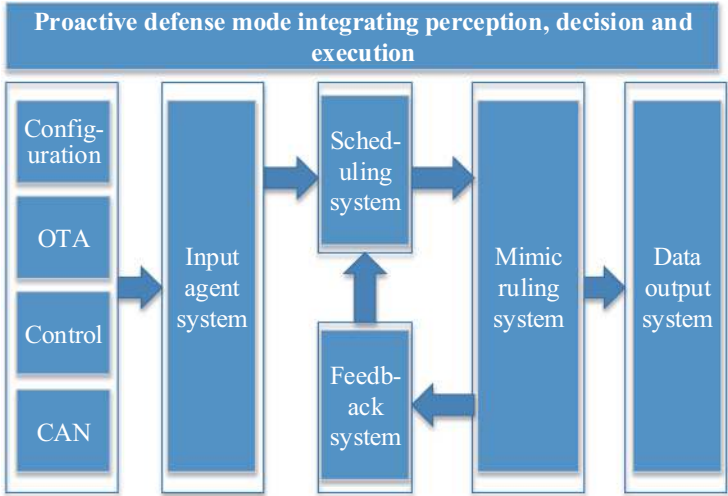


Fig. 7.25 Architecture of defense of critical positions in vehicle-mounted T-BOX

implement inter-domain communication access control for sensitive areas of intelligent driving in the vehicle and the OBD and IVI functional domains, so as to ensure high reliability and trustworthiness of intelligent driving functions.

3. The network controller domain is responsible for vehicle-body data collection and reporting and controller message distribution. It generally uses CAN firewall technologies such as protocol consistency, boundary checking, and black and white lists to realize intra-domain security.

T-BOX is involved in these three domains, so it can be said that each domain is a “critical position” where T-BOX needs to make security enhancements. To this end, configuration, OTA, control, and CAN message collection are regarded as critical defensive positions to be empowered by the ESS architecture, and an adaptive defense model integrating perception, decision, and execution is proposed, as shown in Fig. 7.25. The realization of these functions is incorporated into the DHR architecture of ESS, and the T-BOX component functions with GFS attributes are formed through the DHR-constructed heterogeneous redundant executors, input agents, scheduling feedback, strategic ruling, and error correction output.

7.4.3 System Architecture Design

7.4.3.1 Overall Architecture

The endogenously safe and secure T-BOX system architecture design incorporates the integrated deployment concept of GFS and takes core services such as data reporting, configuration issuance, control demands, and OTA as the focus of ESS

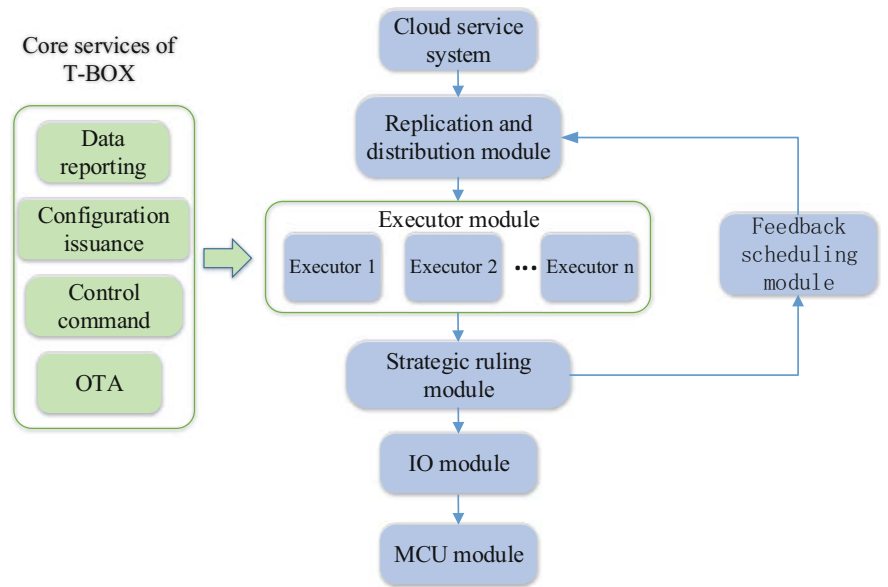


Fig. 7.26 Overall composition framework of the endogenously safe and secure T-BOX system

protection, so as to improve the accuracy of service data and the security of issuance. At the same time, in view of the scarcity of hardware resources in the T-BOX system, technologies such as virtualization and microservices are adopted to realize the lightweight deployment of the DHR architecture as well as the technical-economic benefits of the highly reliable, highly trustworthy, and highly available endogenously safe and secure T-BOX system in coping with generalized uncertain disturbances within the architecture. The endogenously safe and secure T-BOX system mainly includes a replication and distribution module, an executor module, a strategic ruling module, and a feedback scheduling module. Its overall composition framework is shown in Fig. 7.26.

7.4.3.2 Lightweight Design of the System Architecture

Drawing on the existing mainstream T-BOX system architecture design ideas, the endogenously safe and secure T-BOX system architecture adopts the design model of “communication module + SoC + MCU.” Among them, the communication module is responsible for mobile network connection, satellite positioning, and some security functions; the system-on-chip (SoC) is responsible for implementing application functions, mainly service deployment functions, ESS, general security, etc.; the MCU is mainly responsible for accessing the automotive CAN bus and realizing power management control and access to CAN, RS232, RS485, and other channel signals. The real-time nature of the MCU is utilized to process vehicle

control information; control the power-on, power-off, and startup of SoC and the communication module; and reduce the standby power of the device. The hardware architecture of the endogenously safe and secure T-BOX system is shown in Fig. 7.27.

Since T-BOX has very limited computing and storage resources, in order to save onboard resources and control costs, the architecture design of endogenously safe and secure T-BOX follows the “lightweight and low-cost” principle and introduces virtualization, microservices, and other technologies, aiming to integrate different types of instruction sets and architecture computing units and collaboratively use different types of computing resources according to the actual needs of service computing to achieve the lightweight deployment of parallel services and high-performance computation of the T-BOX system.

As shown in Fig. 7.28, in the T-BOX system empowered by the DHR architecture, the virtualization technology is used at first to deploy multiple Docker containers on the SoC to build mutually independent and isolated executor environments. Then lightweight microservices are used as the service carrying mode, and a fine-grained, lightweight T-BOX service multi-executor construction scheme is designed at the architectural level to establish a segmentation and coupling mechanism of core service functions such as control command issuance and status data reporting, realize fine-grained splitting and construction of service functions, and ensure the independence and scalability of each service. At the same time, through the heterogeneity of compilation environment and code language, functionally equivalent variants for key microservice functions are generated to achieve lightweight ESS construction of the core service functions.

7.4.4 Design of Functional Units

7.4.4.1 Strategic Ruling and Scheduling Module

The strategic ruling and scheduling module includes two parts: strategic ruling and executor scheduling. The strategic ruling module is divided into sub-modules such as data reception, executor status monitoring, and ruling. The scheduling module includes sub-modules such as data reception and sending, scheduling strategy, and timing scheduling. In the initial state, the executor scheduling module selects three executors and schedules them online. After the executor initialization is completed, the distribution module inputs data to the executor module. The strategy ruling module receives the data of the executor module for ruling and sends the corresponding data to the I/O module or the scheduling module, and the latter completes the online and offline operations of the executors through the control module. The composition framework of the ruling module and the scheduling module is shown in Fig. 7.29.

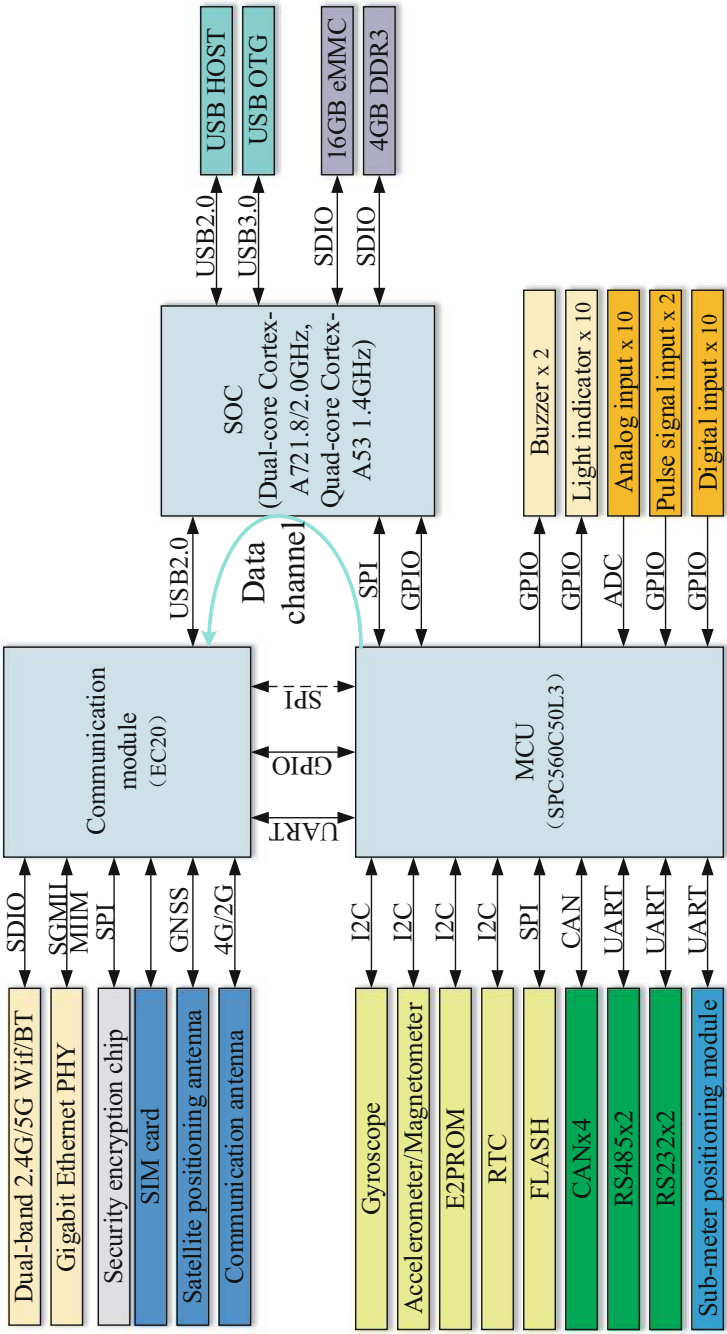


Fig. 7.27 Hardware architecture of the endogenously safe and secure T-BOX system

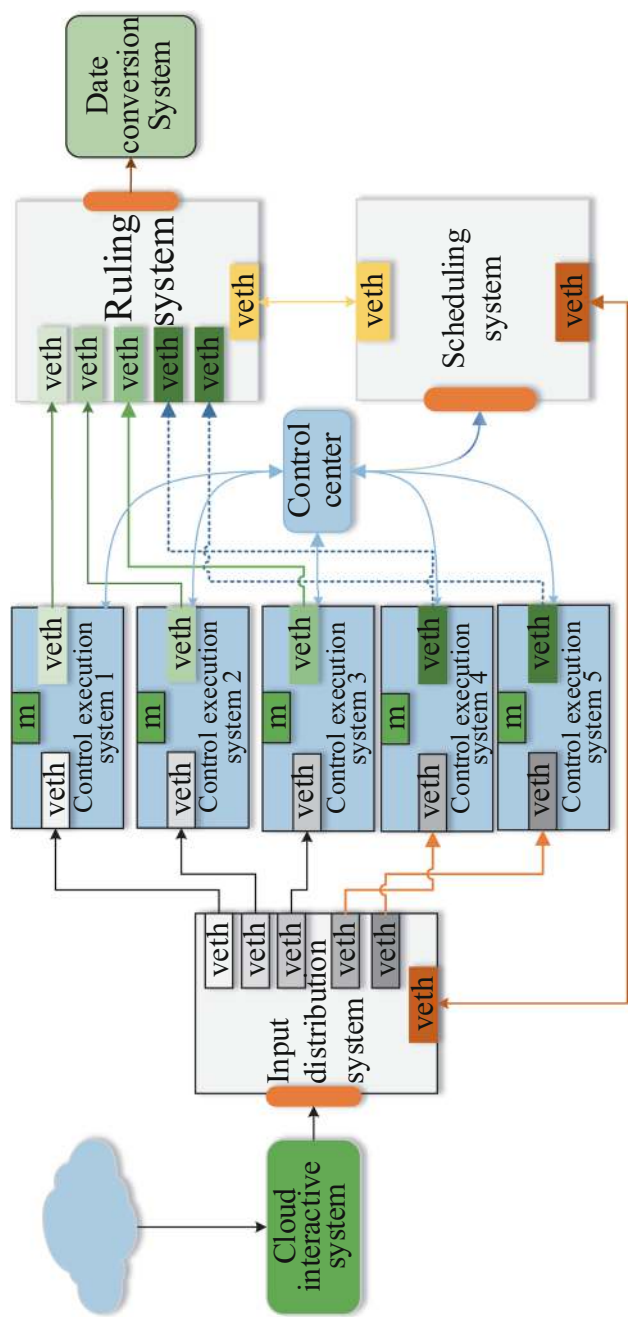


Fig. 7.28 Deployment of the lightweight architecture based on virtualization and microservices

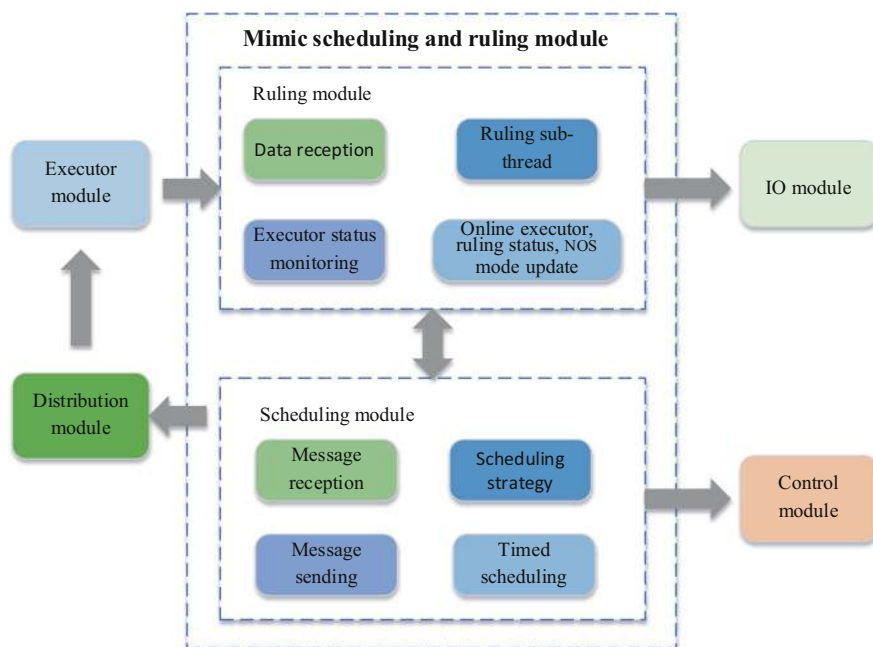


Fig. 7.29 The composition framework of the ruling module and the scheduling module

7.4.4.2 Control Command Issuance Module

Control demand issuance means that vehicle-control commands issued from the cloud are sent down to the automotive ECU through the replication and distribution module, the executor module, the strategic ruling module, and the I/O module to complete the control of the vehicle. The control demands are mainly related to power supply, window/door switches, lighting control, air conditioning control, etc. but less involved in vehicle motion control at present.

As the core of the automotive electronic control system, the MCU module controls various electronic systems inside the vehicle and has many interfaces, such as CAN, SPI, RS232/485, etc. The MCU module is directly connected to the automobile and plugged into the automotive CAN bus. It controls each ECU unit in the vehicle through CAN messages. The CAN messages are transmitted through the CAN bus in the form of broadcasts, and each automotive component only receives CAN messages related to it by retrieving CAN 2. The DHR architecture-empowered T-BOX's control command issuance module controls most of the basic functions of the vehicle. The structure of the control command issuance module is shown in Fig. 7.30.

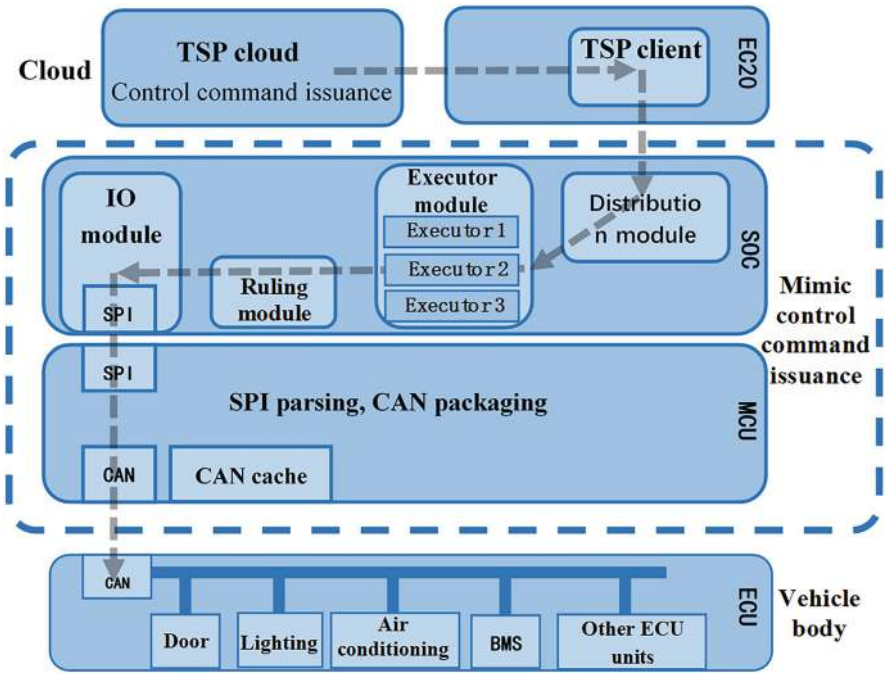


Fig. 7.30 Functional framework of the control demand issuance module

7.4.4.3 Configuration Information Issuance Module

The main function of the configuration information issuance module is to support the distribution/change of configuration parameters from the cloud and make them effective instantly in a real-time running system. The internal components and functions of this module include cloud (TSP), replication and distribution, executors, ruling, and I/O sub-modules, as shown in Fig. 7.31.

Among them, the TSP sub-module is used to issue configuration commands and communicate with the IO sub-module individually; the replication and distribution sub-module is used to replicate and issue configuration commands; the ruling sub-module is used to parse commands and perform heterogeneous processing; and the I/O sub-module is the core module for configuration information processing and is mainly responsible for realizing such functions as configuration command parsing, service processing, and data updating.

7.4.4.4 Vehicle Status Data Reporting Module

Vehicle status data reporting means that each ECU status data is sent to the T-BOX in the form of CAN messages through the CAN interface and then sent to the cloud after further protocol conversion and encapsulation and is presented in a diversified

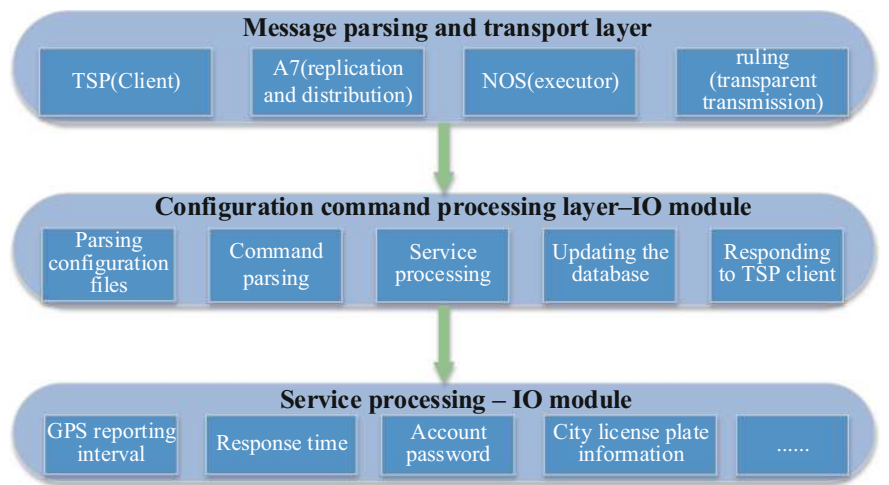
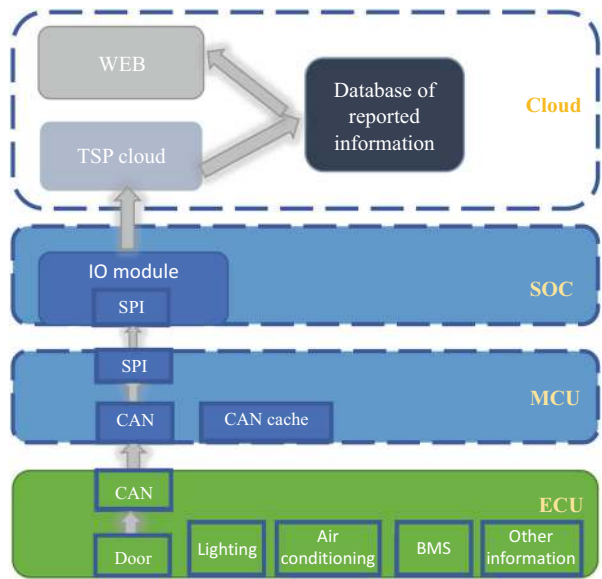


Fig. 7.31 Configuration delivery module framework

Fig. 7.32 Composition framework of the vehicle status data reporting module



manner on the Web page. The vehicle status reporting mainly involves window/door switch status, lighting status, air conditioning control status, IBS/vehicle speed status, EPS steering status, and vehicle control information. The vehicle status data reporting module of T-BOX can complete the reporting of most of the basic status information of the vehicle. The composition framework of this module is shown in Fig. 7.32.

Currently, T-BOX supports the reporting of such data information as ① the window/door switch status, mainly including the door switch status, the anti-pinch status, the emergency status, and the overload status; ② the lighting status, mainly including the double blinkers, the horn signal, the left turn light, the right turn signal, the brake light, the front fog light, and the reversing light; ③ the air conditioning control status, mainly including the air conditioning temperature status; ④ the IBS/vehicle speed status, mainly including the brake pedal opening, the vehicle speed, the driver command gear, the VCU respond gear, the actual accelerator pedal opening, the exit reason, and the feedback mode; ⑤ the EPS steering status, mainly including the steering wheel angle, the steering wheel rotation speed and rotation degree, the steering wheel torque value, the EPS working mode, and the EPS cycle count; and ⑥ the vehicle control information, mainly including data such as the battery charge status.

7.4.4.5 Boundary Security Access Authentication Module

The main function of the boundary security access authentication module is to ensure that the T-BOX is registered in the cloud system for authentication online. It supports the T-BOX terminal in safely receiving commands issued by the cloud and reporting terminal data to the cloud. As shown in Fig. 7.33, its main components include sub-modules such as the TSP server, Jt808 parsing, data distribution and

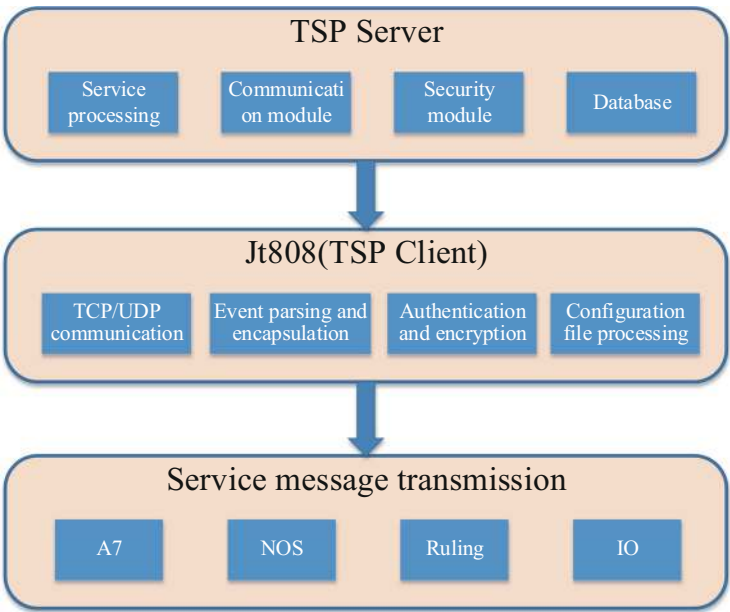


Fig. 7.33 Composition framework of the boundary security access authentication module

ruling, etc. Among them, the TSP server is used to manage and control the T-BOX terminal, issue commands, and collect terminal data; the Jt808 parsing sub-module is responsible for the communication between the terminal T-BOX and the cloud, realizes the registration and authentication of the terminal T-BOX in the cloud according to the standard protocol, and carries out service communication after success; the data distribution sub-module receives data from the Jt808 parsing sub-module, replicates and distributes it to the executor module, and passes it to the IO sub-module through the ruling module; the IO sub-module is responsible for parsing configuration commands and notifying the response information to the Jt808 parsing sub-module.

7.4.4.6 Executor Running Status Monitoring and Early Warning Module

The executor running status monitoring and early warning module monitors the key operating resources in the executor in real time. When abnormal resources are monitored, an alarm is triggered, and timely measures are taken to ensure stable operation of the system. At the same time, the alarm information is stored in the database for subsequent verification, providing data support for comprehensive evaluation of terminal security levels, abnormal alarms, and other functions. The resources monitored by this module mainly include status alarms and trigger alarms, specifically including the core process, core path, and key files within the executor, key network card traffic, the number of executors online, executor CPU utilization, memory utilization, hard disk utilization, executor login status, etc. The framework of this module is shown in Fig. 7.34.

7.4.4.7 Remote Upgrade Management OTA Module

In order to improve vehicle informationization and software management capabilities, it is imperative to build an OTA and information security management platform. The remote upgrade management OTA module involves three parts: the cloud platform, the repository, and the vehicle-mounted T-BOX terminal. The cloud platform is responsible for the update of the version in the repository and the upgrade control of the vehicle terminal version; the SFTP repository is used to store all versions of T-BOX for vehicle terminal upgrades; based on this, the vehicle-mounted T-BOX can complete the OTA upgrade function. The advantages of the OTA function in vehicle development are becoming increasingly prominent. It has the advantages of repairing software defects and security vulnerabilities, introducing and iterating new functions, etc. It can greatly reduce after-sales costs and save repair and maintenance time for customers. The composition framework of the remote upgrade management OTA module is shown in Fig. 7.35.

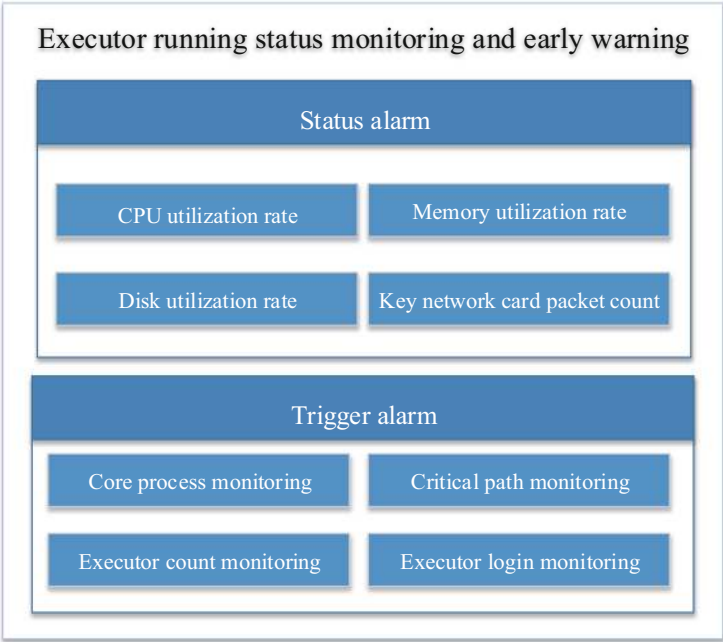


Fig. 7.34 Composition framework of the executor running status monitoring and early warning module

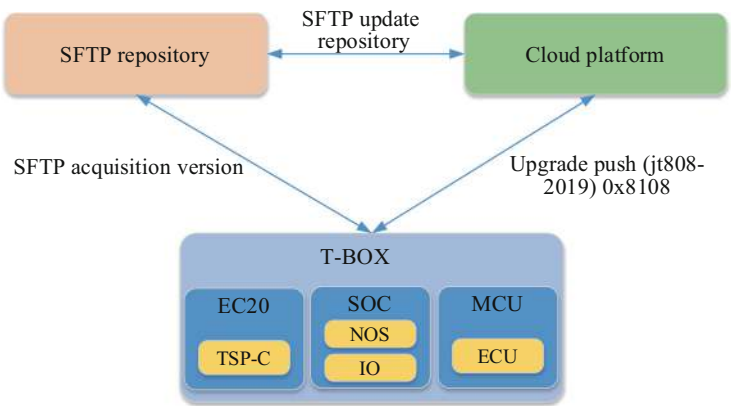


Fig. 7.35 Composition framework of the remote upgrade management OTA module

7.4.4.8 TSP Cloud Service System

Relying on the main functions of the T-BOX system, the functional framework of the TSP cloud service system includes a basic framework layer, a data processing layer, a service logic layer, and a system display layer. The system focuses on

supporting a full-view, multi-granularity visual display of the T-BOX system's operating status, as well as data-driven vehicle anomaly detection and analysis functions. On the one hand, the visual display function intuitively, accurately, and effectively presents the main data of the T-BOX system in the form of graphics, text, reports, maps, etc. On the other hand, the data-driven vehicle anomaly detection and analysis function utilizes AI technologies to train a large amount of data to obtain anomaly detection models and rules, thereby realizing the ability to detect and warn of vehicle anomalies in real time. The component framework of the TSP cloud service system is shown in Fig. 7.36.

7.4.4.9 Interface Modules

The main communication interfaces of the T-BOX system include communication between the cloud and the communication module, the communication module and SoC, SoC and MCU. The main interactive interfaces are shown in Table 7.3.

1. Cloud and communication module. In accordance with relevant national regulations and the general practice of automobile manufacturers, the JT/T808 protocol (2013) and the GB-T32960 standard are adopted. JT/T808 uses the TCP or UDP protocol, which is applicable to communication between the cloud platform and the satellite positioning system (BeiDou-compatible vehicle-mounted terminal) for road vehicles. TCP protocol communication is applicable to data communication between the onboard terminals of pure electric vehicles, plug-in hybrid vehicles, and fuel-cell electric vehicles, automobile manufacturers' platforms, and public platforms.
2. Communication module and SoC. According to the hardware connection scheme, the communication module and SoC are connected through a USB port, and all information sending and receiving operations are realized by the USB interface program.
3. SoC and MCU. Using the SPI interface, the driver layer provides an encapsulation interface, and the upper layer adopts a unified interface program, which needs to support CAN security protection to prevent tampering, fraudulent use, etc.

7.4.5 Feasibility and Safety Analysis

7.4.5.1 Cost Analysis

According to the overall scheme of T-BOX empowered by the DHR architecture and referring to the traditional T-BOX architecture, the increase in production cost of T-BOX with ESS structure is mainly divided into two parts. One is the platform cost of hosting ESS component software such as protocol agents and ruling modules; the other is the platform cost of hosting heterogeneous executor pool resources.



Fig. 7.36 Component framework of the TSP cloud service system

Table 7.3 Main interactive interfaces of the T-BOX system

Interface name	Interface function	Interface properties	Interface rate	Interface Protocol
CAN	Receiving vehicle-body data/OTA upgrade	Multi-master bidirectional	250 kbps	SAEJ1939/UDS
UART	4G module communication, sending real-time information and machine maintenance data;	Bidirectional full-duplex	115,200 bps	TCP/IP JT/T808
	GPS module to obtain satellite positioning information	Bidirectional full-duplex	9600 bps	NMEA
SPI	External FLASH	Bidirectional synchronous full-duplex	8 Mbps	Customized
	Receiving the content of SoC parameter setting and upgrading program files; sending terminal status information and vehicle data	Bidirectional synchronous full-duplex	8 Mbps	Customized
I2C	Gyro	Bidirectional half-duplex	400 kbps	Customized
	Geomagnetic sensor, acceleration sensor	Bidirectional half-duplex	400 kbps	Customized
	E2PROM	Bidirectional half-duplex	400 kbps	Customized
RS485	Fuel-level monitoring	Bidirectional half-duplex	9600 bps	Customized

Because the T-BOX with DHR structure uses the virtualization technology to build a heterogeneous executor pool based on Docker containers on SoC, the executors run on the computation processing card in the form of virtual functions, thereby getting rid of the constraints of physical hardware resources, reducing production cost and maintenance cost (there is also the risk of reducing the isolation degree of the container), and empowering the heterogeneous executor pool to expand flexibly. The endogenously safe and secure T-BOX architecture adopts the hardware design scheme of the “communication module + SoC + MCU,” which is similar to the hardware architecture of commercial T-BOXs of the same class in the market, without additional hardware deployment cost. However, it places higher demands on the virtualization and processing capabilities of SoC, although the architectural design and implementation deployment adopted do not cause additional hardware resource consumption, and due to the integration of virtualization, microservices, and other technologies, it enables lightweight deployment as permitted by the ESS structure. Given that the processing capabilities of current mainstream SoCs are improving rapidly, mid-level SoCs can meet the functional requirements of the endogenously safe and secure T-BOX. Therefore, the cost of the DHR architecture-empowered T-BOX is basically the same as that of the existing mainstream T-BOXs. Since no special network adjustments are required, the

deployment cost of the endogenously safe and secure T-BOX is the same as that of ordinary commercial T-BOXs.

It should be pointed out that the endogenously safe and secure T-BOX, on the basis of realizing the information communication and other functions of the traditional T-BOX, has a DHR structure that is immune to and protects against potential internal vulnerabilities, backdoors, virus Trojans, network threats/attacks, and traditional functional safety disturbances. In other words, while basically not increasing the cost of resource consumption, it significantly improves the functional safety and network security protection capability of the endogenously safe and secure T-BOX system, with unparalleled cost performance. In particular, it provides a fundamental solution to the global problem that traditional T-BOX systems cannot cope with unknown network attacks.

7.4.5.2 Defense Effectiveness Analysis

Based on the innovative ESS mechanism, the mimic-structured T-BOX acquires the ESS function through DHR; realizes generalized robust service performance of “high reliability, high trustworthiness, and high availability”; and theoretically has a deterministic defense effect against known or unknown security threats based on vulnerabilities and backdoors in the mimic interface. In response to the security risks faced by T-BOXs, the mimic-structured T-BOX adopts a critical-position defense strategy and carries out ESS reinforcement for key functions such as configuration, control, CAN information collection, and OTA. Therefore, it makes it impossible for attackers to realize route tampering in the case of a single executor being controlled; even in the case of multiple executors being controlled, it can still correct and clean away the abnormal state or information through the feedback control and dynamic scheduling mechanism, thereby effectively resisting configuration modification and control command tampering based on ESS problems and ensuring that the T-BOX performs the normal information collection and control functions.

7.4.6 Attack and Defense Examples

To demonstrate the security gains brought by the endogenously safe and secure T-BOX, this subsection uses a typical experimental case to show its defense effectiveness. The endogenously safe and secure T-BOX prototype that has been realized previously is shown in Fig. 7.37.

For comparative analysis, a commercially available T-BOX from a certain manufacturer was used for the experiment. It was found that the T-BOX uses the Ubuntu 18.04 version of the system, which has a Polkit pkexec privilege escalation vulnerability (CVE-2021-4034). We deployed an executor of the same Ubuntu 18.04 version on the endogenously safe and secure T-BOX, finding that attackers can still exploit this vulnerability to achieve privilege escalation on the executor.

Fig. 7.37 Prototype of the endogenously safe and secure T-BOX system

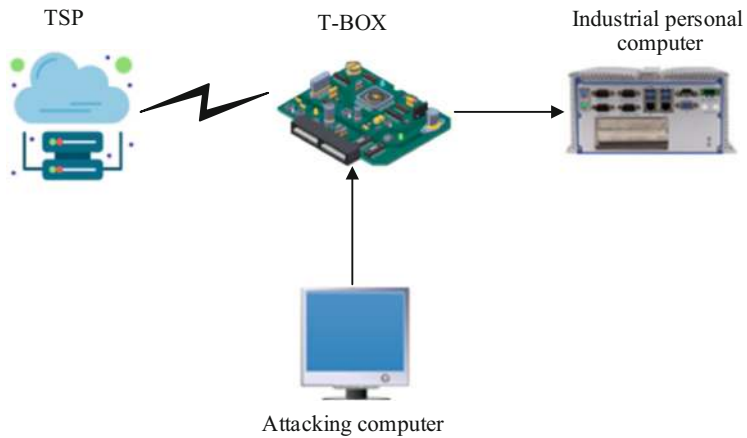


Fig. 7.38 Networking diagram against attacks on T-BOX

The Polkit privilege escalation vulnerability (CVE-2021-4034) is caused by pkexec’s inability to correctly handle call parameters, thus executing environment variables as commands. As a result, an attacker with arbitrary user privileges can exploit this vulnerability by modifying environment variables under the default configuration to gain root privileges on an affected host. Polkit is an application-level toolset in UNIX-like systems to enable communication between processes of different priorities by defining and auditing privilege rules. Pkexec is part of the Polkit open-source application framework that enables authorized unprivileged users to execute commands as privileged users according to defined policies.

This vulnerability affects all Polkit versions released from May 2009 to the present. Since Polkit is preinstalled on multiple Linux versions such as CentOS, Ubuntu, Debian, Redhat, Fedora, Gentoo, Mageia, etc., all Linux systems with Polkit are affected.

The experimental network against T-BOX attacks is shown in Fig. 7.38. The attack process is as follows:

1. Using brute-force cracking and other means to obtain the login name and password of an ordinary user who logs in to T-BOX (test/test).

Fig. 7.39 The door of a simulated moving car is opened



2. Checking the version of the T-BOX's operating system and finding that Ubuntu 18.04 is used and that the version has a Polkit pkexec privilege escalation vulnerability (CVE-2021-4034).
3. Taking out the T-BOX firmware program, conducting reverse analysis, and obtaining the control demands for opening the door.
4. Exploiting iptables to introduce the attack mechanism into the inbound message sent by TSP to T-BOX.
5. Using Burp Suite to capture the message on the attacking machine, tampering with it, and then playing back the tampered message.
6. Externally connecting an external industrial personal computer to the CAN port of T-BOX, installing the Carla automatic driving simulator on the industrial personal computer. After the attack is seen, the door of the simulated moving car is opened, as shown in Fig. 7.39.

According to the ESS theory, the mimic T-BOX empowered by the DHR architecture uses five heterogeneous executors, of which executors 1, 3, and 4 are online, and a version of Ubuntu 18.04 with a privilege escalation vulnerability is preinstalled on executor 1. The status of the vehicle's mimic T-BOX executor can be seen on the TSP cloud, as shown in Fig. 7.40.

An attacker can reproduce steps 1–5 of the commercial T-BOX attack process to complete message tampering. According to the principle of strategic ruling, the attacker can only successfully tamper with messages on the executor that has system vulnerabilities. However, because the other two online executors do not have the same vulnerabilities, the outputs of the three online executors have a differential-mode expression. Inconsistent messages are discovered and blocked in the ruling process, and the CAN message for the door-opening operation could not be sent down to the vehicle body. The process of blocking message tampering based on strategic ruling is shown in Fig. 7.41.

As shown in Fig. 7.42, we can see on the TSP cloud the whole process that after the mimic T-BOX is attacked, executor 1 goes offline, and executor 2 is selected to go online according to the scheduling strategy.

Experiments show that even if a 0-day differential-mode vulnerability or backdoor appears in a stable version of a Linux system some day in the future, the



Fig. 7.40 The status of the mimic T-BOX executor displayed on the TSP cloud page

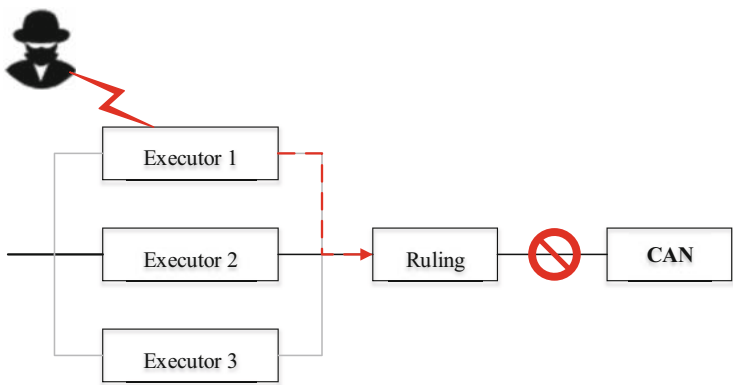


Fig. 7.41 The process of blocking message tampering based on strategic ruling

endogenously safe and secure T-BOX can still achieve effective security defense and protect intelligent networked vehicles from cyberattacks without relying on prior knowledge and add-on security technologies.



Fig. 7.42 The online and offline status of the mimic T-BOX executor displayed on the TSP cloud page

It should be emphasized that because virtual containers use the same operating system and the operating environment provided by the host processor, there is a theoretical possibility of common-mode escape. Therefore, the heterogeneous resource pool should include heterogeneous container resources, and a combination of heterogeneous container resources should be used, if possible, when configuring the executors in the operating environment.

7.5 ESS-Empowered Industrial Control Systems

7.5.1 Threat Analysis

Industrial control systems (ICS) are connected to instruments and networks to control and automate the operation of industrial processes [17]. The ICS support large-scale industrial production, such as power generation, energy production, transportation, manufacturing, etc., thus forming a national critical information

infrastructure [18]. In the fourth industrial revolution, the reliable operation of modern ICS has brought huge social, economic, and environmental benefits.

As shown in Fig. 7.43, internal and external networks of modern ICS are connected to the industrial control network. This architecture brings huge advantages such as diversified products, improved production monitoring efficiency, and reduced management costs. However, since the monitoring layer, control layer, and physical layer are all connected to the network, all aspects of the entire industrial control system are exposed to an unreliable network environment. Figure 7.43 simplifies the interconnection and isolation between the network and the industrial control module. The interconnection between the external network and the industrial control network can be achieved in a variety of ways, such as through firewalls, routers, switches, and other network devices or through host devices with multiple Ethernet ports. The isolation between the two can also be realized in a variety of ways, which may include more advanced three-layer firewalls, network address translation, etc.

The monitoring layer is the uppermost layer of the industrial control system and is primarily responsible for displaying information about the industrial process to the operator. The monitoring layer usually consists of a human-machine interface (HMI), a storage cloud, a computing cloud, etc. The HMI is a user interface that provides operators with real-time, visualized information such as flowcharts, alarm messages, etc. The storage cloud is used to save historical data of industrial processes and the data collected by the monitoring layer, while the computing cloud is used to process the data collected by the monitoring layer and provide analysis reports to operators.

The control layer is the core of the industrial control system and is responsible for controlling the industrial process based on the data collected by the monitoring layer. The control layer usually consists of a programmable logic controller (PLC) and a distributed control system (DCS). PLC is a programmable electronic device whose main function is to control the operating status of external devices based on pre-written program logic to achieve automated control. DCS is a more sophisticated control system used to monitor and control large and complex equipment in industrial processes.

The physical layer is the lowest layer of the industrial control system, which consists of sensors, executors, etc. Sensors are used to sense various parameters in industrial processes, such as temperature, pressure, flow, etc. Executors are used to

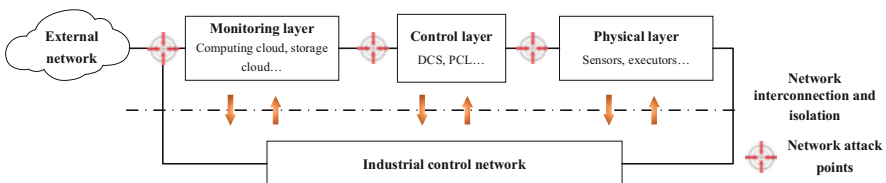


Fig. 7.43 Integration of modern industrial control systems with Internet technology

control various components in industrial processes, such as valves, pumps, motors, and so on.

Compared with the ICS in traditional isolation scenarios running proprietary protocols on dedicated hardware and software platforms (referred to as “traditional ICS”), the integration of ICS and Internet technologies exposes the otherwise isolated ICS to the outside world [19] (referred to as “modern ICS”), making it more vulnerable to various cyberattack threats [20], including but not limited to the following:

1. Malware attacks. Computers and network devices in ICS may be attacked by various versions of malware (such as virus Trojans and worms). The malware can penetrate into the ICS system through the network and damage, tamper with, or steal data from the system.
2. Attacks exploiting embedded system vulnerabilities. The control layer devices in ICS are usually designed based on embedded systems, and these devices have many security vulnerabilities, such as weak passwords, unencrypted communications, etc. Attackers can launch attacks through these vulnerabilities, for example, by controlling PLCs to disrupt industrial processes or tampering with sensor data to interfere with the system.
3. Network isolation attacks. The monitoring layer, control layer, and physical layer in ICS need to communicate with each other, but they also need to be strictly isolated. If the isolation between these layers is not strict, attackers can enter the control layer through the monitoring layer or physical layer and damage the system.
4. Identity authentication and access control attacks. Devices in ICS usually use fixed default accounts and passwords or do not have strict identity authentication and access control mechanisms. Attackers can use these weaknesses to gain administrator privileges and damage, tamper with, or steal data from the system.
5. Social engineering attacks. Attackers can use social engineering means to deceive ICS system users, such as tricking users into opening malicious attachments or links through phishing emails or pretending to be legitimate users to obtain sensitive information such as account passwords through phone calls or emails.

The widespread application of digitization, intelligence, and networking technologies has prompted the functional safety and cybersecurity problems of ICS systems to become increasingly intertwined GFS problems. In addition, the development of cyberattack technologies has completely subverted the original theoretical and technical premise of functional safety affected by natural factors. Cyberattacks on ICS are occurring at an alarming rate, putting public security [21], environmental protection and financial security [22], and other fields at serious risks (as shown in Fig. 7.44), causing immeasurable economic losses and generating serious social impacts.



Attacks on Siemens PLCs paralyzed a nuclear plant.



A power dispatch and control system got out of control due to a cyber-attack.



An oil transmission control system was hijacked.



A water treatment system was tampered with, nearly causing widespread poisoning.

Fig. 7.44 Frequent security incidents in industrial control systems

7.5.2 Design Ideas

Although modern ICS can achieve global control optimization and effectively support the upgrade and transformation of industrial models from automation and digitalization to intelligence and networking, modern ICS also faces serious challenges such as inexhaustible unknown vulnerabilities, backdoors, and trapdoors in software and hardware components, difficulty in thoroughly investigating virus Trojans, increasing network attack surfaces, and rapidly evolving attack techniques. Modern ICS should not only be able to deal with the effects of random and uncertain failures but also have the ability to resist any potential unknown cyberattacks [23]. This requires proposing new theoretical approaches and inventing new technical architectures to solve the GFS problems in ICS. Undoubtedly, the proposal of the ESS theory and the invention of the DHR empowerment architecture have opened up a new way to crack the intertwined problems of functional safety and cybersecurity nowadays.

According to the analysis in Sect. 7.5.1, the threats to modern ICS can be divided into two major categories: one is the type of threats to the traditional ICS ontology, that is, intelligent industrial controllers, and the other is the type of threats to the convergence or intersection of IT and ICS. Therefore, if the ESS-empowered industrial control system wants to meet the integrated GFS requirements, it can be designed and implemented according to the following two ideas:

7.5.2.1 Intelligent Controller with the ESS Structure

Intelligent industrial controllers such as PLC and DCS are diverse and heterogeneous, the localization rate of core software/hardware components is low, and the mechanism of the vulnerabilities in chips, operating systems, application software, and edge intelligent control platforms is complex and scattered, making it difficult to completely detect and eradicate the vulnerabilities and backdoors and defend against the unknown attacks and functional failures that are caused by these vulnerabilities. However, the traditional cybersecurity static defense technologies are based on the traditional information domain attack mechanism for rule feature matching and perception defense. In industrial control scenarios, there are problems such as insufficient attack perception accuracy and defense effectiveness, difficulty in adapting to the resource constraints of intelligent industrial controllers and the heterogeneity of platforms, and inability to realize full compatibility between new and old devices and between domestic and non-domestic devices. Therefore, it is necessary to design an ESS structure that has no interference with and low impacts on the control loop of the industrial control system and ensure the highly reliable implementation of ESS measures. At the same time, it is necessary to accurately and effectively quantify the effects of attack perception and fault prevention under the condition that the characteristics of unknown attacks, vulnerabilities, and backdoors are not obvious and the mechanism of fault propagation is complex, so as to ensure the credibility of the effect of defense against unknown attacks.

7.5.2.2 Intelligent Industrial Control Systems with Proactive Protection

On the one hand, in cloud-side collaborative industrial control scenarios, the flat mixed connection and ubiquitous access of cloud-edge-end devices lead to an increase in the attack surface; the mixed connection of public and private networks, the blurring of security boundaries, and the serious overlapping of information and physical domains lead to random changes of the attack paths; and the unknown characteristics of covert conduction and attack across the information and physical domains make it extremely difficult to accurately identify and avoid cyberattacks in advance. Therefore, it is necessary to explore mechanisms of collaborative attacks on industrial control systems and the complex and diverse covert attack modes across information and physical domains and, on this basis, to propose collaborative evasion methods for different attack paths so that collaborative attacks across information and physical domains can be perceived in advance.

On the other hand, in cloud-edge collaboration scenarios, there is an increasing demand for process updates, continuous iteration of industrial control programs, configuration and development, and dynamic evolution of industrial control logic; traditional firewalls, patches, and other static defense measures are no longer effective enough; traditional cybersecurity defense technologies need to be combined with proactive defense technologies; and collaborative, evolutionary, adaptive,

endogenously safe and secure defense technologies need to be explored that are applicable to the control of all interactive processes in the whole life cycle of industrial control systems.

7.5.3 *System Architecture Design*

Following the above design ideas, the endogenously safe and secure ICS system architecture is shown in Fig. 7.45. The system can be logically divided into a device module layer, a transport layer, and a platform layer. Among them, the device module layer contains PLC nodes with multicore isolation and full-stack heterogeneity and DCS nodes with threat perception and quantification capabilities to realize ESS protection at the level of device module. The transport layer receives threat warnings from the upper platform and the lower device module through vertical collaboration and sends isolation and disposal commands to the upstream and downstream through dynamic horizontal isolation strategies. The platform layer, in response to the problem of expanded attack surface caused by cloud deployment of industrial control systems, forms differential-mode containers or virtual machines to mimic computing resources and storage resources and carry out hierarchical scheduling and cross-domain collaboration.

7.5.3.1 **Industrial Control Devices with Multicore Isolation and Full-Stack Heterogeneity**

The existing industrial control devices such as PLCs are transformed and upgraded, and technical means such as multicore isolation, multilevel heterogeneity, distribution ruling, and error correction output are used to block the propagation of the attack chain, improve the overall redundancy and security capability of industrial control devices, and realize their ontological security, as shown in Fig. 7.46.

1. Mimic architecture MCU chips. These chips realize the industrial program executors of mainstream microcontroller cores such as ARM and RISC-V and improve single-device chip-level controllability and measurability through key technologies such as shared resource analysis of logic functions of the mimic MCU chip prototype, operand isolation, and bus inversion coding.
2. Real-time distribution ruling at the command level. According to the engineering program commands issued by the heterogeneous microcontroller kernel, the mimetic scheduling, ruling, and situational awareness of heterogeneous multicore CPUs are implemented, the heterogeneous kernel's output of error correction for chip peripheral instructions or messages is controlled, and fast recovery at the CPU core transfer transaction level is accomplished.
3. Cleansing and recovery of heterogeneous executors. For heterogeneous CPU cores, program isolation between processors, operation synchronization, key

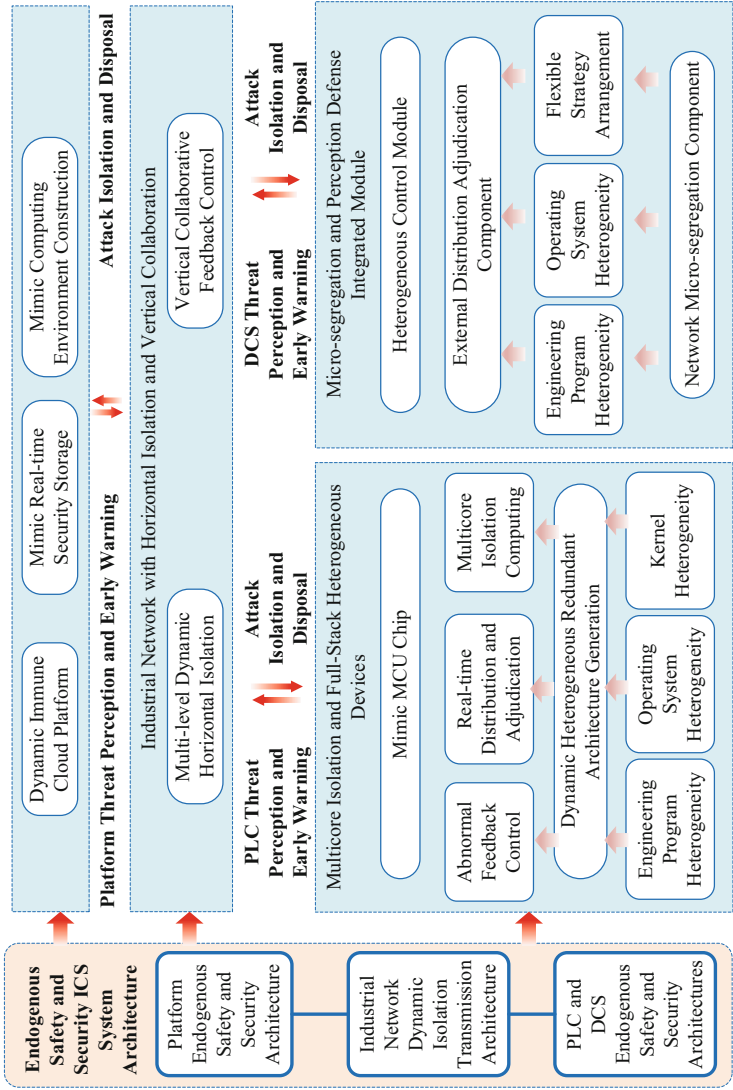


Fig. 7.45 System architecture of endogenously safe and secure ICS

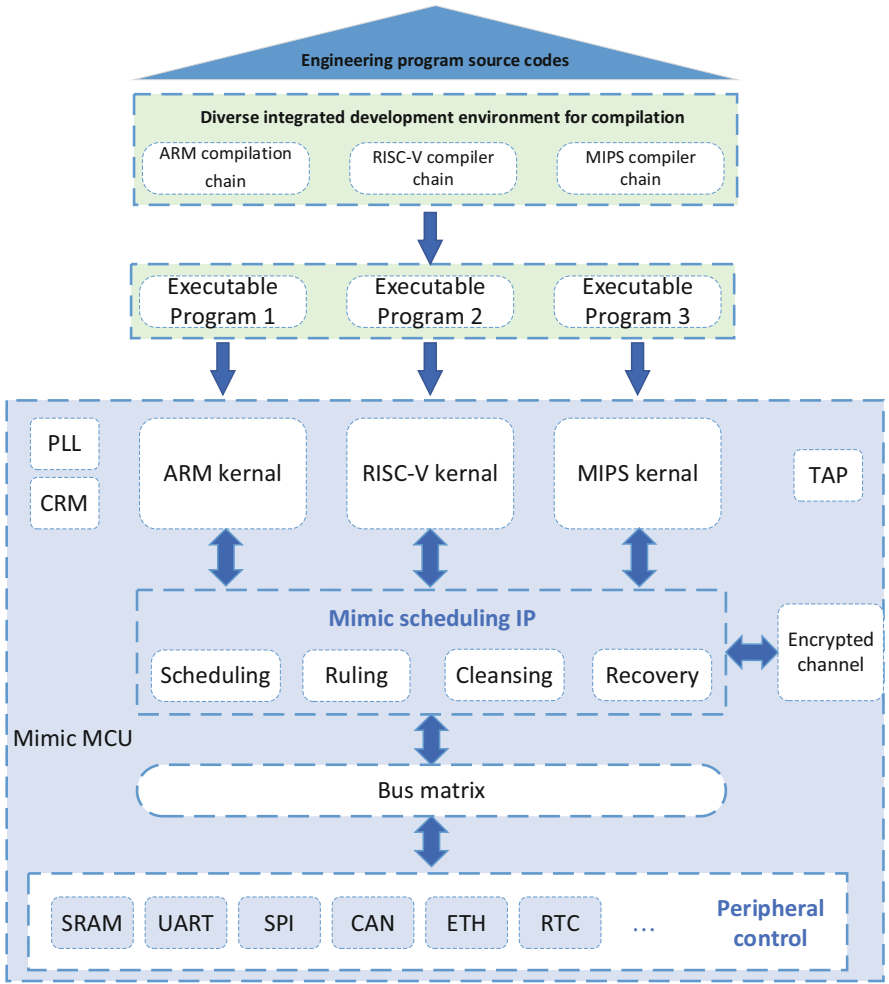


Fig. 7.46 Industrial control device with multicore isolation and full-stack heterogeneity

context storage and backup, and fault cleansing function under abnormal state are achieved through the checkpoint rollback scheme, the instruction set conversion scheme, etc.

7.5.3.2 Industrial Control Module Integrating Micro-Isolation, Perception, and Defense

1. Module-level ESS structure. I/O executors based on heterogeneous MCUs and micro-operating systems, as well as computing executors based on heterogeneous

CPUs and real-time operating systems are constructed to form a multi-objective defense chain architecture and solve the problem of random functional failures of complex industrial control modules. Techniques such as quantitative analysis and strategic voting are used to reduce the probability of hidden and apparent faults, in addition to the design of dual heterogeneous redundant I/O modules, quadruple heterogeneous redundant architecture modules, and so on.

2. Multimode ruling of industrial control modules. By utilizing the flexible wiring and high-integration characteristics of field-programmable gate arrays and formulating precision tuning strategies, abnormal data and instructions can be blocked and safely filtered, and the problem of biased ruling over the consistency of bidirectional conversion between digital and analog quantities can be solved.
3. Clock synchronization for working condition information perception. The synchronization and delay module of the stagewise output execution cycle of the mimic security control surface and the industrial control service surface is reset to solve the problem of process hanging caused by a single-point blocking and ensure the correctness and integrity of the process pipeline and the timeliness and robustness of the industrial control task execution, ruling, response, and feedback control within the specified logical operation cycle.

7.5.3.3 Industrial Network with Horizontal Isolation and Vertical Collaboration

1. Cloud-edge collaborative network isolation. The principle of least privilege is applied to implement process-level micro-isolation of strategic control, detect and prevent lateral movement of threats, reduce the available attack surface, and achieve access control of lateral network traffic. System operation information is analyzed based on the industrial control cloud platform; traffic and host network are isolated and protected based on intelligent analysis.
2. Network isolation modeling and dynamic strategic tuning. This function is used to model host network behavior, construct host network behavior profiles including the source IP, destination IP, communication ports, traffic, communication period, and other metadata; track and control the active and passive network traffic behavior of management and control instances; dynamically optimize network instance isolation boundaries; and manage instance networks of industrial control devices, modules, containers, and so on.
3. Dynamic authentication and access control based on feature analysis. This function is used to analyze the characteristics traffic monitoring, flow tracking, access policies, etc., dynamically adjust horizontal isolation strategies, dynamically identify object access rights, and implement access blocking and collaborative feedback for abnormal instances to achieve fine-grained authorization of the cloud-edge collaborative control platform.

7.5.3.4 Dynamically Reconfigurable Edge Intelligent Control Platform

Constructing a dynamically reconfigurable edge intelligent control platform (as shown in Fig. 7.47) enables the ICS system to have ESS capabilities such as vulnerability tolerance, attack avoidance, and failure suppression.

1. Cloud-edge collaborative cross-domain linkage control. This function is used to build a closed loop of point-to-surface ESS architecture and cloud-edge collaboration cross-domain linkage control, which supports cross-domain threat alarm collection and security strategy issuance; ensures the safe operation of industrial control systems in toxic and open environments; realizes threat discovery, attack avoidance, and fault suppression; and solves the problem of cascading faults caused by single-point failure of cloud-edge collaboration.
2. Applying random heterogeneity mapping. This function is used to set up heterogeneous containers or virtual machine executors and implement hierarchical scheduling and cross-domain collaboration strategies based on heterogeneous metrics, load balancing, timing control, and other mixed strategies. The differential-mode cloud platform and spatiotemporally triggered application deployment can quickly perceive, shield, and locate generalized uncertain threats to cloud platform functions, data, and management and solve the problem of expanded attack surface of cloud deployment of industrial control systems.
3. Mimic distributed storage system. Replicated heterogeneous deployment of metadata such as directories, files, and replicas constitutes a dynamic heterogeneous operating environment with management and identity authentication functions. In a dynamic, heterogeneous, and redundant operating environment, data object tampering and unauthorized access will trigger the synchronization of trusted source metadata, realize the perception of storage layer threats, complete the cleansing and recovery of the storage system, and solve the problem of data storage object tampering and unauthorized access.
4. Reconfigurable mimic secure computing environment. Heterogeneous executors are constructed that cover CPU, operating systems, middleware, and applications. Application of consistency ruling of I/O data flow to the fields, values, texts, pictures, and other types of security consensus forms the basis and path of backward control such as response normalization, threat perception and positioning, and dynamic reconfiguration and realizes ESS application integration and microservice deployment.

7.5.4 Design of Functional Units

Based on the endogenously safe and secure ICS system architecture, this subsection introduces the design of some key functional units in the endogenously safe and secure ICS system, mainly including security support devices, safe PLC, and safe DCS.

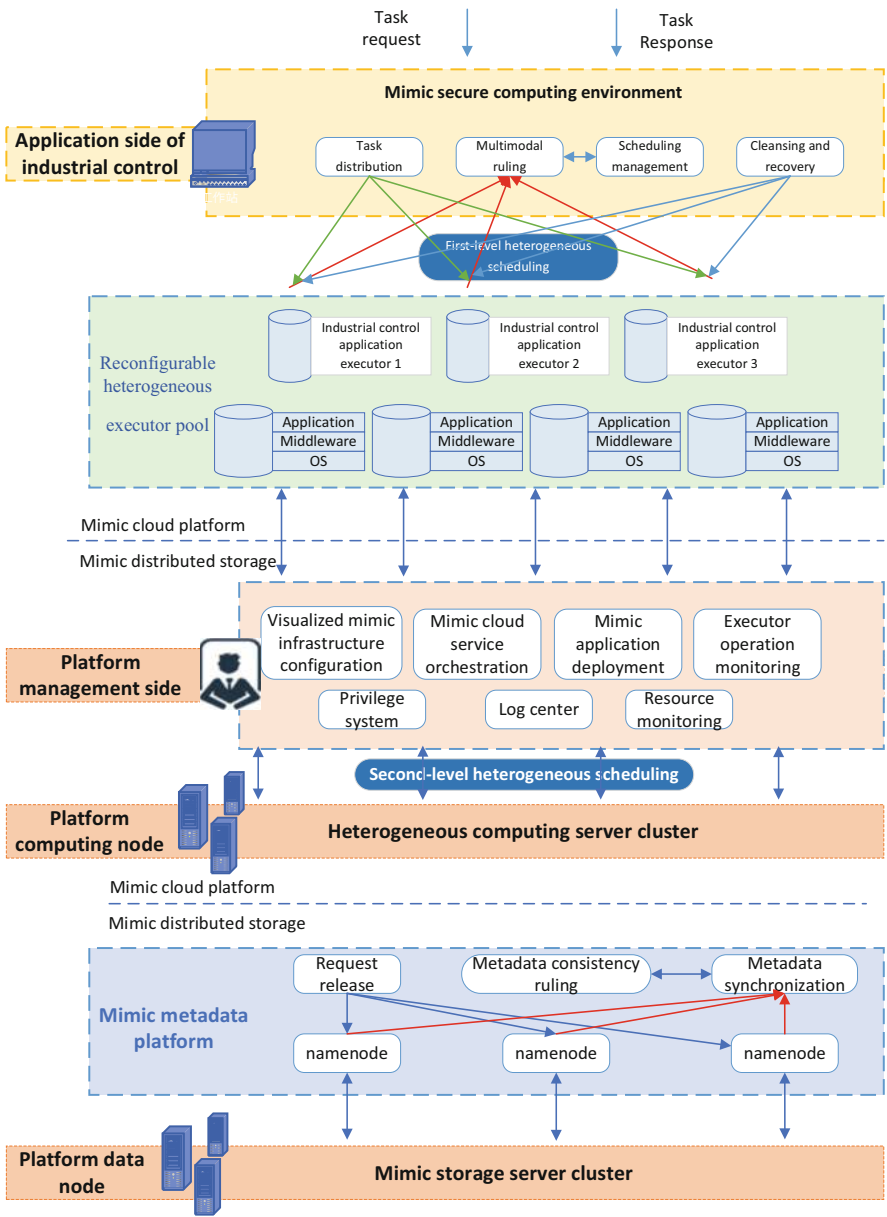


Fig. 7.47 Dynamically reconfigurable edge intelligent control platform

7.5.4.1 Security Support Devices

Security support tools, chips, and modules provide endogenously safe and secure components and technical support for ICS systems (as shown in Fig. 7.48).

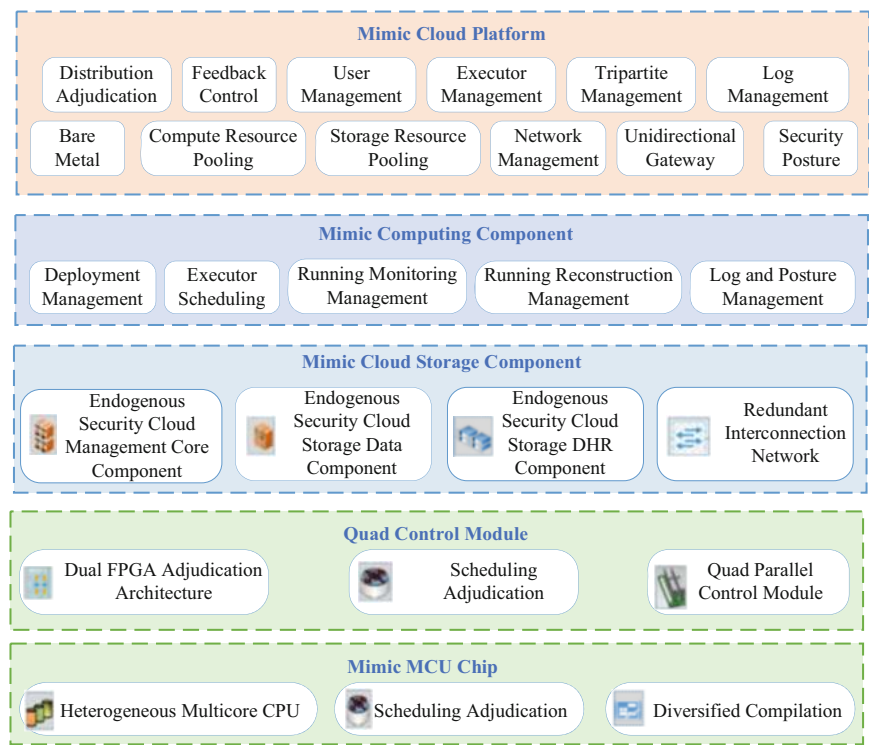


Fig. 7.48 Security support tools, chips, and modules

1. Mimic MCU Chip

MCU is the “brain” of many kinds of industrial equipment, and its performance, stability, and reliability are crucial. MCU products currently on the market are basically single-core or isomorphic multicore and have “genetic” defects in terms of security against network attacks. In view of the security challenges posed by unknown vulnerabilities and backdoors in industrial control nodes, a heterogeneous multicore fully compiled security MCU chip dedicated to PLC nodes is developed to support the scheduling, ruling, cleansing, recovery, error correction, and other operations of the MCU’s internal executors, providing a comprehensive foundation for realizing highly reliable, highly trustworthy, and highly available programmable logic controllers (PLCs) that have generalized functional safety (GFS).

2. Quadruple Control Module

A two-out-of-four collaborative voting dual-FPGA architecture can be designed. The two FPGAs are placed on both sides of the backplane base of the heterogeneous control module to receive its operation results and exchange

information and make a vote based on built-in strategies in support of the safe distributed controller of industrial control systems.

3. Cloud Resource Components

Mimic cloud storage components should at least include heterogeneous meta-data storage components and attack voting defense components, provide a dynamic heterogeneous operating environment for storage resources, and support threat perception and handling of storage resources. The mimic computing component integrates management deployment, executor scheduling, operation monitoring management, operation reconfiguration management, and log and situation management. The new ESS-oriented cloud platform supports multiple cross-domain scheduling functions such as distribution ruling, feedback control, user management, and executor management.

7.5.4.2 PLC Security Evolution

The sequential idea of “device-equipment-system” is adopted to build a safe PLC device with adaptive defense mechanism, as shown in Fig. 7.49. Mimic MCU chips and their compilation platform are adapted, and a secure environment for PLC firmware codes, real-time data, control operation, and communication interconnection is constructed according to the working principle of PLC and its programming

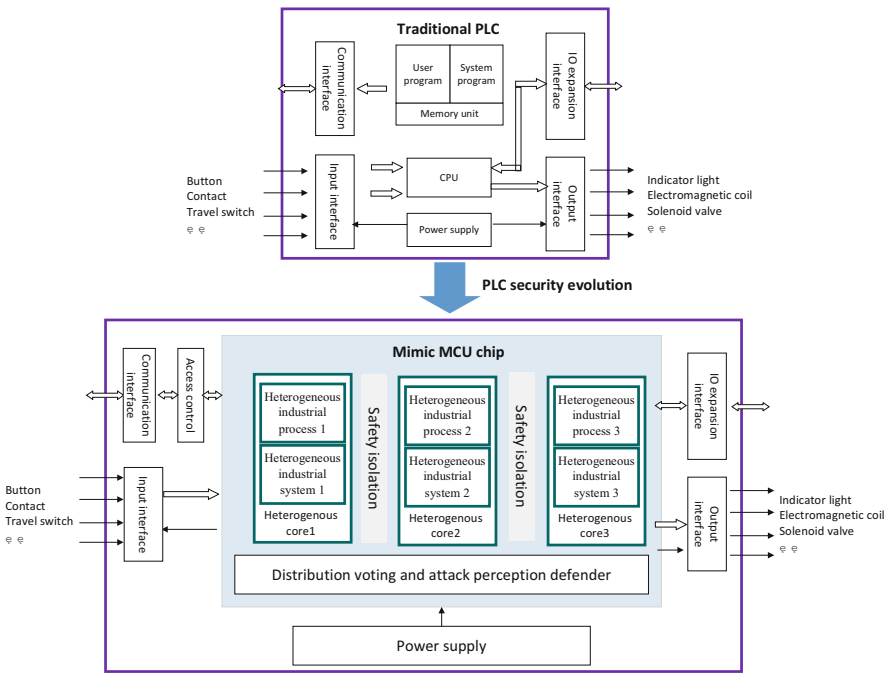


Fig. 7.49 PLC security evolution

language. It is also required to develop instruction-level redundancy voting modules, heterogeneous executor scheduling modules, abnormal executor cleansing modules, randomized feature compilation algorithm modules, innovative and diversified compilation design modules, and equivalent heterogeneous program coding modules with PLC software-hardware collaboration.

7.5.4.3 Secure DCS

As shown in Fig. 7.50, an operating environment with the endogenously safe and secure mechanism is provided for DCS according to the working principle of DCS, on the basis of compatible passive protection capabilities such as identity authentication supporting the fingerprint technology, communication encryption based on commercial passwords and data integrity check, configuring dual redundancy at the input and output ends, and configuring quadruple redundancy at the control end and by means of configuration operation update, data synchronization, I/O real-time input processing, user program scheduling, I/O real-time output processing, network communication data receiving and processing, fault diagnosis, etc.

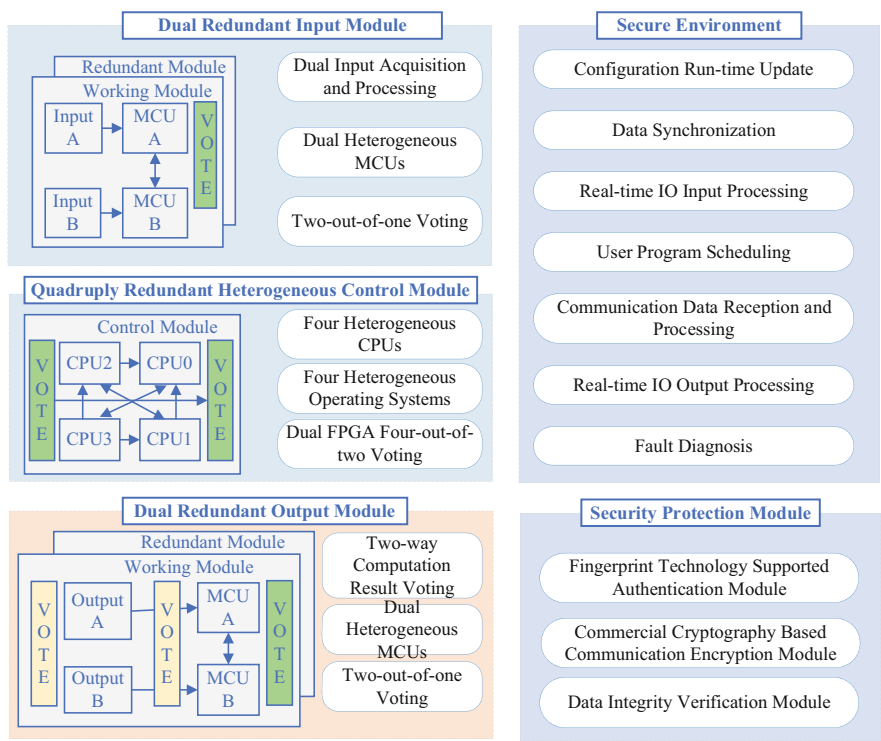


Fig. 7.50 Secure DCS based on a quadruple control module

7.5.5 Safety Practices

7.5.5.1 Edge-End Mimic Gateway

Edge gateway is close to the edge of the network of industrial equipment, sensors, and other physical devices and is the last physical entities for end-side device data to the cloud. How to realize ESS and effectively defend against unknown vulnerabilities and backdoors in edge gateways is of great significance to the realization of “end-edge-cloud” collaborative computing. In the booming development of industrial Internet, industrial edge gateways are not only faced with external threats from the Internet but also intertwined with the internal security problems of industrial production, involving severe and complex security risks.

In response to the security integration application requirements of industrial control systems and through research on the ESS defense technology of edge gateways, a multiplicative DHR architecture model, redundant synchronization mechanism, and strategic voting approach have been established, the technical challenge of dynamic port migration and dynamic switching of protocol stacks for industrial protocols has been overcome, and an endogenously safe and secure mimic edge gateway device [24] has been developed, as shown in Fig. 7.51.

The mimic edge gateway contains three heterogeneous CPU daughter cards, each of which runs a heterogeneous operating system and hosts a mimic industrial protocol stack and diversified compiled APPs. The three daughter cards together constitute the heterogeneous executor resource pool of the mimic system. The strategic ruling and scheduling function is carried by the two FPGAs of Xilinx’s XQVU7P and XC7Z045; the data is distributed to the three heterogeneous CPUs through the strategic arbitrator agent on XQVU7P, multimode ruling is made when the execution results of the three heterogeneous CPUs are synchronized to the



Fig. 7.51 Design view of an industrial Internet mimic edge gateway

strategic arbitrator, and the consistency of the data results is checked to determine whether the heterogeneous executor is under attack; a critical data isolation area is set up on the FPGA of XC7Z045 to save critical historical data of CPU operation; when the executor comes back online, status tracking can be performed from this area to return to the normal working condition.

Integrated on the FPGA of XQVU7P are hardware functional units, which include a Layer-2 point switching module to implement configurable flow table forwarding, a hardware real-time encryption module to ensure the secure transmission of cloud data, a software-definable protocol conversion module to achieve deterministic cross-network protocol conversion and transmission, and various interface controllers. The FPGA of XC7Z045 reserves a service configuration management module and service upgrade debugging management interface. The mimic edge gateway device can not only realize service upgrade and configuration remotely through the network but also carry out safer service deployment and application upgrade through the peripheral interface of XC7Z045 in the field.

The mimic edge gateway supports the identification and conversion of 36 industrial protocols, global clock synchronization, hardware real-time encryption, audio/video acceleration, and other functions to meet the application needs in industrial fields. It also has ESS to defend against APT attacks.

Mimic edge gateways have been intensively applied in 22 enterprises in petrochemical, electric power, energy, and communication industries on a pilot basis, with an application scope covering multiple links such as production data collection and production environment monitoring in the safe production chain of the industrial Internet, providing effective solutions for protocol conversion interoperability, network security protection, and other weak links. The on-site application of the product by these enterprises showed that the product ran stably and effectively during the trial period, indicating that industrial Internet mimic edge gateway equipment has good market promotion value.

The industrial Internet mimic edge gateway participated in the “Zhijiang Cup-Industrial Internet Endogenous Safety and Security International Elite Challenge” held on December 20, 2020. At the event, 40 “white hat hacker” teams from the United States, Germany, Russia, Japan, South Korea, and China launched 950,000 high-intensity attacks on the core mimic-structured devices of industrial Internet developed by Zhijiang Laboratory, but none of the white hat hackers or teams succeeded. This result verifies the universality and effectiveness of the ESS theory and methods in the field of industrial control networks.

7.5.5.2 Edge Intelligent Control Platform

The development of the edge intelligent control platform is based on the prior development of security protection components and automated response components. The security protection component has the capabilities of rapid authentication of security domains, hierarchical dynamic encryption of programs and data, process security monitoring, multi-domain distributed redundant storage, and asynchronous recovery of engineering data and is responsible for the development, deployment, operation, and update of the intelligent control platform.

Expert knowledge can be used to analyze the influence of security incidents on coordination and ruling of technologies, processes, and personnel and build a security protection algorithm library and knowledge base. Based on the orchestration of software-defined security protection components, the pattern of security incidents in industrial control is identified, the response strategy is automatically selected for the known events in the library, and the response strategy is automatically recommended for unknown events. According to the decision results, the security components are automatically linked to protect against and block threats, and the algorithm base and knowledge base are updated to complete the automatic response to security incidents. The automatic response component is also responsible for the hierarchical management and disposal of security incident alarms, making operational process metrics recordable, measurable, and traceable.

The edge intelligent control platform is compatible with functions such as logic control and process control of multiple programming languages such as FBD, LD, SFC, and ST that support the IEC61131-3 standard (as shown in Fig. 7.52). It also has security protection functions such as security control of configuration authority, quick identity authentication of security domain, hierarchical dynamic encryption of engineering program and real-time data, multi-domain distributed redundant storage of engineering data, and asynchronous recovery, thereby ensuring that the edge intelligent control platform has available field data collection, reliable historical data records, controllable access privilege processing, manageable alarm and incident records, complete security logs, trustworthy process control, accurate animation display, and true report output.

7.6 Summary of This Chapter

First, this chapter briefly introduces the concept of ESS construction engineering application baseline, including initial application baseline, engineering iterative analysis, and multi-angle evaluation improvement. Then, typical IT, ICT, ICS, or CPS systems such as the network communication control system, the cloud computing service system, the IoV system, and the industrial control system are selected, and the current engineering application of ESS-empowered cyber resilience in many fields is introduced from the aspects of threat analysis, the overall ESS architecture, heterogeneous executor construction, strategic ruling design, and specific practical cases. The multi-field, multi-scenario, and diversified engineering application practices have fully verified the effectiveness and vitality of ESS-empowered cyber resilience engineering. At the same time, it is also pointed out that although the ESS enabling architecture has universal application significance, different types of application systems still need to be individually designed and created according to the characteristics of their respective fields. At the same time, it is pointed out that although the ESS-empowered architecture has universal application significance, different types of application systems still need to be individually designed and created according to the specific characteristics of their respective fields.

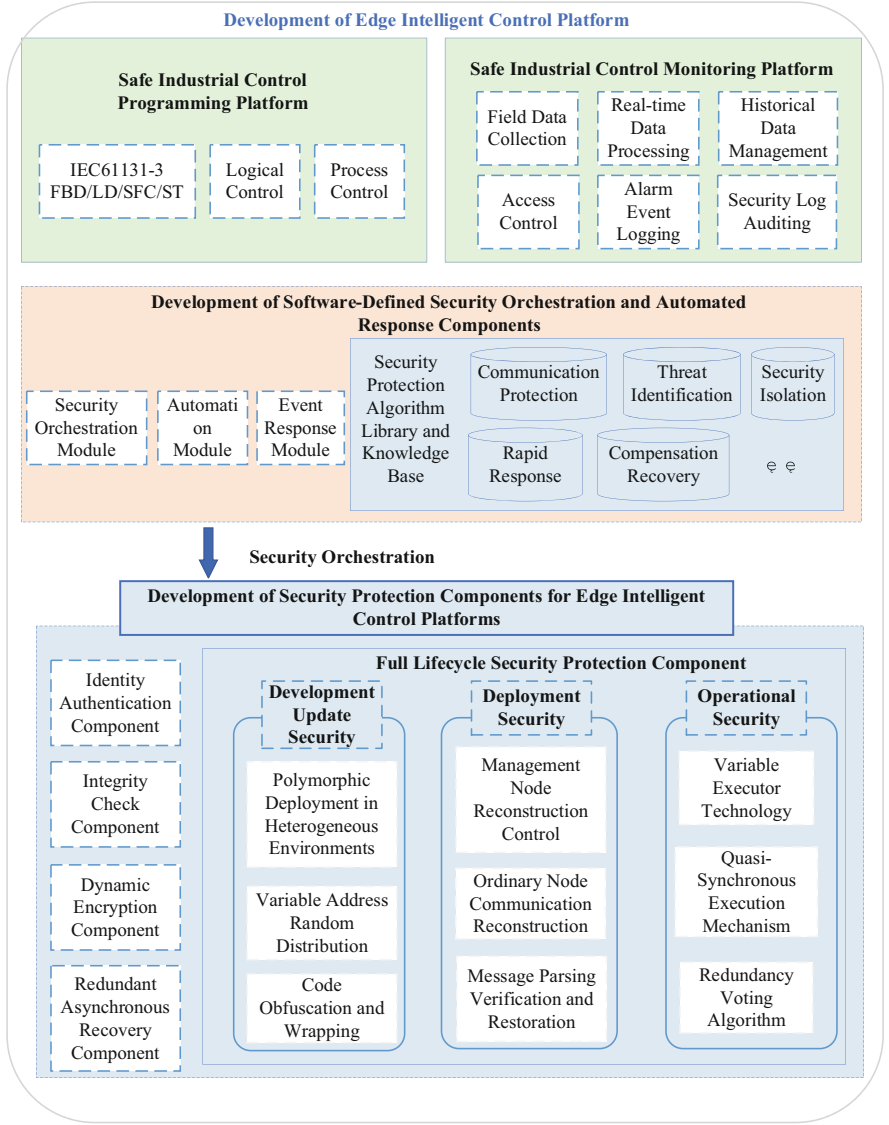


Fig. 7.52 Edge intelligent control platform

References

1. Hailong M, Peng Y, Yiming J, et al. Dynamic heterogeneous redundancy based router architecture with mimic defenses [J]. J Cyber Secur. 2017;2(1):29–42.

2. Hailong M, Yiming J, Bing B, et al. Tests and analyses for mimic defense ability of routers [J]. J Inf Secur. 2017;2(1):43–53.

3. Xu L, Huang J, Hong S, et al. Attacking the brain: races in the SDN control plane [C]. Paper presented at: 26th USENIX Security Symposium (USENIX Security 17); 2017. p. 451–68.
4. Ujcich BE, Okhravi H, Jero S, et al. Cross-app poisoning in software-defined networking [C]. Paper presented at: The 2018 ACM SIGSAC Conference ACM; 2018. p. 648–63.
5. Mousavi SM, St-Hilaire M. Early detection of DDoS attacks against SDN controllers [C]. Paper presented at: 2015 International Conference on Computing, Networking and Communications (ICNC); 2015. p. 77–81.
6. Cloud Native Industry Alliance. Cloud native development white paper [R]. 2020.
7. Gao X, Steenkamer B, Gu Z, et al. A study on the security implications of information leakages in container clouds [J]. *IEEE Trans Dependable Secur Comput*. 2021;18(1):174–91.
8. Monteiro L, Hazin R, Lima A, et al. Survey on microservice architecture-security, privacy and standardization on cloud computing environment [C]. Paper presented at: Proceedings of 12th International Conference on Software Engineering Advance, Athenas, Greek; 2017. p. 1–7.
9. Adam G. Introducing domain-oriented microservice architecture [EB/OL]. 2022. <https://eng.uber.com/microservice-architecture>
10. Lin X, Lei L, Wang Y, et al. A measurement study on Linux container security: attacks and countermeasures [C]. Paper presented at: Proceedings of 34th Annual Computer Security Applications Conference, New York, USA, ACM; 2018. p. 1–15.
11. Agache A, Marc B, Andreea F, et al. Firecracker: light-weight virtualization for serverless applications [C]. Paper presented at: Proceedings of 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI). Santa Clara, USA, USENIX; 2020. p. 419–34.
12. Avrahami Y. CVE-2020-2023 [EB/OL]. 2020. <https://github.com/kata-containers/community/blob/master/VMT/KCSA/KCSA-CVE-2020-2023.md>
13. Andreas B. What is Log4Shell? [EB/OL]. 2021. <https://www.dynatrace.com/news/blog/what-is-log4shell/>
14. Lin C, Khazari H. Modeling and optimization of performance and cost of serverless applications [J]. *IEEE Trans Parallel Distrib Syst*. 2020;32(3):615–32.
15. Henrique Z, Guilherme P, Mauricio A, et al. Run C and Kata runtime using Docker: a network perspective comparison [C]. Paper presented at: Proceedings of Latin-American Conference on Communications (LATINCOM), Santo Domingo, Dominican Republic; 2021. p. 1–6.
16. Cloud Native Computing Foundation. Cloud native landscape [EB/OL]. 2022. <https://landscape.cncf.io>
17. Ding D, Han QL, Xiang Y, et al. A survey on security control and attack detection for industrial cyber-physical systems [J]. *Neurocomputing*. 2018;275:1674–83.
18. McLaughlin S, Konstantinou C, Wang X, et al. The cybersecurity landscape in industrial control systems [J]. *Proc IEEE*. 2016;104(5):1039–57.
19. Zhou C, Hu B, Shi Y, et al. A unified architectural approach for cyberattack-resilient industrial control systems [J]. *Proc IEEE*. 2020;109(4):517–41.
20. Rubio JE, Alcaraz C, Roman R, et al. Current cyber-defense trends in industrial control systems [J]. *Comput Secur*. 2019;87:101561.
21. Cervini J, Rubin A, Watkins L. Don't drink the cyber: extrapolating the possibilities of Oldsmar's water treatment cyberattack [C]. *Int Conf Cyber Warf Secur*. 2022;17(1):19–25.
22. Abrams L. Computer giant acer hit by \$50 million ransomware attack [EB/OL]. 2021. <https://www.bleepingcomputer.com/news/security/computer-giant-acer-hit-by-50-million-ransomware-attack/>
23. Shameli-Sendi A, Aghababaei-Barzegar R, Cheriet M. Taxonomy of information security risk assessment (ISRA) [J]. *Comput Secur*. 2016;57(3):14–30.
24. Zhenping Z, Yansong W, Shunbin L. A mimic industrial edge gateway and mimic processing method [P]. China, CN202010389778.1.2020-05-11.

Chapter 8

Exploration of Emerging Fields Empowered by Endogenous Security and Safety



8.1 Endogenously Safe and Secure Wireless Communication

8.1.1 *Development Paradigms of Wireless Communication*

The development paradigms of wireless communication (on which there is no consensus yet) are natural laws, theoretical basis, and practical norms related to the evolution of wireless communication technologies, which, like the development paradigms of science in general, develop in a step-by-step manner.

Wireless communication is a process of transmitting information through electromagnetic waves in the wireless environment, and it is also a process of countering generalized uncertain disturbances that are inseparable from the wireless environment. Generalized uncertain disturbances in wireless communication refer to unexpected events caused by direct or indirect use of the uncertainty and uncontrollability of the wireless environment, including disturbances caused by natural factors and human factors. As shown in Fig. 8.1, natural disturbance factors include terrain, landforms, surface features, weather, fading, propagation media, electromagnetic dispersion, etc. Man-made disturbance factors include unintentional interference, intentional interference, access attacks, etc.

From the perspective of the cognitive process of the interaction between electromagnetic waves and the wireless environment, the development process of wireless communications is a history of confronting generalized uncertain disturbances. From the perspectives of the world outlook and methodologies, the development of wireless communication involves four paradigms: discovering and using electromagnetic waves, passively adapting to wireless environments, actively using wireless environments, and transforming and customizing wireless environments. Below we briefly describe the four existing paradigms of wireless communication development, summarize the characteristics of these paradigms and practice norms, and discuss the development and dilemma of wireless communication.

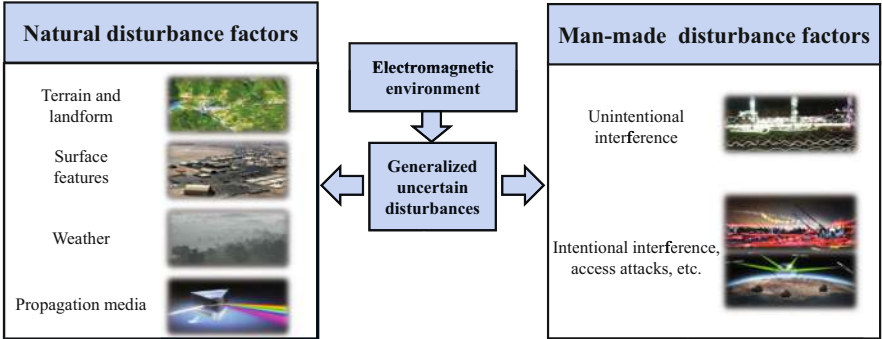


Fig. 8.1 Generalized uncertain disturbances in wireless communication

8.1.1.1 Wireless Communication Development Paradigm 1: Discovering and Using Electromagnetic Waves

In 1873, Maxwell put forward the propagation mechanism of electromagnetic waves in *Treatise on Electricity and Magnetism*. In 1887, Hertz experimentally verified the existence of electromagnetic waves, thus creating conditions for the emergence of radio communication and laying the foundation for Marconi’s first transoceanic wireless telegraph communication using electromagnetic waves in 1897. This formed the initial development paradigm of wireless communication. The significance of this paradigm is that on the basis of the work of Maxwell and Hertz, the scientific law of the existence and propagation of electromagnetic waves was discovered for the first time and that human beings can get rid of the shackles of wired communication and open a new era of radio communication. The methodology corresponding to this paradigm is how to use electromagnetic waves to carry information to create long-distance wireless communication from the scratch, the iconic technology of which is the wireless telegraph. The initial development paradigm of wireless communication only uses the methods of increasing transmission power, simple coding (such as Morse code), and repeated transmission to counter natural disturbances, without recognizing human disturbances and security problems in complex wireless environments.

8.1.1.2 Wireless Communication Development Paradigm 2: Passively Adapting to the Wireless Environment

With the development of wireless communication, generalized uncertain disturbances have gradually become the main challenge. Before Shannon put forward the information theory, it was generally believed that the error probability of fixed-rate information transmission was non-negligible. Shannon theoretically proved that when the communication rate is lower than the channel capacity, a coding method can always be found to transmit information with any low error rate [1], that is, the

maximum error-free transmission rate on a specific channel is given, which provides an upper bound for effective information transmission and lays the foundation for digital communication. Therefore, the idea contained in the second paradigm is represented by Shannon's information theory and channel capacity. It illustrates the inherent relationship between the disturbances caused by the wireless environment and the quality of wireless communication, that is, in order to transmit information reliably on unreliable channels, there must be matching modes of transmission. To achieve the above purpose, the methodology of this paradigm focuses on how to adapt to complex wireless environments and find ways to approach the limits of wireless communication capabilities.

In terms of resisting natural disturbances, this paradigm has mainly gone through two stages: analog communication and digital communication. By means of coding, modulation, filtering, and waveform design, the signal is converted into a form suitable for transmission on the channel, thereby improving the reliability of information transmission. Specifically, the precoding technology designs the signal at the transmitting end so that the transmitted signal matches the channel, thus minimizing inter-symbol interference, improving transmission performance, and reducing the processing complexity of the receiver. At the receiving end, the equalization technology equalizes the signal that has undergone channel distortion, and filters are used to correct and compensate for distorted pulses to reduce the impact of inter-code interference. The adaptive modulation and coding technology determines the channel capacity according to the channel conditions fed back by the terminal, thereby adaptively adjusting the modulation method and coding rate so that data transmission can adapt to channel changes and achieve a higher communication rate.

In terms of countering human disturbances, the initial means was the division and management of wireless public resources such as time and frequencies. Countries around the world scientifically and rationally planned wireless public resources such as time and frequencies, divided special frequency bands for various radio services according to the characteristics of different application fields or scenarios, and set up relevant departments for strict supervision and management [2]. Later, technologies like frequency division multiple access (FDMA), time division multiple access (TDMA), and code division multiple access (CDMA) were applied in mobile communication, and interferences between users were resisted by allocating orthogonal frequencies, time, and code block resources to different users. However, radio spectrum and other resources are limited natural resources, and wireless communication is increasingly dependent on spectrum. Therefore, the industry is constantly developing new frequency resources, such as millimeter wave and terahertz communications [3], to solve the problem of increasingly tight spectrum resources and capacity limitations of current wireless systems. In addition, in order to improve spectrum utilization, the cognitive radio technology has received widespread attention from the academic community in recent years. It realizes dynamic spectrum allocation and spectrum sharing by sensing available spectrum from the environment and adaptively changing communication parameters.

In terms of defending against wireless security threats, this paradigm adopts some iconic technologies such as spread spectrum, encryption, authentication, and

integrity protection. Taking mobile communication as an example, the first-generation analog communication system (1G) basically did not adopt any security protection mechanism, but the second-generation digital communication system (2G) began to add security protection means such as air interface information encryption, identity authentication and identity identification codes, and constantly changed and evolved. For example, in order to make up for the shortcomings of encryption algorithms, mobile communication encryption algorithms have evolved from 2G A51/A52 and 3G KASUMI to 4G and 5G advanced encryption standards (AES), SNOW3G, the ZUC algorithm, and other group key or stream key encryption algorithms with higher encryption strength [4], and the key length has increased from 64 bits in 2G to 256 bits in 5G. In order to solve the problem of pseudo base stations caused by the one-way authentication mechanism in 2G, a two-way authentication mechanism was introduced at the beginning of the 3G era, which can simultaneously authenticate the legitimacy of users and mobile communication networks. In order to cope with the identification, positioning, and tracking of the subscription permanent identifier (SUPI) transmitted in plaintext intercepted by unscrupulous users, 5G utilizes public key encryption methods based on the public key infrastructure (PKI) mechanism to encrypt the SUPI and then transmit it [5].

This paradigm improves the ability to deal with wireless disturbances, but it has the following inherent deficiencies: ① the ability to adapt to the wireless environment is limited by the ability to perceive the wireless environment, and it lacks the “refined” perception of the “differentiated” wireless environment; ② due to problems such as limited spectrum resources and interference between users, improving spectrum utilization and system capacity through dynamic spectrum allocation and spectrum sharing will be restricted by the “air-spectrum wall”; ③ time/frequency/code domain resources in wireless communications are public resources, which are irreconcilable in nature, unable to fundamentally block interference and eliminate security threats, and lack the development and utilization of “personalized” resources in the wireless environment; ④ patching, plug-in security technologies, and other security means based on cryptography lack the endogenous safety and security (ESS) mechanism to explore the integration of security and communication from the original attributes shared by security and communication.

8.1.1.3 Wireless Communication Development Paradigm 3: Actively Using the Wireless Environment

With the continuous expansion of communication services, the way of passively adapting to the public resources in time/frequency/code domains in wireless communication against disturbances cannot meet the growing demand, and there is an urgent need to tap and utilize new wireless environment resources. Since the electromagnetic wave propagation mechanism can be described by Maxwell's equations and their boundary conditions and the boundary conditions corresponding to differentiated wireless environments determine the differentiated equation solutions, we can use the differences in air domain resources to break through the

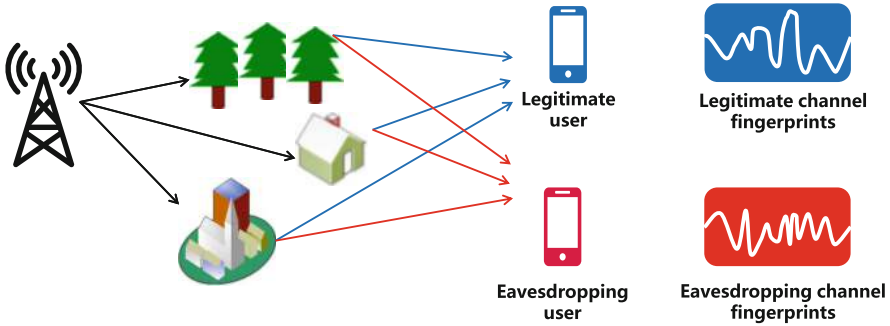


Fig. 8.2 Physical-layer security technology based on channel fingerprints

constraints of the public attributes of wireless resources such as time, frequencies, and codes. The idea contained in the third paradigm is that compared with public resources such as time, frequency, and code domains, air domain resources are personalized wireless resources, in which natural and inherent differences can be used to deal with wireless disturbances.

In the aspect of resisting natural disturbances, this paradigm mainly reduces the influence of fading and noise by means of transceiver diversity, spatial equalization/preequalization, space-time coding, and so on. Iconic technologies include diversity transmission/reception, beamforming, centralized and distributed multiple-input multiple-output (MIMO), massive MIMO, etc. Among them, the diversity technology compensates for fading channel loss by transmitting multiple copies of the same signal in the wireless channel and combining them at the receiving end, which can effectively improve communication quality and reduce transmission power. The multi-antenna technology is one of the hot technologies in the rapid development of wireless communication in recent years. From 3G smart antennas to 5G massive MIMO, the antenna scale, power gain, spatial diversity gain, spatial multiplexing gain, and array gain are constantly increasing. Space-time coding is a signal coding technology that constructs code words by combining space and time, which can obtain diversity gain and coding gain and offset fading.

In the aspect of resisting human disturbances, this paradigm mainly achieves interference suppression through directional transmission and spatial filtering. Take cellular cells and air domain anti-interference technologies as examples: cellular systems are widely used in mobile communications, that is, adjacent cells use different frequencies while distant cells reuse the same frequency, which can improve spectrum utilization while resisting interference; air domain anti-interference technologies use the adaptive antenna to make the array reception pattern form a null in the interference direction to avoid interference.

In the aspect of countering wireless security threats, this paradigm mainly uses ESS attributes such as randomness, diversity, and time variability of wireless channels to resist security threats at the physical layer, as shown in Fig. 8.2. Iconic technologies are wireless physical-layer security technologies, such as channel fingerprint encryption [6], channel fingerprint authentication [7], and

physical-layer secure transmission [8]. Among them, the channel fingerprint encryption technology and the authentication technology resist active and passive wireless attacks at the signal level by extracting unique and reciprocal wireless channel fingerprints of the sending party and receiving party and using their fingerprints for key generation and data authentication; the physical layer secure transmission technology is to design a strongly location-related signal transmission and processing mechanism according to the differences in channel fingerprints, so that only users in the desired location can demodulate the signal correctly, while signals in other locations are scrambled and unrecoverable.

This paradigm changes to “personalized air domain resources” and exploits the differences in the wireless environment to counter wireless disturbances, but it does not fundamentally change the situation limited by the wireless environment. The idea of expanding the degree of freedom of air domain resources is to improve the communication capacity and other performance indicators by increasing the unit size of antenna array elements. However, the limited antenna aperture and technical, economic, and engineering problems in reality lead to the restriction of an “air-spectrum wall,” as a result of which the ability to resist wireless interference is still in the stage of “relying on Heaven for food.”

8.1.1.4 Wireless Communication Development Paradigm 4: Transforming and Customizing the Wireless Environment

The restriction of “air-spectrum wall” makes the third paradigm unable to cope with the explosive growth of wireless communication data traffic and the diverse service quality requirements of terminals. Therefore, it is necessary to explore a new wireless communication development paradigm of “transforming and customizing the wireless environment.” The idea of this paradigm is wireless generalized uncertain disturbance can be attributed to the lack of controllability of the wireless environment; to completely solve this problem, the degree of freedom of wireless environment control must be improved, and the communication optimization goal can be achieved by transforming and customizing environmental constraints; and to transform and customize the wireless environment, we must proceed from wireless endogenous attributes. Therefore, the methodology corresponding to this paradigm is how to mine wireless endogenous attributes, realize endogenous structures based on endogenous attributes, and create an optimal environment against wireless disturbances. Specific manifestations include the following aspects: ① dynamic coding and on-demand reconstruction of the wireless environment (or channels) to resist natural disturbance, ② shaping and expanding the differences in the wireless environment to resist human disturbance, and ③ shaping and strengthening the ESS attribute of the wireless environment to defend against wireless security threats. This paradigm evolves from uncontrollable wireless channels to transformation and customization to realize the technological transformation from “relying on Heaven for food” to the integration of nature and man.

The DHR architecture of ESS provides a new mechanism to solve the problem of generalized uncertain disturbances for the new wireless communication development paradigm (paradigm 4). The theory and practice have proven that the DHR architecture of ESS can simultaneously cope with generalized uncertain disturbances such as reliability failures, human interference attacks, and loss of service trustworthiness [9] to achieve generalized robust control functions integrating functional safety, cybersecurity, and even information security. Therefore, wireless generalized uncertain disturbances can also be dealt with if a structure that conforms to the DHR attribute can be found in wireless communication. As a landmark technology of the new paradigm, the wireless ESS technology [10] explores the ESS effect and its scientific laws in the wireless network's own structure or operation mechanism from the common and original problems caused by electromagnetic waves and endogenous defects in cyberspace and solves the problems caused by natural and human disturbances in the wireless environment based on the utilization and transformation of endogenous properties in the wireless environment, providing the ability to resist "known unknown" and "unknown unknown" threats. The following is a detailed introduction to the problems and concepts of wireless ESS.

8.1.2 Wireless ESS Problems

Wireless ESS mainly ensures the safety and reliability of information transmission between the transmitter and the receiver. Different from cyberspace, wireless communication faces unique ESS problems caused by the inherent defects of electromagnetic wave propagation. The expected function of wireless communication systems is to use the broadcast characteristics of information-carrying electromagnetic waves to realize information interaction with anyone or anything at anytime and anywhere. However, due to the multipath propagation characteristics of electromagnetic waves and the uncertainty and uncontrollability of the electromagnetic environment, there are inevitably visible side effect functions (VSEFs) such as random fading and noises at the receiver. At the same time, the broadcast characteristics will also lead to invisible dark functions (IDFs), that is, passive eavesdropping is possible at any unknown place within the broadcast range, and unknown wireless access attacks can be launched.

Figure 8.3 shows the functions and effects of wireless communication, including expected functions, VSEFs, and IDFs. Due to the multipath propagation effect of electromagnetic waves, the signal will go through many different paths to reach different locations after it is sent from the transmitter. We expect the signals to pass through some accessible paths and converge at the legitimate receiver with great energy. However, signals experiencing different accessible paths have different phases and may fade (i.e., explicit side effects) when superimposed. Because fading occurs on the accessible path of the desired function, it can also be called the on-path VSEF. At the same time, the attacker may also receive signals from a path unknown to the legitimate person (i.e., off-path) to eavesdrop or launch an attack. This is the

Fig. 8.3 Causes for wireless ESS problems

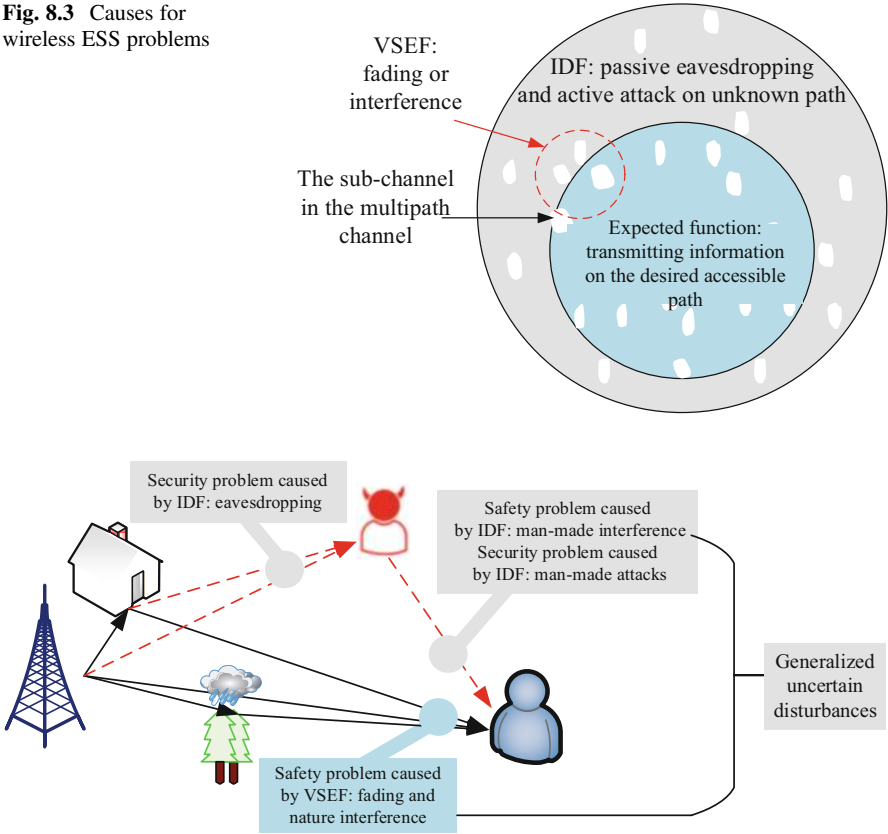


Fig. 8.4 Wireless ESS problems

off-path IDF, a dark function, of wireless communication. The above side effects and dark functions will cause ESS problems once triggered. As shown in Fig. 8.4, wireless ESS problems can also be divided into two categories. Generalized uncertain disturbances in wireless communication include functional safety problems caused by natural and man-made interference, as well as threats and damages caused by man-made attacks and eavesdropping.

8.1.3 Wireless ESS Attributes and Architecture

The key to solving ESS problems is to explore ESS attributes, build the ESS architecture, and realize ESS functions. Inspired by this idea, this subsection first discusses the ESS attributes of wireless communication and then attempts to design the wireless ESS architecture.

8.1.3.1 Wireless ESS Attributes

Different from wired network systems, wireless communication systems themselves have ESS attributes, which are caused by the propagation characteristics of electromagnetic waves and the electromagnetic environment. The propagation mode of electromagnetic waves can be expressed by Maxwell's equations as follows:

$$\begin{aligned}
 \nabla \cdot E &= \frac{\rho}{\varepsilon_0} \\
 \nabla \cdot B &= 0 \\
 \nabla \times E &= -\frac{\partial B}{\partial t} \\
 \nabla \times B &= \mu_0(J + \varepsilon_0 \partial E / \partial t)
 \end{aligned}
 \tag{8.1}$$

subject to $\Delta\xi$

wherein E represents the electric field intensity; B represents the magnetic induction intensity; ρ represents the charge density; J represents the current density; ε_0 and μ_0 represent the dielectric constant and magnetic conductivity in vacuum, respectively; ∇ is the vector differential operator; $\nabla \cdot$ and $\nabla \times$ represent divergence and curl, respectively; and $\Delta\xi$ is a set of boundary conditions or a set of equations that determine the relationship between the electromagnetic fields on both sides of the interface.

It can be seen from the completeness of Maxwell's equations that when the initial conditions and boundary conditions are determined, the system of Eqs. (8.1) has a unique solution. The boundary conditions are determined by various media (i.e., wireless environment) in electromagnetic wave propagation. Generally speaking, the medium experienced by electromagnetic waves received at each point in space is different. This means that the boundary conditions are different at different positions, resulting in different solutions to the above equations. Moreover, the complexity of the wireless environment usually makes it difficult to determine boundary conditions and predict electromagnetic waves at different positions. Therefore, the unpredictability of the wireless environment is the ESS attribute of wireless communication. A common concrete manifestation of the wireless environment is the wireless channel. Below we introduce the DHR characteristics in the ESS attributes of wireless channels.

1. Heterogeneity

Heterogeneity results from the unique propagation mode of electromagnetic waves and the complex propagation environment. Electromagnetic waves arrive at the receiving end in different paths and at different times through refraction, scattering, diffraction, and reflection effects. This causes the signals of each path to be superimposed on each other according to their respective phases, resulting in random multipath fading [11]. It can be seen from the above analysis that the boundary conditions of two points with large differences in space are different, so the fading of wireless channels is also different (or irrelevant). Unless the attacker

and the legitimate user are completely consistent in the air domain, time domain, and frequency domain, there must be differences between the channel from the attacker to the communicating parties and the legitimate channel [12]. The diversity and complexity of the wireless channel environment make boundary conditions difficult to predict. Therefore, for third parties in different locations, the wireless channel between legitimate communication parties has heterogeneity that cannot be replicated or measured.

2. Redundancy

Compared with traditional wired communication, the broadcast propagation of electromagnetic waves blurs the communication boundaries to a certain extent, allowing signals from the transmitter to be received at multiple points in space. When a multi-antenna system is used, each antenna within the array can acquire the transmitted signal and restore its information content. Moreover, the channels of antennas at different positions are also different, which allows a multi-antenna system to utilize redundant air domain resources to achieve diversity or multiplexing gains. Therefore, wireless channels have redundancy in the air domain [13].

3. Dynamics

The dynamic nature of wireless channels comes from natural time variability. Time variability is caused by the movement of transceivers and scatterers in the electromagnetic environment and changes in the propagation medium. The real-time changing electromagnetic environment causes the channels at the same location to be different at different times, which makes the channels dynamic. Although the dynamics of wireless channels are often regarded as a factor that is detrimental to communication reliability in wireless communication systems, inspired by the concept of ESS, we can use natural or artificial dynamics to construct an ESS mechanism. Specifically, the wireless channel can be used as a heterogeneous executor with dynamics and redundancy to build a DHR architecture, so that the protected system can obtain “unpredictable” attributes, thereby increasing the difficulty of attacks.

8.1.3.2 Wireless ESS Architecture

Taking advantage of the inherent ESS properties of wireless channels introduced in the previous subsection, we can design a wireless ESS architecture to inherit the security effects of the classic DHR architecture and cope with unknown security threats and their damage. As shown in Fig. 8.5, the heterogeneous executor corresponding to a typical mimic system is the wireless channel, the input agent is the transmitter, and the output ruling is the receiver. The area included in the above three is called the mimic interface, which is the scope that our wireless ESS mechanism should protect. In addition, the diversity technology for output ruling (such as maximum ratio combining, etc.) can reduce the impact of random noises, similar to the strategic ruling mechanism in the error correction output of the mimic system. After the transmitter uses the pilot sent by the receiver to estimate the

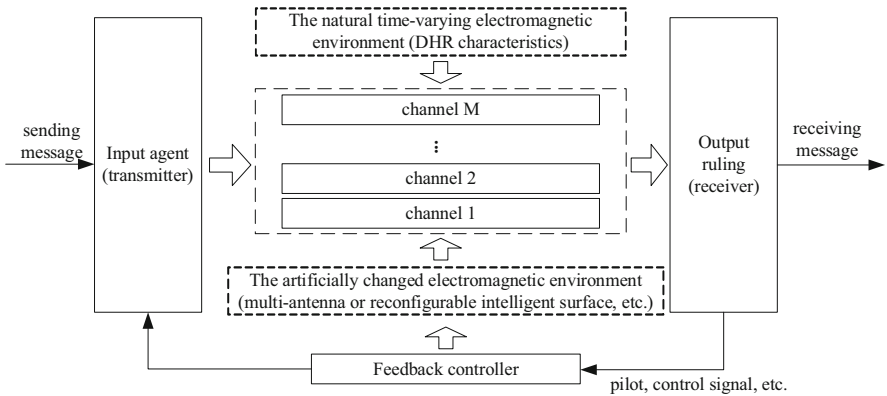


Fig. 8.5 Wireless ESS architecture

channel, the beam-forming technology can be analogous to the input distribution, strategic scheduling, and feedback control of the mimic system.

The working principle of the wireless ESS architecture is to use the wireless channel as the executor and bind the executor structure to the information while transmitting information on the executor. Its advantage is that legitimate users can take advantage of the DHR characteristics of the executor and its “unpredictable” attributes relative to third parties, so that attackers cannot obtain information about the structure of the executor and therefore cannot launch active attacks or passive eavesdropping on this mechanism. It can be seen that although the wireless ESS architecture does not eliminate the dark functions caused by the broadcast characteristics of electromagnetic waves, it avoids its adverse effects. Therefore, the wireless ESS architecture, like the classic DHR architecture, can be used to defend against unknown security threats.

Comparing wireless ESS with classical wireless communications, we can find that the essence of wireless ESS is to solve the problem of natural and artificial generalized uncertain disturbances based on the use of ESS attributes (DHR attributes) of the channel, while the essence of classical wireless communication is to combat the disturbances caused by natural channel uncertainty. The difference between the two lies in the design concept. Wireless ESS is based on fine cognition, optimization, and customization of the electromagnetic environment, while classical wireless communication passively adapts to the electromagnetic environment. What they have in common is that they both take advantage of the ESS properties of the channel. This shows that, different from the case of traditional add-on, bundled, and spliced security mechanisms, there is a symbiotic relationship between wireless ESS and classic wireless communications, i.e., they not only are endogenously integrated but also develop symbiotically. For example, advances in channel status information (CSI) estimation technologies can not only better resist random fading of channels but also make better use of CSI to scramble signals and enhance the ESS algorithm. Similarly, there is the channel coding technology. Good coding can not only

improve the robustness of the communication system but also facilitate the realization of the physical-layer secure transmission (wireless endogenous information security) technology.

8.1.4 The Assumption About the Wireless ESS System and Mechanism

Based on the wireless ESS architecture, we prospected the desired wireless ESS system, mechanism, and technical characteristics.

8.1.4.1 Desired Wireless ESS System

The wireless ESS system includes relevant specifications, standards, structural elements, usage mode, and solutions for wireless ESS. The desired wireless ESS system covers the following five aspects:

1. The wireless ESS system should be an open-air interface signal-processing architecture, which does not exclude any ESS problems involved in wireless signal transmission.
2. The wireless ESS system should be an integrated structure that can simultaneously provide high reliability, high trustworthiness, and high availability of functions.
3. The wireless ESS system should be able to take advantage of the attributes of complete intersection of dynamics, variety, and randomness (DVR), the three most important defense elements.
4. The wireless ESS system should have the structural elements of heterogeneity, redundancy, dynamics, as well as ruling and feedback control at the same time.
5. The wireless ESS system should be able to naturally accept the use of traditional security protection technologies or other technologies and obtain exponential defense gains.

8.1.4.2 Desired Wireless ESS Mechanism

The wireless ESS mechanism refers to the relationship and laws among the attributes, principles, structures, and other elements of wireless ESS. The desired wireless ESS mechanism includes the following five aspects:

1. The relationship between the wireless ESS mechanism and generalized uncertain disturbance should be a human-machine, machine-machine, and machine-human relationship.

2. The wireless ESS mechanism should be able to conditionally control or suppress generalized uncertain disturbances without attempting to eliminate their influence.
3. The effectiveness of the wireless ESS mechanism should rely neither on prior knowledge about the attacker nor on any additional, built-in or endogenous security measures or technical means.
4. The generalized security brought by the wireless ESS mechanism should have stability robustness and quality robustness with quantifiable design and verifiable measurement.
5. The effectiveness of the wireless ESS mechanism should have nothing to do with or at most weak correlation to the technical capabilities and past experience of the operation and maintenance managers.

8.1.4.3 Desired Technical Characteristics of Wireless ESS

The desired technical characteristics of the wireless ESS architecture include the following four aspects:

1. Wireless ESS is an intrinsically safe and secure function of electromagnetic objects. It has a point-to-surface integrated defense function similar to the nonspecific and specific immune mechanisms of vertebrates and is inseparable from the intrinsic functions of the target object.
2. The wireless ESS function does not rely on any prior knowledge about or behavioral characteristics of the attacker, so it naturally suppresses the effect of “differential-mode attacks” formed by independent attack resources, attack technologies, and methods.
3. To break through wireless ESS defenses, escape is possible only through precise coordinated attacks with spatial-temporal consistency. However, to create a situation of common-mode escape, the first thing to do is overcoming the observer effect of spatial-temporal inconsistency and crossing the distance of physical isolation formed by the heterogeneous environment or deciphering the “structural coding.”
4. Theoretically, differential-mode escape is impossible to occur, and common-mode escape is also a minimal probability event, so the actions or results of attacks in an ESS environment have neither stability robustness nor quality robustness.

8.1.5 Functions and Technologies of Wireless ESS

Relying on the wireless ESS architecture, ESS functions can be realized and ESS problems can be solved. This subsection illustrates two main applications of wireless communication ESS functions, including the wireless endogenous information security technology and the wireless signal functional safety technology.

8.1.5.1 Wireless Endogenous Information Security Technology

The purpose of the wireless endogenous information security technology is to protect the confidentiality, trustworthiness, and integrity of information. As shown in Fig. 8.6, we use the wireless channel as the executor to build a dedicated channel for secure communication. It can not only realize the physical-layer secure transmission technology that randomly scrambles signals [14] of randomly scrambling signals but also can realize the physical-layer key generation technology based on one-time pad [15].

The essence of the physical-layer secure transmission technology is to design a signal transmission and processing mechanism that is strongly related to the wireless channel, so that only users on the coded channel can correctly demodulate the signal, while signals at other locations and on non-coded channels are scrambled and unrecoverable. The essence of the physical-layer key generation technology is to directly use the channel to generate the key, which can be regarded as a simplified representation of the executor. After the key is used for authentication or encryption, the information can be bound to the executor structure, thereby preventing attacks or eavesdropping from other unknown locations. The realization of the above two wireless ESS functions does not rely on any prior knowledge, behavioral characteristics, or additional security measures of the attacker, so it can deal with unknown security threats and damage.

Moreover, the existing literature [16] points out that the physical-layer key generation technology can be used to implement one-time pad. The said literature points out that the amount of information in the electromagnetic environment is large enough, and the communication process of transmitting the information source naturally contains the same amount of information about the electromagnetic environment, which can provide sufficient keys. Therefore, as long as fine cognition and shaping of the electromagnetic environment (encoded channels) can be achieved, no matter how high the achievable communication rate is, a matching key rate can be

Fig. 8.6 Wireless endogenous information security technology

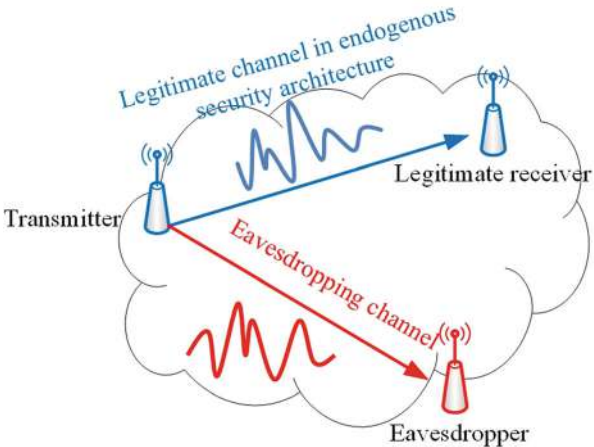
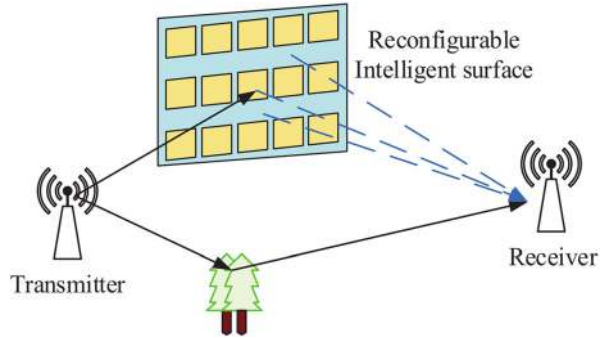


Fig. 8.7 RIS-assisted wireless communication



extracted from the endogenously rich mine of the electromagnetic environment, and no extra energy or resources are consumed in the extraction process.

Reference [16] also shows that we can improve the randomness and dynamics of channel fingerprints through some enabling techniques. As shown in Fig. 8.7, reconfigurable intelligent surface (RIS) can effectively, quickly and flexibly control electromagnetic waves, enrich and amplify DHR characteristics of the electromagnetic environment, and enhance wireless endogenous information security and functional safety. Generally, information security and functional safety related to RIS are called RIS-based information transaction (RBIT) and RIS-assisted wireless communications (RAWC), respectively [17]. For example, literature [18] provides a method to improve the key generation rate by increasing channel entropy through RIS in a static environment. These enabling technologies help realize one-time pad ESS functions based on the ESS attributes of the electromagnetic environment.

8.1.5.2 Wireless Signal Functional Safety Technology

Wireless ESS provides not only highly trustworthy information security but highly reliable and available functional safety and realizes integrated ESS functions of communication, cybersecurity, and anti-interference. The purpose of many traditional wireless communication technologies is to ensure the functional safety of communication, such as the spread spectrum communication [19], the multi-antenna diversity technology, etc. They are mainly used to counter generalized uncertain disturbances including uncertain interference of natural channels and artificial disturbances. These technologies generally use some DHR features and are ESS technologies based on a typical DHR architecture, such as the wireless communication anti-interference technology. By domains, anti-interference technologies are generally categorized into time domain [20], air domain [21], and frequency domain [22] anti-interference technologies. Taking the classical time-hopping, frequency-hopping, space-hopping, and code-hopping communications as examples, they use the redundancy of time domain, frequency domain, air domain, and code domain, respectively, and randomly select different time and frequency points or executors on the antenna (i.e., wireless channels) to communicate according to the hopping

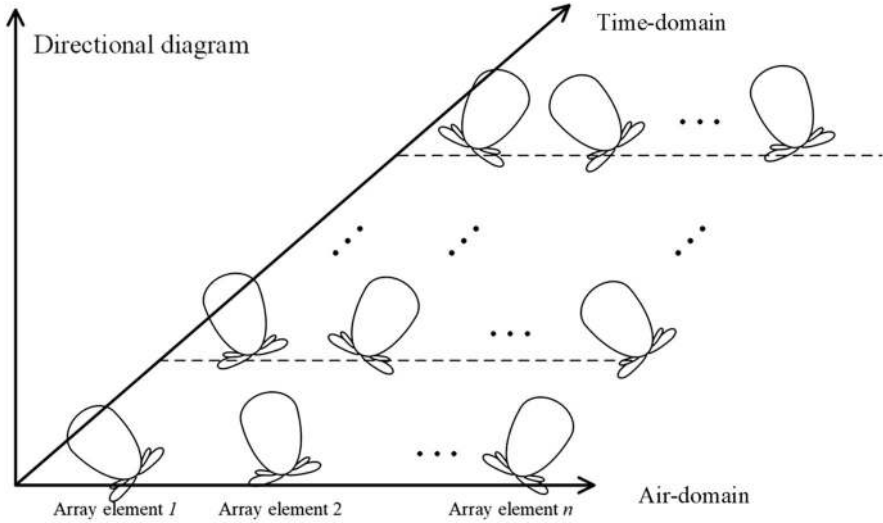


Fig. 8.8 DHR antenna array based on metamaterials

image. Therefore, the anti-interference technology of wireless communication introduces dynamic and random elements, which enable the defense mechanism to have an observer effect and effectively increase the difficulty for the attacker to interfere with wireless communication. However, most of the existing arrays use isomorphic array elements, which have limited heterogeneity and freedom. Literature [23] proposes a space-time heterogeneous antenna array, in which each heterogeneous antenna can quickly scan and distinguish multiple paths, so as to achieve refined perception and manipulation of the wireless environment. The DHR antenna array based on metamaterials is constructed with spatial diversity so that array elements in different positions have heterogeneous and uncorrelated patterns. The array can free air domain resources from the constraints of the air-spectrum wall and improve performance under the condition of limited antenna aperture. As shown in Fig. 8.8, theoretically, N elements can be agile for k times to realize the actual degree of freedom of Nk , and the receiving capacity increases linearly with the degree of heterogeneity (including the degree of time domain heterogeneity and the degree of air domain heterogeneity). Therefore, the DHR antenna can not only obtain array gain but also obtain array element gain, thereby amplifying the dynamics and heterogeneity of the wireless ESS architecture, improving the signal-to-noise ratio and anti-interference capability, and obtaining better endogenous functional safety of wireless communication.

8.1.6 Wireless ESS Performance Analysis

In essence, wireless ESS takes advantage of the channel heterogeneity of different time and space environments; the redundancy of space, time, and frequency resources; and the dynamic changes in the wireless propagation environment to achieve optimal reception of expected channels, while increasing the uncertainty of received signals on other heterogeneous channels and resisting wireless access attacks from abnormal channels by sensing the heterogeneity of channels. Therefore, on the one hand, we can put forward the quantitative design and verification measurement method of wireless ESS based on the reliability theory and realize the analysis and evaluation of wireless ESS performance. On the other hand, we can use the ESS attributes of wireless channels and the principle of CESS to design and evaluate the generalized uncertain disturbances and security risks faced by the wireless ESS system mechanism through the correlation and heterogeneity metrics between legitimate and eavesdropping channels.

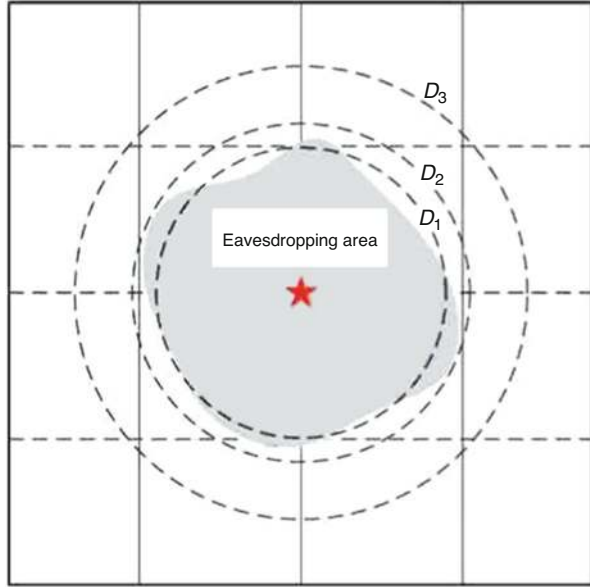
8.1.6.1 The Principle Wireless ESS Assessment

Since the DHR architecture is a logical expression and implementation of the “relatively true axiom,” the uncertain disturbance problem that cannot be perceived in the non-heterogeneous redundant scenario in the classical reliability theory can be transformed into a probability-based perceptible differential-mode or common-mode problem in the equivalent scenario of the relatively true axiom, and it has nothing to do with the specific nature or behavioral characteristics of uncertain disturbances, that is, it does not depend on prior knowledge. Therefore, referring to the principle of CESS and the classical reliability theory, we can transform the wireless ESS threats that are difficult to calibrate, design, test, and measure into probabilistic generalized robust control problems (GRCPs), develop the methods of quantitative design and verification measurement for wireless ESS, establish an evaluation model reflecting the relationship between wireless channel ESS attributes and wireless ESS performance, form a technically feasible and standardizable evaluation indicator system and testing method, and realize the integrated verification, evaluation, and quantitative design of service availability, communication reliability, and security trustworthiness.

8.1.6.2 Safe Isolation Distance

By establishing the mapping relationship between reliability indicator, wireless channel safe isolation distance, and threat area, the above safety performance can be transformed into a signal reception error rate and recognition success rate under physical isolation distance in heterogeneous environments. Specifically, since wireless-side security originates from channel differences, the security performance

Fig. 8.9 Schematic diagram of safe isolation distance



depends on the correlation between legitimate and eavesdropping channels, and channel correlation can be characterized by a safe interval. Therefore, the safe isolation distance can be used to evaluate the security performance and guide actual system design.

The safe isolation distance is defined as follows: if the legitimate user's bit error rate is 10^{-6} , the distance between the eavesdropper and the legitimate user is greater than D , the eavesdropping user's bit error rate is $>40\%$ with a probability of 95% , and at this time, the eavesdropper's signal-to-noise ratio is at least 10 dB lower than that of the legitimate user, and then D is called the safe isolation distance.

In order to evaluate the safe isolation distance, it is necessary to first determine the safety requirements and reliability requirements [24]. Take the data service in communication as an example. Generally, reliability usually requires that the receiving bit error rate of legitimate users is $<10^{-6}$, and security requires that the bit error rate of eavesdroppers is $>40\%$. If this standard is met, it is considered that the legitimate user can communicate normally in practice and the eavesdropper cannot obtain any information. While ensuring security, it is also necessary to ensure that the communication rate can meet certain service needs. When the system bandwidth is fixed and the eavesdropper's bit error rate is close to 50% , the eavesdropping rate is close to 0 bps. At this time, the communication rate of the legitimate user is the safe communication rate. Since the communication rate is positively related to the signal-to-noise ratio, the difference in the signal-to-noise ratio can be equivalent to the size of the safe transmission rate, and a difference of 10 dB can ensure that the safe channel at this time has a sufficient information transmission rate.

The assessment method of safe isolation distance is shown in Fig. 8.9. The legitimate receiving device is located at the five-pointed star mark in the figure, where communication is normal and meets reliability requirements. At the same time, eavesdroppers are placed around the legitimate receiving device, and whether the security requirements are met is measured. If the security requirements are not met (i.e., the eavesdropper's bit error rate is $<40\%$), the eavesdropper's position at this time belongs to the threat area and is marked gray; outside the gray area, the eavesdropper's bit error rate is $>40\%$.

Then, we take the legitimate receiving device as the center of the circle and delineate a number of circular areas, that is, the areas enclosed by the dotted lines in the figure, until it is ensured that on the delimited circle, the arc length that intersects with the gray area is 95% of the circumference of the circle, and then the radius of the circular area is the safe isolation distance D of transmission. With the expansion of radius D , the eavesdropper's bit error rate will approach 50%, the difference in the signal-to-noise ratio between the eavesdropper and the legitimate user will widen, and communication security will gradually improve.

The security performance assessment metrics and corresponding measurement methods of wireless ESS systems based on the safe isolation distance are the basic idea of taking location safety as the practical technical route for wireless ESS, which can provide theoretical support for realizing calibratable design and testable measurement in engineering.

It is foreseeable that the theory of CESS and the principle of the DHR architecture will play an important guiding role in the future development of wireless communication technologies. Moreover, the unique DHR architecture of wireless communication systems will also become an important supplement to the CESS theory. For example, there will be a large number of machine-like communication scenarios in the future, where communication nodes will have the characteristics of dense distribution, high concurrent communication, low communication delay, and dynamic migration, making the mobile Internet face more complicated problems in security management. Because the wireless channel has natural DHR characteristics, when used as a heterogeneous executor, it can not only reduce the collaboration of attackers but also relax the heterogeneity requirements of the DHR architecture on the software/hardware structure or algorithms, thus reducing the cost of introducing heterogeneous redundancy, and this has remarkable engineering significance. Thus, the integration and mutual penetration of the ESS theory and traditional wireless communication information security and functional safety technology can not only provide guidance for the development of new wireless communication reliability and security methods but also provide continuous progress for the ESS theory based on the mimicry principle.

8.2 ESS-Empowered Artificial Intelligence

8.2.1 An Introduction to Artificial Intelligence Application Systems

Artificial intelligence (AI) is an important driving force for the new round of scientific and technological revolution and industrial transformation. Accelerating the breakthrough in key AI technologies and industrial development is of great significance to enhancing the country’s core competitiveness, reshaping new development momentum, and stepping up the intelligent transformation of the economy. In 2017, the Chinese government issued the “Development Plan for a New Generation of Artificial Intelligence” [25] and implemented the arrangement and deployment of major scientific and technological projects for a new generation of artificial intelligence. After years of construction, progress has been achieved in stages, and a large number of application systems have emerged with AI as the core. These systems have created new business formats, new models, and new markets and promoted the digital and intelligent transformation in nearly all fields. As a special type of information system, AI application systems can be understood in both a broad sense and a narrow sense.

In a broad sense, an AI application system generally refers to a service system that uses the AI technology. Figure 8.10 shows the typical component structure of deep learning at present, in which the basic software and hardware environment, AI

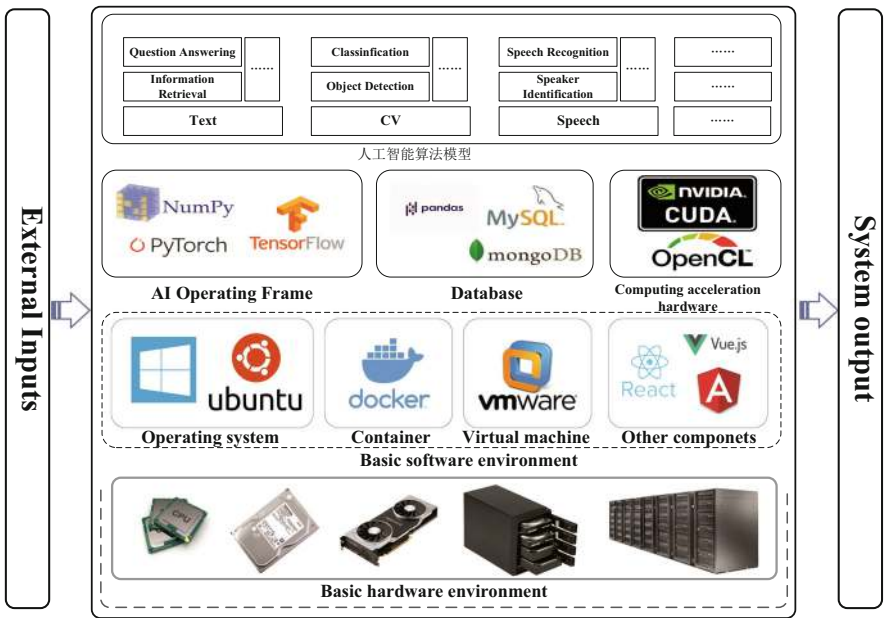


Fig. 8.10 Components required for the operation of AI application systems

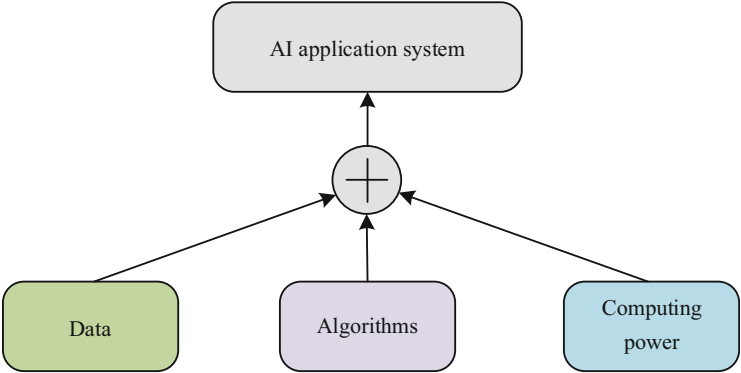


Fig. 8.11 Core elements of AI application systems in a narrow sense

operating framework, database, and computing acceleration hardware together constitute the “base” of the AI application system, and on this basis, an AI algorithm model for a specific application can be obtained through massive data training and learning.

In a narrow sense, AI application systems only focus on the three core elements that determine the AI model, namely, data, algorithms, and computing power. The relationship between these three is shown in Fig. 8.11. Among them, data is the “blood” of the AI application system, and the cornerstone for algorithms to gain a comprehensive insight into data semantics; algorithms are the “soul” of the AI application system, and the driving force for the in-depth application of AI; computing power is the “body” of the AI application system and an important support for the operation of intelligent algorithms. It is precisely because of the continuous accumulation of data resources, the innovation of algorithms, and enhancement of computing power that AI application systems have made substantial progress and are driving AI application in various fields toward a new development stage of all-factor intelligent transformation.

According to the above analysis, whether in a broad sense or a narrow sense, all AI application systems take artificial intelligence as the technical core and contain three core elements: data, algorithms, and computing power.

8.2.2 Analysis of Security Threats Faced by AI Application Systems

With the widespread application of AI technologies represented by deep learning, its security problems have attracted more and more attention from academia, industry, and even the society. As the absolute core technologies in the future digital and intelligent era, if the security of AI systems cannot be guaranteed, then the use of AI to do evil against human beings will not only stay in the movie plot of science fiction

films such as *The Matrix* and *I, Robot* but will become a real disaster one day in the future. From the perspective of CESS, this section discusses the typical security threats and damages in AI application systems from three aspects: common ESS problems, individual ESS problems, and GFS problems.

8.2.2.1 Common ESS Problems in AI Application Systems

Like other application systems, the entity of an AI application system relies on physical information systems, so its algorithm model “base” is bound to face common ESS problems. Domestic and foreign research reports show that there are widespread security vulnerabilities in the software and hardware environments on which the mainstream deep learning framework relies. Once these vulnerabilities are exploited by attackers, AI systems will face the risk of destruction, tampering, and information theft.

In terms of software, current foreign platforms such as TensorFlow, Torch, and Caffe have all been reported to have security vulnerabilities. According to the data of GitHub, an open-source software community, since 2020, TensorFlow has been exposed to more than a 100 security vulnerabilities, which can lead to system instability, data leakage, memory corruption, and other problems. In 2021, the “360 Company” conducted a security evaluation of mainstream open-source AI frameworks at home and abroad and found more than 150 vulnerabilities in seven machine learning frameworks (including the currently most widely used TensorFlow, PyTorch, etc.), and more than 200 vulnerabilities in the framework supply chain. This finding was confirmed by the DoS attacks, evasion attacks, and system downtime in TensorFlow, Caffe, and Torch exposed in 2017. A Tencent security team also found that there were major vulnerabilities in a TensorFlow component, which can make the robot program written by developers based on this component easily controlled remotely by hackers.

In terms of hardware, the GPU hardware products that AI systems mainly rely on also have security vulnerabilities. Among them, the most severe ones are the “Meltdown” and “Specter” vulnerabilities exposed in 2018, which affected multiple series of products including GeForce, Tesla, Grid, NVS, and Quadro, basically covering most of the product lines of NVIDIA, a mainstream GPU manufacturer. In the same year, researchers at the University of California, Riverside, targeted security vulnerabilities in NVIDIA GPUs [26] and discovered three methods that could be used by hackers to breach user security and privacy. In addition, research shows that the neural network model can be destroyed through GPU/CPU overflow, which makes the model invalid or become a backdoor network.

8.2.2.2 Individual ESS Problems in AI Application Systems

The individual ESS problems of AI application systems are security problems caused only by the “contradictory” characteristics of the AI model algorithm without

considering the “base” of the algorithm model. Taking the deep neural network (DNN), which is almost dominant in AI applications at present, as an example, the root causes for its individual ESS problems mainly come from three aspects.

First, the model space of DNN is extremely large and lacks interpretability. As a typical big data-driven technology, the most obvious characteristic of DNN is “knowing what it is, but not knowing why.” AlphaGo Zero, launched by Deepmind in 2017, trained 4.9 million chess games on its own in 3 days without human intervention and defeated all the best human players in the world in 40 days. But so far, its designers have not been able to give the meaning of weights, nodes, or layers in its algorithm model, let alone predict the possible influence of each model parameter on the performance of the whole model. This, to some extent, has limited the application of AI systems in decision-making.

Second, DNN’s overreliance on samples leads to a lack of adaptability in its results. The learning and training process of DNN is a feature fitting process of sample data, but this fitting is often incomplete because the samples are usually difficult to cover all complex conditions in reality. Tesla’s “blue sky and white clouds” out-of-control accident occurred when the onboard image recognition system of its Model Y self-driving car mistook a white semitrailer truck for the sky and judged that it could pass normally, leading to a major car accident. This shows that the current AI systems have limited ability to identify and judge target objects in some complex scenarios.

Third, the unstable gradient of DNN leads to uncertainty in learning results. The learning and training of DNN are processes of repeatedly adjusting the input-output mapping, and the extreme value of its loss function is generally obtained by the gradient descent of the model. In this process, the model needs to be placed in a dense space of input sampling points in order to learn a reliable mapping from the input space to the output space. However, the commonly used back-propagation mechanism leads to the gradient descent of DNN in the training process, as a result of which, it is impossible to stably guide the system to achieve optimality, and the learning results will be uncertain.

Facts have proved that by effectively utilizing the defects of the DNN model, various attack methods such as data backdoors, data poisoning, and adversarial samples can be formed.

A data backdoor means that the attacker buries a backdoor in the model so that the infected model behaves normally under normal circumstances and outputs the set malicious target when the backdoor is activated. This attack method is easy to implement when training on a third-party platform or deploying a model provided by a third party. It is well concealed and difficult to discover, locate, and track. Data poisoning relies on the contamination of training data sets to induce AI algorithms to produce indiscriminate errors or comprehensively reduce the performance of the model, thereby destroying the usability of an AI system.

Adversarial samples are the hottest issue in current DNN security researches. Based on the analysis and test of the “weaknesses” of the target system, the attacker adds carefully constructed tiny “disturbances” to the target input without changing the DNN model. This causes DNN to produce abnormal output and then leads to

serious misjudgments or errors in the entire application system. In 2014, Szegedy et al. verified the existence of adversarial samples [27], and subsequent researchers successively put forward methods of adversarial attacks such as FGSM (Fast Gradient Sign Method) [28], BIM [29], PGD (projected gradient descent) [30], DeepFool [31], C&W [32], and Nattack [33]. According to whether the attacker can know and use the network structure and parameters of the attacked model in the analysis stage, adversarial attacks can be divided into black box attacks and white box attacks; according to whether the attacker sets a specific wrong target for the abnormal output generated by the disturbance, adversarial attacks can be divided into targeted attacks and nontargeted attacks.

8.2.2.3 GFS Problems in AI Application Systems

The common and individual ESS problems of an AI application system mainly explain the causes for its security threats, while the GFS problems focus on the consequences of the security threats. In recent years, deep learning has experienced rapid development. It has been widely used not only in natural language processing, speech recognition, and computer vision but also in manufacturing, transportation, medical care, and people's livelihood, showing a development trend of "AI +Everything." As AI is deeply integrated into human production and life, its GFS problems will become increasingly prominent. Based on the individual and common problems of the AI application system, attackers can wantonly destroy and maliciously tamper with the functions of the target system, resulting in adverse effects.

First of all, the field of traffic driving will be one of the hardest hit areas for GFS problems of AI. In 2015, researchers remotely controlled the steering, braking, acceleration, and engine shutdown functions of Jeep Chrysler by exploiting a vulnerability in its onboard system Uconnect, resulting in the recall of 1.4 million vehicles. At the 4th "Qiangwang" International Elite Challenge on Cyber Mimic Defense in 2021, all the 15 sets of eight commercial ADAS were found to have vulnerabilities, which were exploited by many domestic and foreign teams. Some of the vulnerabilities could even be used to remotely control the car and make it suddenly turn, accelerate, and open or close the door without any warning.

In addition, robots and the manufacturing industry will also be severely affected. In 2017, IOActive, a security company, tested 50 robots of various types and found many vulnerabilities. Hackers could exploit these vulnerabilities to remotely hijack and manipulate the robot's arms and legs or control the microphone and camera, which could endanger the personal and property safety of users. In September 2021, Weir Group, an intelligent manufacturing enterprise, suffered a cyberattack, which caused interruptions in shipping, manufacturing, and engineering systems, resulting in indirect losses of 50 million pounds in a single month.

In the future, the intertwined impact of cybersecurity, functional safety, and information security problems will only become more serious. Therefore, AI systems must be able to deal with random disturbances and artificial or non-artificial uncertain interferences and have the GFS attributes to solve the problems of

functional safety, network security, and information security in some cases in an integrated manner. Otherwise, it will be difficult to meet the “resilience or toughness” requirements of system function/performance under cyberattacks.

8.2.3 Design of an Endogenously Safe and Secure AI Defense Framework

AI application systems face ESS problems at two levels: the “base” of software/hardware environments and the “ontology” of model algorithms, so the threats and challenges they face are more severe. In the aspect of common problems, through the practical development of ESS in cloud platforms, storage systems, routing and switching, and other network devices in recent years, the “base” environments of AI application systems such as information communication networks, clouds, and data centers have the ESS attributes of trusted services, which provide a feasible solution to the common ESS problems of AI. With respect to individual problems, the author takes DNN, a hot topic at present, as an example for research and discussion.

According to the analysis in Sect. 8.2.2, the neural network of AI is a feature engineering based on gradient optimization for fitting. This process makes the existing models pay attention to all the features instead of ignoring the subtle features as human beings do. Therefore, at present, whether it is a black box or white box adversarial attack, the basic idea is to achieve the attack goal of deceiving the model by constructing the input of model loss gradient descent on the premise of ensuring less disturbance (microscopic characteristics). Just as the vulnerabilities and backdoors in the software/hardware of an information system cannot be predicted or exhausted in advance, the optimization methods adopted by neural networks at present can only try their best to approach but can never achieve the perfect goal of “understanding” everything when the training set is limited. This problem is an architectural defect of the neural network itself and can be regarded as a vulnerability in the neural network model algorithm. From this point of view, adversarial attacks, as the most representative individual ESS problem of AI, are similar to common ESS problems in terms of root causes, presentation forms, and exploitation methods, and it seems that they can be defended against in an integrated manner from the perspective of ESS.

From the perspective of the ESS defense paradigm, the feasibility of implementing security protection at the algorithm level of the DNN model based on the dynamics, variety, redundancy of the DHR architecture, and the SR-FC mechanism mainly includes the following two aspects:

1. The adversarial performance of each functional equivalent conforms to the relatively true axiom. It is found that the correct classification boundaries of networks with different structures are similar when searching for gradients in different directions at the same decision point [34, 35], as a result of which the

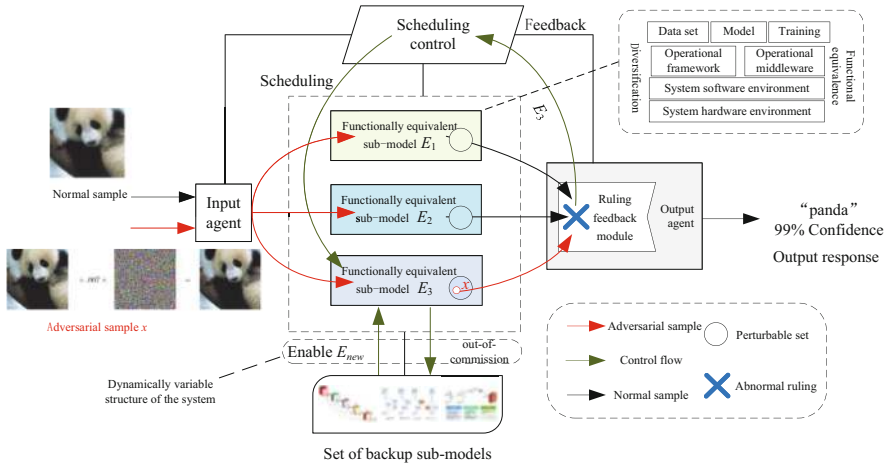


Fig. 8.12 DHR-based ESS defense framework of AI

same disturbance can make different models go wrong, that is, adversarial attacks are migratable. At the same time, however, the direction of the gradient descent of each model is extremely random, which leads to different wrong results, so it is difficult to achieve a migratable attack on a specific target. Therefore, the effectiveness of diversity and heterogeneity of neural network sub-models in DHR is guaranteed.

2. The input and output interfaces of functionally equivalent reconfigurable executors can be normalized or standardized. For a neural network, the input data to be identified or classified is its normalized input, and the result of identification or classification is its normalized output. On this interface, under the excitation of a given input sequence, neural network sub-model executors with functional equivalence have the same probability of multiple output vectors or states, which makes it possible to judge and ensure the equivalence among sub-model executors through the consistency test method of a given function or performance.

Figure 8.12 shows the DHR-based ESS defense framework of AI, where multiple functionally equivalent neural network sub-models are used to construct a heterogeneous redundant operating environment, the input agent distributes samples to each sub-model for independent processing, and the identification or classification results enter the strategic ruling. For normal samples, each sub-model can give the same or similar results. For adversarial samples, the sub-model will be triggered to produce differential-mode output, so they are very likely to be discovered by the ruling module; the error correction output link and system scheduling module will be activated, and the algorithm model will be dynamically replaced according to certain rules, thus avoiding the current adversarial attack.

Under the above defense framework, how to mine and construct effective diversity for neural networks becomes the key. The core elements of neural networks

include the data set, the network model, and the training method, all of which can be used to construct the entry points of heterogeneous sub-models. What needs to be noted is that due to the similarity in the internal mechanisms of the trained models and methods, even if there are differences in the model structure, the characteristics and methods of learning are still similar, so the same adversarial sample can make different models go wrong, that is, the problem of target migratability still exists. Therefore, if we want to achieve system-level robustness from dynamics and unknowns, we need to conduct in-depth research and experiments to further explore how to obtain differentiated neural network models.

8.2.4 ESS Experiment of AI

In order to build a robust AI application system based on the ESS architecture, a variety of heterogeneous AI model algorithms are needed to make it present different output states or maintain correct output when subjected to adversarial attacks and ultimately ensure the consistency of the entire system. Below we take the image classification and recognition service, which is most widely used and most frequently attacked with various methods, as an example to verify the feasibility of ESS-empowered AI application systems.

8.2.4.1 Experiment on the Differences of Adversarial Attacks Against Neural Network Models

In this subsection, various DNN models and their corresponding adversarial samples are cross-tested to verify and analyze the differences when adversarial attacks occur.

Two white box attack methods (FGSM and PGD) and one black box attack method (NATTACK) are used to test the attacks on the currently disclosed models with different structures (the model architecture, width and depth). The image recognition framework adopts the TIMM library, and the target DNN models (hereinafter referred to as sub-models) are RESNET-152, RESNET-18, RESNET-50, vgg-11, vgg-19, vit, efficientnet-b0, densenet121, dla34, etc. Three attack methods are used to generate 1000 adversarial samples, respectively, and the classification results of cross-attack tests are counted and compared. The test results are shown in Fig. 8.13.

The experiment shows that the adversarial attacks on all sub-models are effective in the test. However, statistical analysis of these error distributions shows that most of the model error classification results tend to be different. In a random test example of 1000 pictures, the number of cases where all sub-models is biased toward the same error is 0; more than half (>4) of the sub-models are biased toward the same error, and the proportion is $<5\%$ (FGSM 1.5%, PGD 0.4%, NATTACK 4.3%); the proportion of those with no or only one model error that is consistent with itself is more than 80% (FGSM 91.1%, PGD 93.9%, NATTACK 84.6%). Therefore, even

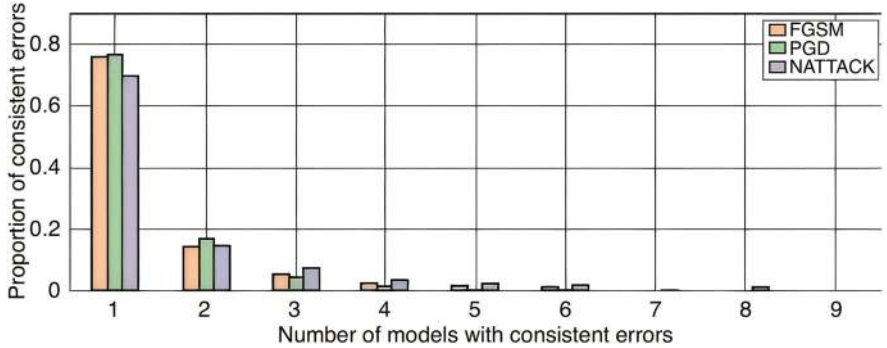


Fig. 8.13 Quantitative distribution of multi-model error classification consistency under different attack methods

when adversarial attacks are in effect, different DNN models show differences in error classification performance.

8.2.4.2 Experiment on Adversarial Sample Migration Targets for a Single Neural Network

This subsection continues to focus on the differences of DNN under adversarial attacks. Based on the previous experiment, the restrictions on the attacker are increased. After selecting a single model to generate adversarial samples, they are directly input into other model networks to observe the target migration. The method of adversarial attack is PGD, and the number of adversarial samples generated is 1000. The experimental results are shown in Fig. 8.14.

The experiment shows that adversarial samples generated by attacking a single model have certain migratability in different models. Without adding any other disturbance, the adversarial samples generated by one model attack will, more or less, cause other models to make mistakes, but the range of decline of each model is obviously different. This shows that the number of layers, the width, and the architecture of the model will have a certain impact on the migration effect, that is, the more similar the structure of the model, the stronger its migratability of successful attacks. As shown in Fig. 8.14, VGG11 and VGG19 models have significantly higher migratability than other models; the VIT network based on the patch mechanism obviously has better independence and higher immunity to disturbances and attacks, and the migratability of its own adversarial samples is significantly lower than other models.

This experiment, together with the above experiments, confirms the establishment of the prerequisite of Sect. 8.2.3, “The adversarial performance of each functional equivalent conforms to the relatively true axiom,” and provides a certain basis for “constructing neural network sub-models with the greatest possible difference for adversarial attacks.”

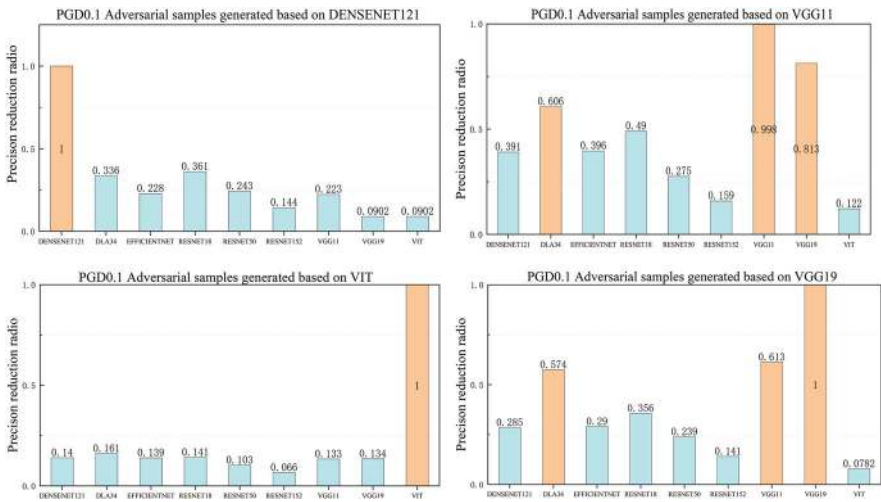


Fig. 8.14 Migration ability under the same attack method (strength 0.1)

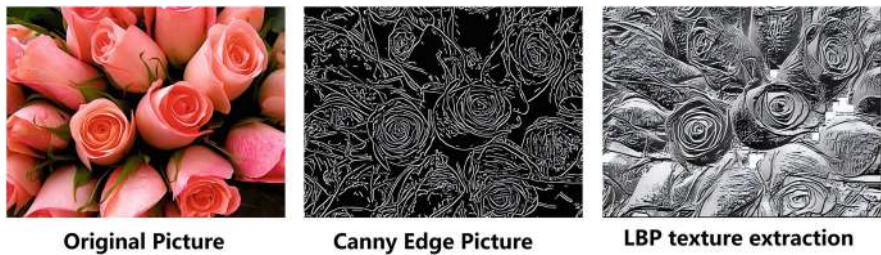


Fig. 8.15 Different feature extraction renderings for the original picture

8.2.4.3 Neural Network Model-Integrated Defense Experiment Based on Feature Heterogeneity

As mentioned earlier, the basis of feature learning is data sets. If the data sets are homologous and similar, then even if there are differences in a network structure, each model will have similar understanding of features, resulting in stronger migratability of adversarial samples. Therefore, this subsection explores the possibility of constructing diverse heterogeneous DNN sub-models from the perspective of training and learning data sets.

Firstly, a five-classification data set of flowers (about 10,000 pictures) is collected and constructed. Based on this data set, features are separated through a variety of feature extraction methods, among which the Canny algorithm for edge detection, the LBP algorithm for texture extraction, the GLCM algorithm, and their different statistical variants are adopted, as shown in Fig. 8.15. In this way, different training

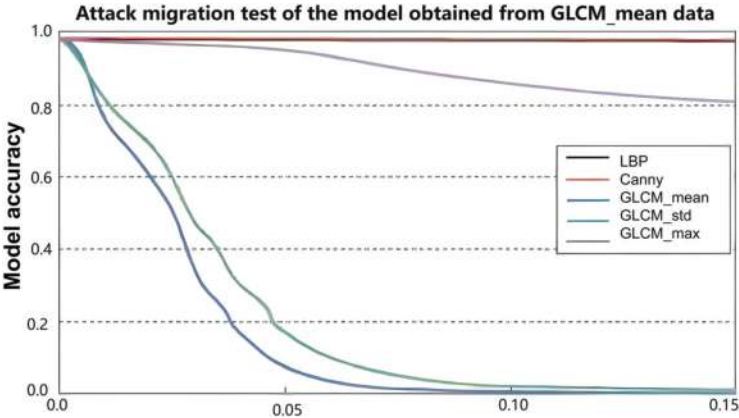


Fig. 8.16 Migratability of adversarial samples for different training sets

data sets can be separated based on the same original data set. For different data sets obtained through conversion, the model training remains consistent with the original task goal of completing this simple five classification. Here, the simple RESNET-18 network provided in the TIMM library is used for training, which is stopped when the test set accuracy of all models reaches 90%.

Secondly, to verify the model corresponding to a certain type of features, several pictures are selected to generate adversarial samples and obtain the coefficient of adversarial disturbance, and then the disturbance is added to the input pictures corresponding to the other models to test the migratability of the disturbance. Here, the BIM method is used to generate adversarial samples, the number of iterations is limited to 40, and the iteration step size is controlled to obtain adversarial samples with different disturbance intensities. By evaluating the overall accuracy of the selected simple data sets, we measure whether the adversarial disturbance can simultaneously affect the models generated based on different training sets. The different data sets converted by the GLCM method are described by the characteristic parameters of the grayscale conjugate matrix derived from the GLCM method. The GLCM_mean is the average value of the selected parameters, the GLCM_std is the standard deviation, and the GLCM_max is the maximum value. The test results are shown in Fig. 8.16.

As shown in Fig. 8.16, when the accuracy (vertical axis) of the current model (GLCM_mean) being attacked decreases rapidly with the increase of disturbance, there are obvious differences in the models trained based on the features of other data sets. Among them, the model obtained from the data converted by the Canny and LBP algorithms is almost unaffected, while the GLCM_std has the same rapid decline in accuracy as the current model. The reason is that the image feature extraction methods of Canny and LBP algorithms are quite different from the current model, while the GLCM_std, like the current model, is based on the gray value

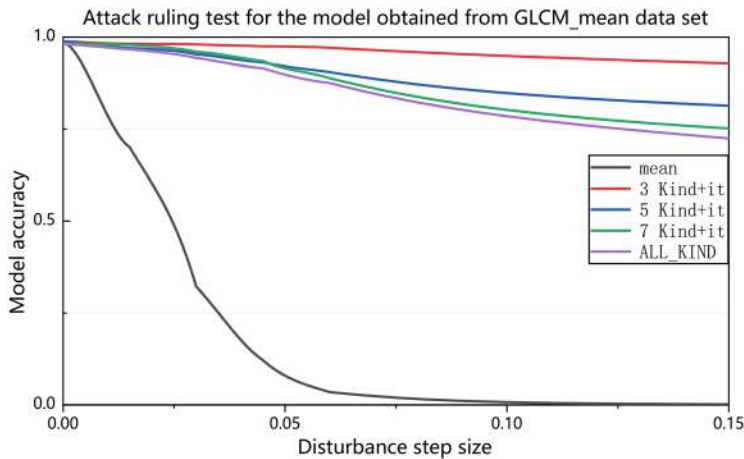


Fig. 8.17 Ruling results of various models built based on different training sets

results obtained by the GLCM algorithm, and the mean value and standard deviation are calculated according to the gray value obtained by the GLCM algorithm, so the calculation method is similar.

On this basis, a simple DHR model is constructed to test the results of multi-model rulings when a single model (with the GLCM_mean building an adversarial sample as an example) is attacked. In Fig. 8.17, the symbol “ n kind+it” indicates that the number of selected models is n plus the model being attacked, and ALL_KIND indicates that all models (11 types) are selected. The results show that the accuracy of the whole system is very high under the condition of low disturbance. With the increase of disturbance, the accuracy decreases slowly, but the overall model is still in a usable state. It is judged that the main cause for the decline in overall accuracy is the weakening of the normal recognition ability of the sub-model after being subjected to greater disturbances, rather than the attack effect that the adversarial sample should show. Therefore, the ESS construction experiment based on feature heterogeneity has achieved the expected effect.

However, the feature heterogeneity of the data set used in the above experiment is based on the premise of abandoning the picture information, which will result in the performance of the sub-model being inferior to the model trained by the original training set. How to ensure that information is not lost or that the overall system performance is not lower than the model trained by the original data set is a problem to be solved in the future research on data feature heterogeneity.

8.2.4.4 Neural Network Model-Integrated Defense Experiment Based on Model Heterogeneity

In addition to the feature heterogeneity of training data sets, the model architecture heterogeneity of neural networks is also a topic worth exploring. In order to build a more differentiated multi-model integration strategy, a feasible heterogeneous network construction method and its effectiveness are preliminarily verified here by using the neural architecture search (NAS) [36]. At the same time, in order to prove that the constructed heterogeneous network can resist the migration of attacks, the lottery hypothesis is used for further analysis and verification.

1. Neural Network Model Heterogenization Method Based on Automatic Neural Architecture Search

In this section, a dynamic random integration strategy is used under the ESS defense framework of AI, that is, each time the model is inferred, sub-models are randomly selected from the model base for integration.

In the experiment, the number of network models, model depth, model width, and connection density are selected as the attribute hyperparameters of the integrated model. For the CIFAR-10 data set, PGD attack is used to observe the defense effect of the integrated model. The attacked deep neural network model is taken as benchmark model A, and model B is an integration under the dynamic defense framework with the deep neural network model as the sub-network model. Considering the impact of adversarial robustness, Robnet [34] is added to the experiment and set as model C. Model D is a single NAS sub-network model, and model E is an integrated model with the NAS model as a sub-network model, which contains the same number of sub-network models as B. In particular, F and E use the same sub-network model base. However, ten heterogeneous network models are sampled from F to evaluate their defense performance, and these sampled heterogeneous models are attacked by adversarial samples generated by heterogeneous integrated networks. By observing the attack success rate, we can judge whether the integrated model can weaken the migration of adversarial examples. In order to control potential probabilistic risks, the Monte Carlo threshold α is set to 0.3 to reduce the probability of erroneous evaluation.

As shown in Fig. 8.18, deep neural networks with standard training are very vulnerable to white box attacks. Under a white box attack, the attack success rate can be quite high when the amount disturbance is low. It can be seen that for deep neural network integration method B and NAS integration method E and F, with the same number of sub-network models, the NAS integration mode requires a higher disturbance intensity to reduce its classification accuracy, and its defense performance is better than other methods. In observation of C and F, the attacker successfully attacked Robnet with a lower disturbance intensity. The heterogeneous network model search method improves the adversarial robustness of the integrated model by enhancing the diversity of sub-network models and increases the vulnerability of the integrated model to white box attacks. Comparing the

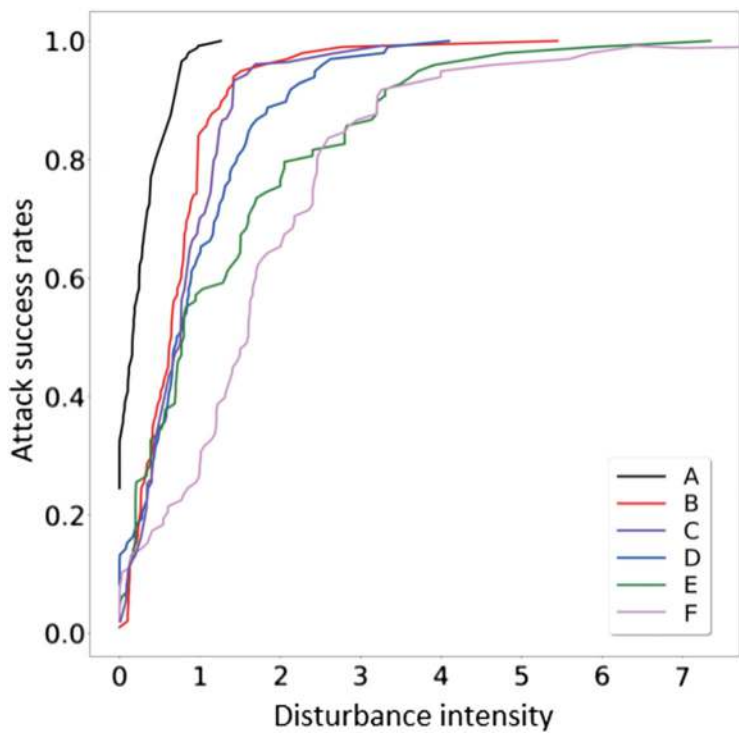


Fig. 8.18 Curves of success rates and disturbance intensity of nontargeted white box attacks based on PGD attacks

defense performance of D and F, this method does provide random gradient information, which is helpful in confusing the gradient direction.

In addition, the experiment also completes the migration evaluation of the adversarial samples generated by each CIFAR-10 model. It is measured by classifying the correct rate of the network model against samples under nontargeted attacks and the success rate of misleading the network model to a specific target tag under targeted attacks. It is measured by the correct rate of classifying the adversarial sample network model under nontargeted attacks, and the success rate of misleading the network model to a specific target label under targeted attacks.

Two sets of models are evaluated here: one consists of three heterogeneous NAS models, and the other consists of three deep neural network models designed by experts (denoted as “Base” models), both adopting PGD as the attack method. The disturbance parameter for both attacks is set to $8/255$. The experiment tests nontargeted attacks and targeted attacks, respectively. The evaluation criteria of the two modes are different, which are, respectively, the classification accuracy of the network model under nontargeted attacks and the attack success rate of predicting target labels by the specific guidance network

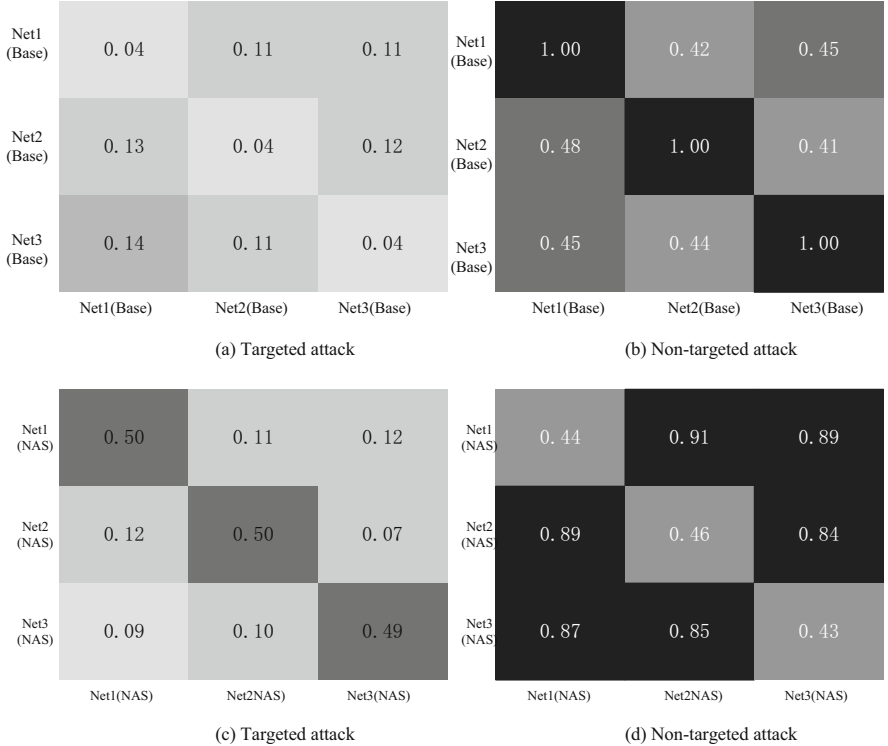


Fig. 8.19 Migratability evaluation of adversarial examples generated by each CIFAR-10 model

model under targeted attacks. Figure 8.19 shows the migratability of the three NAS models and the three deep neural network models. Adversarial examples will affect the performance of the network models, but in both modes, the heterogeneous network models will hinder the migration of adversarial examples between them, effectively verifying that the NAS method provides model heterogeneity for neural networks.

2. Migration Diversity Method of the Neural Network Model Based on Lottery Hypothesis

Heterogeneous search methods empirically improve the architectural diversity of sub-network models. The lottery hypothesis holds that a deep network model usually contains sparse sub-networks with similar accuracy, which is called a lottery network. Fu et al. [37] used pruning in combination with adversarial training to find a lottery sub-network with almost no loss of accuracy but with adversarial robustness. Furthermore, multiple lottery sub-networks with different sparsities in a single network model have the characteristics of poor migratability of adversarial samples and further explore the diversity of sub-network models in adversarial migration under different architectures and sparsities [38].

Four representative network structures are selected for the experiment: ResNet18, ResNet34, WideResNet32, and WideResNet38. Through pruning combined with adversarial training, 40 sub-network models with different basic structures and sparsities are obtained. Each basic structure has 10 sub-networks with different sparsities. The specific performance is shown in Table 8.1.

Figure 8.20 shows the adversarial migratability between lottery sub-network models tested under the same conditions. For fair comparison, ResNet18 and WideResNet32 sub-networks with different sparsities are selected as defense models, and ResNet34 and WideResNet38 sub-networks with the same sparsity are selected to generate adversarial samples. In the experiment, sparsities of 0.07, 0.2, and 0.6 are set for the models. Figure 8.20 mainly shows the robust accuracy achieved by the defense model in generating adversarial samples under different structures.

It can be seen that the ResNet18 and WideResNet32 sub-network models with different sparsities in Fig. 8.20 (a) have poor adversarial migratability against adversarial samples generated using the same network. Compared with Fig. 8.20 (b) and Fig. 8.20 (c), the adversarial migratability of adversarial samples generated by structurally different sub-network models is weaker. For example, except the recognition accuracy on the diagonal, the recognition accuracy of ResNet18 is 65.5–69.9% when generating adversarial samples with the same structure. Facing migration attacks on ResNet34 and WideResNet38 sub-networks, the accuracy increases to 66.5–73.6% and 66.9–70.9%. Similarly, the accuracy of the WideResNet32 sub-network improves by 5.7–9.3% and 2–5.9%.

Under the ESS defense framework of AI, even if the adversarial migration between the above sub-networks is weak, good adversarial robustness can be obtained by means of dynamic random integration. In the experiment, the adversarially trained ResNet18/WideResNet32 dense network is set as the baseline, and the results of dynamic random integration of our method and a single model's sub-networks with different sparsities obtained by Robust2D adversarial sample (R2S) [39] are compared. For better evaluation, the experiment uses an adaptive attack based on expectation over transformation (EOT) [40]. By sampling the gradient of the candidate robust lottery sub-network model, the adversarial samples are generated, and the expectation of defense parameters is obtained, which covers the possibility of more defense strategies and improves the migratability of the adversarial samples. For EOT attacks, the network structure type and sparsity of the lottery sub-network model base are opened to the attacker in the experiment, so that the attacker can sample the expectations for the states of different sets. In order to comprehensively and accurately observe the defense effect, the experiment tested the white box attacks with 0, 2, 4, 8, 12, and 20 disturbances under the constraint of the infinite norm.

As shown in Table 8.2, the robust recognition accuracy of our method is 3.02–10.12% higher than that of R2S, and 15.42% higher than that of adversarial training. Compared with adversarial training, R2S's clean sample recognition accuracy drops by 3.59–3.67%, but the accuracy of our method is 1.08% higher under dynamic integration of different structures. Figure 8.21 also shows that our

Table 8.1 Recognition accuracy of the lottery sub-network model under clean samples and attacks on CIFAR-10

Type of front-end network	Sparsity	Number of sub-network models	Clean sample recognition accuracy	Average clean sample recognition accuracy	Recognition accuracy under attack	Average recognition accuracy under attack
ResNet18	0.07,0.1, 0.12,0.15,	10	76.8~79.8	77.9	45.1~47.3	46.3
ResNet34	0.2,0.3,0.4,0.5,0.6,0.7	10	77.6~80.1	79.1	46.1~48.6	47.6
WideResnet32		10	79.2~82.4	81.1	48.5~49.6	49.1
WideResnet34		10	79.9~83.1	81.9	49.2~50.3	49.7

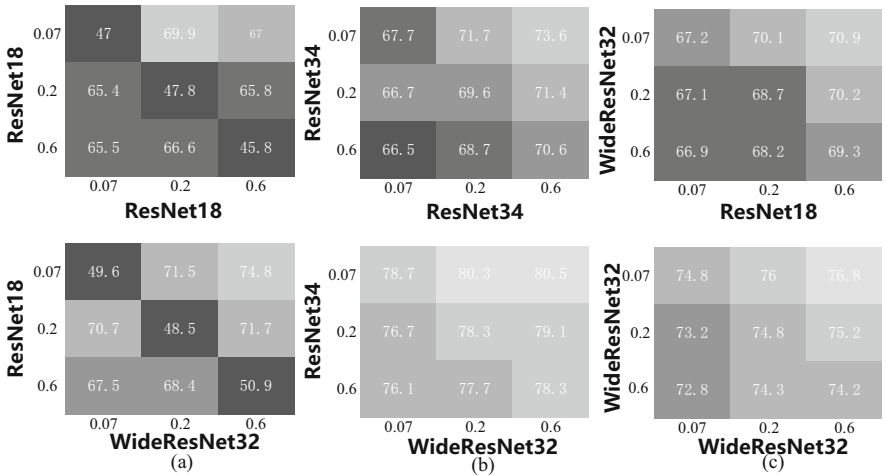


Fig. 8.20 Adversarial migratability between sub-network models: (a) using self-generated adversarial samples to attack, (b) using ResNet34 to attack, (c) using WideResNet38 to attack

Table 8.2 Adversarial robustness tested under EOT attack on CIFAR-10

Network structure	Resnet18	WideResnet32	Resnet18	WideResnet32
Metrics	Clean sample recognition accuracy/%	Clean sample recognition accuracy/%	Adversarial sample recognition accuracy/%	Adversarial sample recognition accuracy/%
Adversarially trained dense networks	81.73	85.93	51.2	52.3
R2S	78.06	82.34	57.6	64.98
Our method	87.01		67.72	

method has better adversarial robustness than R2S and adversarial training in the overall attack environment with multiple disturbances.

8.2.4.5 Verification Platform of the ESS Defense Principle of AI

On the basis of the above experimental analysis, we set up an ESS defense principle verification platform of AI and formed an interface-based experimental operation. The platform allows users to choose different attack methods, attack different pictures, and view the comprehensive defense effect of multiple models in real

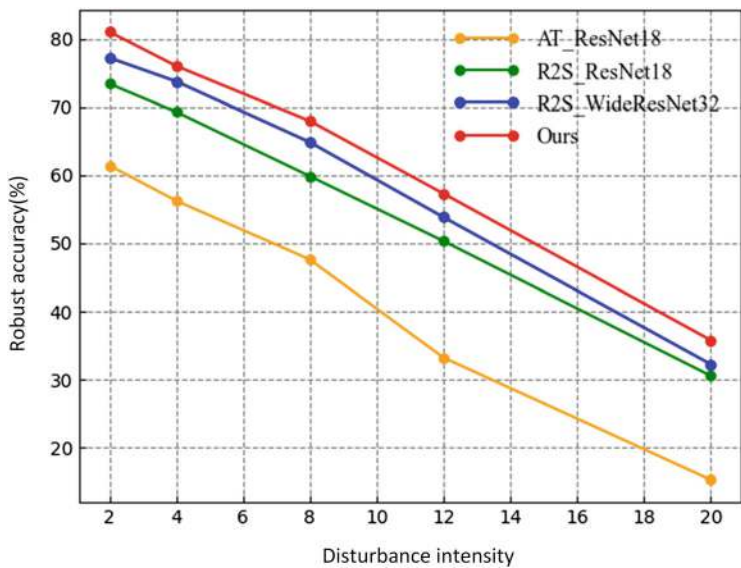


Fig. 8.21 Adversarial robustness against EOT attacks on CIFAR-10

time. The interface effect is shown in Fig. 8.22. In the future, we will continue to improve the function and display interface of the platform; enrich the algorithm library, attack library, and ESS strategy library; and provide end-to-end support for various experiments of AI attack-defense confrontation.

8.3 Performance Analysis of Endogenously Safe and Secure Chips

8.3.1 A Brief Introduction to Endogenously Safe and Secure Chips

According to the abstract model of the DHR architecture, endogenously safe and secure chips mainly include modules such as the reconfigurable executor set, strategic ruling, the feedback control scheduler, and the input agent. This subsection will focus on the design idea of the reconfigurable executor set of endogenously safe and secure chips. For such chips, the realization of reconfigurable executor sets can be divided into hardware-level heterogeneity and software-level heterogeneity.

Hardware heterogeneity is realized by heterogeneous CPUs with different instruction set architectures, such as ARM, RISC-V, MIPS, etc. But a larger number of executors are needed in order to build a more diverse set of reconfigurable executors. However, the types of instruction set architectures of CPUs are often

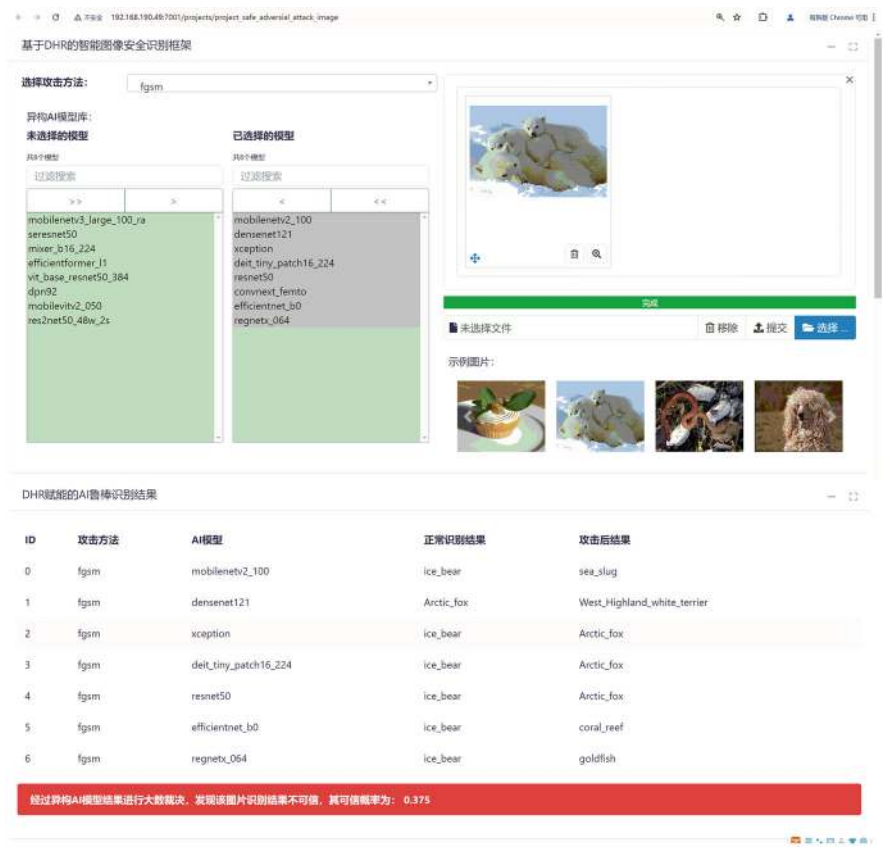


Fig. 8.22 Interface of the verification platform of the ESS defense principle of AI

limited, so heterogeneity at the software level can enrich the reconfigurable executor set of endogenously safe and secure chips.

The concrete realization of software heterogeneity is the introduction of diversified compilation mechanisms. Diversified compilation can achieve code obfuscation, equivalent transformation, randomization at related levels, and stack-related diversification through three different levels: source codes, intermediate codes, and target codes. Many different variants can be generated from the same source codes.

The enriching effect of introducing diversified compilation mechanisms on the reconfigurable executable set can be illustrated by the following example. Assuming there are m heterogeneous CPUs, the diversified compilation mechanism for each CPU can produce n different executable files, and the number of executors per response input is i . When there is no diversified compilation mechanism, there are C_m^i different executor response combinations. When the diversified compilation mechanism is introduced, the number of combinations must be at least $C_m^i \cdot n^i$, or n^i times of the original number. See Table 8.3 for details.

Table 8.3 Examples of a reconfigurable executor set of endogenously safe and secure chips

Hardware heterogeneity	Software heterogeneity
CPU ₁	$S_{11}, S_{12}, S_{13}, \dots, S_{1n}$
CPU ₂	$S_{21}, S_{22}, S_{23}, \dots, S_{2n}$
...	...
CPU _m	$S_{m1}, S_{m2}, S_{m3}, \dots, S_{mn}$

8.3.2 *Uncertainty of the Advantages of Endogenously Safe and Secure Chips in Soft Error Rate Assessment*

In security analysis, compared with other system application examples, a unique and important indicator of endogenously safe and secure chips is the soft error rate (SER). Unlike hard errors caused by defective mechanisms or reliability degradation mechanisms, soft errors generally do not damage the circuit itself, hence called “soft” errors. But they will cause wrong state upset in digital circuits, such as incorrectly changing state 1 to state 0. For a qualified and well-designed circuit, the main cause for soft errors is particle radiation [41].

When passing through the sensitive area of electronic devices, high-energy particles cause abnormal state changes, thus interfering with or even failing their functions. This radiation effect is called the single event effect (SEE), which includes the single effect upset (SEU), the single effect transient (SET), the single effect latch (SEL), and so on. Among them, SEU is a kind of SEE that has the greatest impact on SRAM-based FPGA devices. It shows that high-energy particles change the logic state of the circuit by attacking electronic devices [42].

Although SEU only causes soft errors to occur, it may still have disastrous consequences for the entire system. For example, the FengYun-1B meteorological satellite launched by China in 1990 got out of control due to multiple SEUs, as a result of which, the satellite only operated normally for 285 days and failed to reach its design life [43].

Then, do endogenously safe and secure chips have advantages in the evaluation of the soft error rate? What we know is that in endogenously safe and secure chips, each executor in the reconfigurable executor set is heterogeneous. Compared with isomorphic executors, it not only greatly reduces the impact of common-cause failures but also significantly reduces the reuse rate of attack experience and attack results. Therefore, the heterogeneous executor set exhibits superior functional safety and cybersecurity.

In an environment of particle radiation, the probability of soft errors caused by particle radiation is only related to the capacity of memory devices [44]. The larger the capacity, the greater the probability of being disturbed by radiation (or more precisely, the more bits the memory actually uses, the greater the probability of being disturbed.). However, the probability of radiation disturbance to each memory cell is the same and is related to the anti-disturbance characteristics and quantity of the lowest physical devices but has nothing to do with the upper structure of the system. Therefore, in the evaluation of soft error rates, whether a set of heterogeneous

redundant executors has more advantages than a set of isomorphic redundant executors is a question worth exploring.

Most of the literature reviewed so far takes a single electronic device or a single CPU system as the research object to evaluate the soft error rate caused by SEU. For example, in terms of SEE statistics of a single device, Xilinx provides SEM IP, which can evaluate the impact of SEU events on the system by inserting errors into the configuration memory [45]. Literature [46] and literature [47] conducted bombardment experiments on devices through heavy ion accelerators to obtain SEU. Literature [48] studied SEE of FPGAs of different processes through spaceborne experiments. However, no published literature has been found to evaluate the overall soft error rate of a system composed of isomorphic or heterogeneous redundant executors, so the next section will carry out relevant researches.

8.3.3 Quantitative Analysis of Chip Safety and Security Performance Based on the Markov Model

8.3.3.1 Simplification of a System Architecture

In order to evaluate the probability of soft errors occurring between the DHR architecture and the dynamic architecture in which the reconfigurable executor set is composed of isomorphic executors, we can choose, for SER analysis, some moments when the two dynamic structures are in a stable state and the set of executors is not reconfigured. Without taking into account the differences in other factors such as scheduling strategy and feedback control response time, after comparing and evaluating the static structures at a certain moment, the soft error rates of the two systems in long-term dynamic operation can also be qualitatively evaluated by drawing on the idea of integration.

The DHR architecture in the transient steady state is equivalent to the dissimilar redundant structure (DRS). For the convenience of explanation, the heterogeneous redundant system is used to refer to the static DHR architecture at a certain moment. When in a steady state, the dynamic isomorphic redundant architecture can also be regarded as a static isomorphic redundant system.

The heterogeneous redundant system of MCU chips described in this section will take the prototype design of the mimic security system described in literature [49] as an example. As shown in Fig. 8.23, three CPUs with different instruction set architectures are selected to form a heterogeneous CPU system. They are, respectively, Cortex-M3 with the ARM instruction set architecture, Hummingbird E203 with the RISC-V instruction set architecture, and MicroAptiv with the MIPS instruction set architecture. Multimode ruling is performed on the output of the heterogeneous CPU system through the mimic scheduling unit. Here, it is assumed that the mimic scheduling unit can not only be found by the ruling link when one CPU outputs the wrong result and can give the correct system output but also can identify

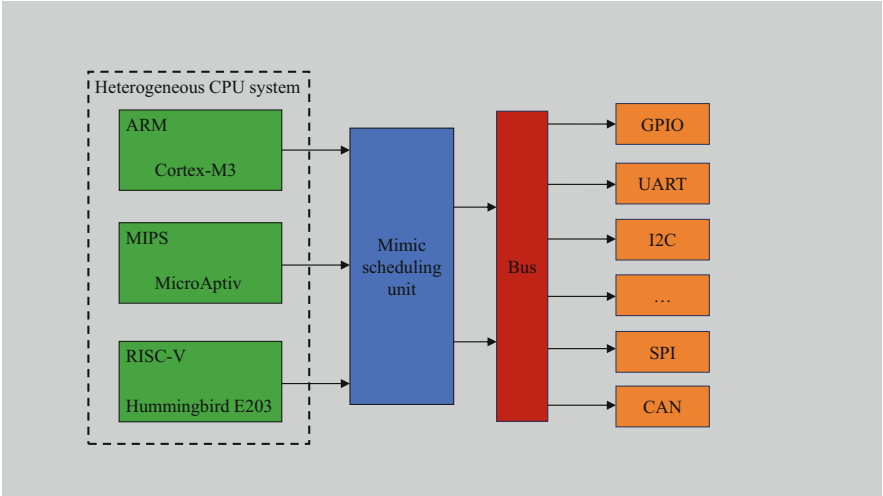
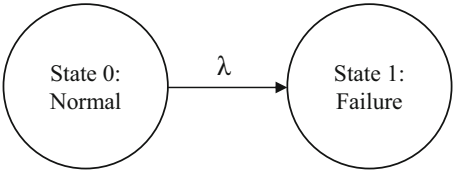


Fig. 8.23 An example of a heterogeneous redundant architecture

Fig. 8.24 Markov model



the only CPU with normal function through iterative ruling when both CPUs fail due to radiation disturbance.

It should be noted that in order to ensure the ruling efficiency of the mimic scheduling unit, the CPU with similar performance is usually selected when selecting a heterogeneous CPU. Because the performance of the Hummingbird E203 is lower than the other two CPUs, the three CPUs mentioned here are not the choice to constitute an ideal heterogeneous CPU system and are only used as examples.

8.3.3.2 An Introduction to the Markov Model

The Markov model is a widely used analysis method in system reliability analysis.

As shown in Fig. 8.24, a simple system consists of two states: normal operation and failure, respectively, represented by state 0 and state 1. The speed of the transition from state 0 to state 1 is λ [50].

We use $P_{j(t)}$ to represent the probability that the system is in state j at time t . The derivative of the probability that the system is in state 0 with respect to time is the

product of the probability $P_{0(t)}$ that the system is in state 0 at time t and the transition rate λ :

$$\frac{dP_0}{dt} = -\lambda P_0 \quad (8.2)$$

Equation (8.2) is the basic formula of the Markov model, which is also the key to reliability analysis of other complex systems. In the same way, the derivative of the probability that the system is in state 1 with respect to time is as follows:

$$\frac{dP_1}{dt} = \lambda P_0 \quad (8.3)$$

It is assumed that in the initial state the system works normally, i.e., $[P_0(0) \ P_1(0)]^T = [1 \ 0]^T$. Solving the above differential equations and combining the constraint $P_{0(t)} + P_{1(t)} = 1$, we can get the following:

$$P_0(t) = e^{-\lambda t}, \quad P_1(t) = 1 - e^{-\lambda t} \quad (8.4)$$

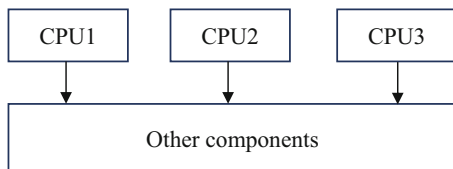
8.3.3.3 Modeling and Calculating of Heterogeneous Redundant Systems

For convenience, CPU1, CPU2, and CPU3 are used to refer to the CPUs with the three instruction set architectures, and the failure rates of them are represented by λ_{c1} , λ_{c2} , and λ_{c3} , respectively. It is assumed that except for the CPU subsystem, all other components in the heterogeneous redundant and isomorphic redundant systems are the same. Therefore, when analyzing the performance difference between the two, the exact failure rate of this part is not the focus of our concern. In order to simplify the analysis process, the components including mimic scheduling unit, bus, and peripherals are represented by other components. The simplified heterogeneous redundant system is shown in Fig. 8.25.

Next, we analyze the possible states of the simplified heterogeneous redundant system, which are denoted by Z_i :

- Z_0 : The system is working normally.
- Z_1 : CPU1 fails, but CPU2, CPU3, and other components are normal.
- Z_2 : CPU2 fails, but CPU1, CPU3, and other components are normal.
- Z_3 : CPU3 fails, but CPU1, CPU2, and other components are normal.

Fig. 8.25 Simplified heterogeneous redundant system architecture



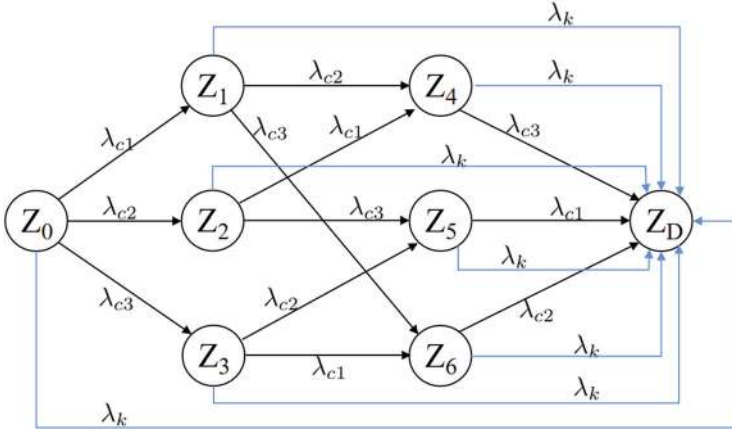


Fig. 8.26 Markov model of heterogeneous redundant systems

- Z_4 : CPU1 and CPU2 fail, but CPU3 and other components are normal.
- Z_5 : CPU2 and CPU3 fail, but CPU1 and other components are normal.
- Z_6 : CPU3 and CPU1 fail, but CPU2 and other components are normal.
- Z_D : Overall system failure.

According to the discussion in the first part of this section, the failure rate of the DHR architecture can be roughly deduced from the failure rate of the DHR architecture at the moment of the transient steady state. Therefore, the scheduling and cleansing functions of the DHR architecture do not need to be taken into account. Precisely because the system's self-repair ability is not considered, the soft error rate of the static architecture at this moment when exposed to particle radiation can be more intuitively reflected. This Markov model obtained without considering the repair ability is called the open-loop Markov model. Compared with the closed-loop model, the open-loop model is more concise and therefore easier to calculate.

Based on the above analysis, the Markov model of the heterogeneous redundant system can be derived, as shown in Fig. 8.26.

Now, we briefly analyze Fig. 8.26, taking the four transfer paths related to state Z_0 as an example. ① When entering state Z_1 , CPU1 fails, and the transfer rate of this path is CPU1's failure rate λ_{c1} ; ② when entering state Z_2 , CPU2 fails, and the transfer rate is λ_{c2} ; ③ when entering state Z_3 , CPU3 fails, and the transfer rate is λ_{c3} ; ④ when entering state Z_D , the state of the system changes from normal operation to overall failure because all other components of the system have failed, so the transfer rate is λ_k . According to the basic Eq. (8.2) of the Markov model, the transient analysis of the system can be obtained:

$$\frac{dP_0}{dt} = -(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k) \cdot P_0 \quad (8.5)$$

$$\frac{dP_1}{dt} = \lambda_{c1} \cdot P_0 - (\lambda_{c2} + \lambda_{c3} + \lambda_k) \cdot P_1 \quad (8.6)$$

$$\frac{dP_2}{dt} = \lambda_{c2} \cdot P_0 - (\lambda_{c1} + \lambda_{c3} + \lambda_k) \cdot P_2 \quad (8.7)$$

$$\frac{dP_3}{dt} = \lambda_{c3} \cdot P_0 - (\lambda_{c1} + \lambda_{c2} + \lambda_k) \cdot P_3 \quad (8.8)$$

$$\frac{dP_4}{dt} = \lambda_{c2} \cdot P_1 + \lambda_{c1} \cdot P_2 - (\lambda_{c3} + \lambda_k) \cdot P_4 \quad (8.9)$$

$$\frac{dP_5}{dt} = \lambda_{c3} \cdot P_2 + \lambda_{c2} \cdot P_3 - (\lambda_{c1} + \lambda_k) \cdot P_5 \quad (8.10)$$

$$\frac{dP_6}{dt} = \lambda_{c3} \cdot P_1 + \lambda_{c1} \cdot P_3 - (\lambda_{c2} + \lambda_k) \cdot P_6 \quad (8.11)$$

Assuming that the initial state of the system is normal, then there is $[P_0(0) P_1(0) P_2(0) P_3(0) P_4(0) P_5(0) P_6(0)]^T = [1 0 0 0 0 0 0]^T$. According to the constraint that the sum of all probabilities is 1, $P_i(t)$ can be obtained by solving the above differential equations, where $i = 0, 1, 2, \dots, 6$. The specific results can be found in the appendix.

In this section, the failure rate of the system is chosen to describe the system's reliability. The failure rate of a transient system is defined as follows [50]:

$$\lambda_{\text{sys}} = \frac{(dP_D/dt)^+}{1 - P_D}, \quad (8.12)$$

wherein $(dP_D/dt)^+$ denotes the rate at which the system transitions from other states to state Z_D . The specific process of calculating the heterogeneous redundant system's failure rate $\lambda_{\text{sys-he}}$ can be found in the appendix. Listed here are only the calculation results:

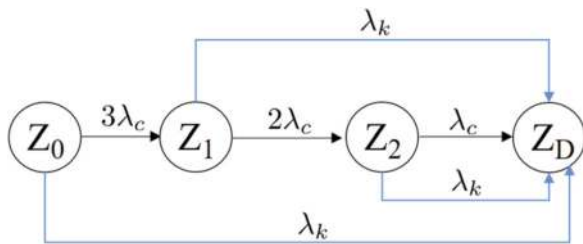
$$\lambda_{\text{sys-he}} = \frac{A_1 \cdot e^{-A_1 t} - B_1 \cdot e^{-B_1 t} - B_2 \cdot e^{-B_2 t} - B_3 \cdot e^{-B_3 t} + C_1 \cdot e^{-C_1 t} + C_2 \cdot e^{-C_2 t} + C_3 \cdot e^{-C_3 t}}{e^{-A_1 t} - e^{-B_1 t} - e^{-B_2 t} - e^{-B_3 t} + e^{-C_1 t} + e^{-C_2 t} + e^{-C_3 t}} \quad (8.13)$$

wherein $A_1 = \lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k$, $B_1 = \lambda_{c2} + \lambda_{c3} + \lambda_k$, $B_2 = \lambda_{c1} + \lambda_{c3} + \lambda_k$, $B_3 = \lambda_{c1} + \lambda_{c2} + \lambda_k$, $C_1 = \lambda_{c3} + \lambda_k$, $C_2 = \lambda_{c1} + \lambda_k$, $C_3 = \lambda_{c2} + \lambda_k$.

8.3.3.4 Modeling and Calculating of Isomorphic Redundant Systems

The analysis of isomorphic redundant systems can be achieved by repeating the steps described in the previous section, that is, drawing the Markov model (Fig. 8.27), performing transient analysis, solving the system of differential equations, etc. But a simpler approach is to simplify the results obtained from heterogeneous redundant

Fig. 8.27 Markov model of isomorphic redundant systems



systems. The only difference between them is that heterogeneous systems use CPUs with different instruction set architectures, while isomorphic systems use the same CPU, so the CPU failure rate is also the same, that is, $\lambda_{c1} = \lambda_{c2} = \lambda_{c3}$. Herein, λ_{c1} , λ_{c2} , and λ_{c3} are collectively represented by λ_c . Except for CPU, the other components of the two types of systems are the same, so the failure rate of the remaining components is still λ_k . By simplifying Eq. (8.13), the failure rate $\lambda_{\text{sys-ho}}$ of isomorphic redundant systems is as follows:

$$\lambda_{\text{sys-ho}} = \frac{A \cdot e^{-At} - 3B \cdot e^{-Bt} + 3C \cdot e^{-Ct}}{e^{-At} - 3e^{-Bt} + 3e^{-Ct}} \quad (8.14)$$

wherein $A = 3\lambda_c + \lambda_k$, $B = 2\lambda_c + \lambda_k$, $C = \lambda_c + \lambda_k$.

8.3.3.5 Analysis of Results

So far, the overall failure rates of isomorphic and heterogeneous redundant systems have been deduced. Equations (8.13) and (8.14) will be analyzed next.

Firstly, assuming that the failure rates of the three CPUs are $\lambda_{c1} < \lambda_{c2} < \lambda_{c3}$, we can get $C_2 < C_3 < C_1 < B_3 < B_2 < B_1 < A_1$. Let's find the limit of Eq. (8.13):

$$\lim_{t \rightarrow +\infty} \lambda_{\text{sys-he}} = \lim_{t \rightarrow +\infty} \frac{C_2 \cdot e^{-C_2 t}}{e^{-C_2 t}} = \lim_{t \rightarrow +\infty} \frac{(\lambda_{c1} + \lambda_k) \cdot e^{-(\lambda_{c1} + \lambda_k)t}}{e^{-(\lambda_{c1} + \lambda_k)t}} = \lambda_{c1} + \lambda_k \quad (8.15)$$

It is found that $\lambda_{\text{sys-he}}$ approaches the limit value $\lambda_{c1} + \lambda_k$ with time. From this we can get the following propositions:

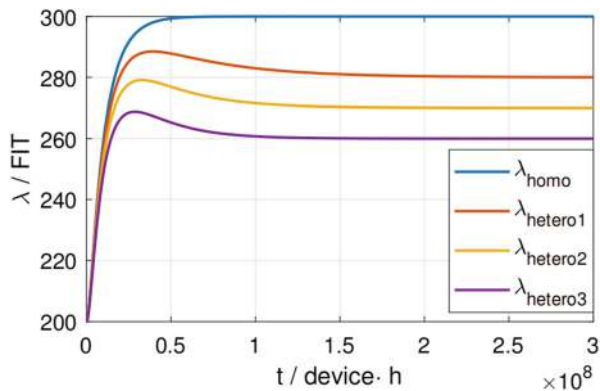
Proposition 8.1. The failure rate of a heterogeneous redundant system increases over time toward the sum of the failure rate of the CPU that has the smallest failure rate and the failure rates of other components.

In the same way, it can be deduced that for an isomorphic redundant system, $\lambda_{\text{sys-ho}}$ approaches the limit value $\lambda_c + \lambda_k$ with time.

Proposition 8.2. If the total failure rates of the three CPUs in the heterogeneous redundant system and the isomorphic redundant system are equal and the differences

Table 8.4 Assigning values to failure rates λ_{ci}

	Examples	Failure rates (FIT/Mb)
Isomorphic redundant system		$\lambda_c = 100$
Heterogeneous redundant system	1	$\lambda_{c1} = 100, \lambda_{c2} = 100, \lambda_{c3} = 120$
	2	$\lambda_{c1} = 70, \lambda_{c2} = 100, \lambda_{c3} = 130$
	3	$\lambda_{c1} = 60, \lambda_{c2} = 100, \lambda_{c3} = 140$

Fig. 8.28 Failure rate analysis of the triple modular redundancy isomorphic system and the triple modular redundancy heterogeneous system in Table 8.4

in the failure rates of the three CPUs in the heterogeneous redundant system are equal, that is, if λ_{c1} , λ_{c2} , λ_{c3} , and λ_c satisfy the following conditions:

$$\lambda_{c1} + \lambda_{c2} + \lambda_{c3} = 3\lambda_c \quad (8.16)$$

$$\lambda_{c2} - \lambda_{c1} = \lambda_{c3} - \lambda_{c2} = \Delta\lambda > 0 \quad (8.17)$$

then the overall failure rate of the heterogeneous redundant system is smaller than that of the isomorphic redundant system, that is:

$$\lambda_{\text{sys-he}} < \lambda_{\text{sys-ho}} \quad (8.18)$$

According to the conditions required by Proposition 8.2, we assign values to λ_{c1} , λ_{c2} , and λ_{c3} , as shown in Table 8.4.

Because the other components of the redundant system have not yet been determined, the value of λ_k should be within a reasonable range. Here we take $\lambda_c = 200$ FIT/Mb. We substitute the above data into Eqs. (8.13) and (8.14) for calculation and plot the obtained results of $\lambda_{\text{sys-ho}}$ and $\lambda_{\text{sys-he}}$, as shown in Fig. 8.28. The results are consistent with Proposition 8.1, that is, the overall failure rate of heterogeneous redundant systems is smaller than that of isomorphic redundant systems. Figure 8.28 also shows what is stated in Proposition 8.2: the overall failure rate of each system eventually approaches the sum of the failure rate of the CPU that has the smallest failure rate and the failure rates of the other components, that is, for a heterogeneous redundant system, it approaches $\lambda_{c1} + \lambda_k$, while for an isomorphic

redundant system, it approaches $\lambda_c + \lambda_k$. According to $\lambda_{\text{hetero1}} > \lambda_{\text{hetero2}} > \lambda_{\text{hetero3}}$, it can be seen that under the condition that Proposition 8.2 is satisfied, the larger λ is, the smaller the system failure rate will be. This conclusion can be proved by Proposition 8.1.

So far, the theoretical deduction of the failure rate of redundant systems based on the Markov model and the analysis of the characteristic of the conclusions have been completed, and the results of the theoretical deduction are intuitively displayed by assigning arbitrary values that meet the conditions for plotting. However, how to obtain true or objective λ_{ci} and λ_k is still an urgent problem to be solved, which will be elaborated in the next subsection.

8.3.4 *Simulation and Analysis of Anti-SEU Capabilities of Chips with Different Architectures*

When evaluating the SER caused by the single effect upset (SEU) for a certain device (i.e., an incomplete system), two steps are required [51].

First, the physical characteristics of the device need to be clarified. For SRAM-type FPGA, the “device” here refers to configuration memory, BRAM, DSP, etc. The “physical characteristics” here refer to the static SEU cross section of the device, which describes the sensitivity of the device to particle radiation. The main methods to obtain the static SEU cross section include ① conducting irradiation experiments on the device through a heavy ion accelerator, as shown in literature [52], ② observing the SEU of the device under actual cosmic radiation conditions through spaceborne experiments (the above two methods can be used to obtain the SEU cross section of any device, but the disadvantage is that the observation period is long and the cost is high), and ③ obtaining the SEU cross section through simulation, such as using CREME-MC for simulation [53]. Literature [38] demonstrates the specific method of using CREME-MC to obtain the SEU cross section of a SEU-reinforced SRAM.

Second, it is necessary to use a modeling and simulation tool to evaluate the SER caused by SEU according to the physical characteristics of the device and the radiation environment. We use the CREME96 simulation tool to evaluate the SER caused by SEU in each CPU in heterogeneous and isomorphic redundant systems. CREME96 has been updated based on CREME (Cosmic Ray Effects on Micro-Electronics), and some improvements have been made in the modeling of cosmic rays, the composition of solar particles in the near-Earth environment, and the calculation method of SEU [54].

8.3.4.1 Obtaining the Static SEU Cross Section

The static SEU cross-section $\sigma(L)$ of a device is a characteristic parameter of the device provided by the user when doing simulation using CREME96. The Weibull curve is usually used for fitting:

$$\sigma(L) = \sigma_{\text{sat}} (1 - \exp\{-[(L - L_{\text{th}})/W]^S\}) \quad (8.19)$$

wherein σ_{sat} represents the saturation cross section of the device, L_{th} represents the critical LET value that causes SEU of the device, W represents the width factor, and S is a dimensionless index [55].

Because how to obtain the SEU cross section of a device is not the focus of the research here, this subsection refers to literature [56] for the convenience of subsequent discussion. In this literature, the Xilinx Kintex UltraScale series FPGA development boards were irradiated in a heavy ion accelerator, and the Weibull fitting parameters of the SEU cross section of each device were obtained. We chose this series of FPGA development boards as the device carrier for system operation and converted the experimental results in literature [57] to get the Weibull fitting parameters of the SEU cross section per bit, as shown in Table 8.5.

8.3.4.2 In-Orbit Radiation Conditions

Assuming that the device operates in a low-Earth orbit (LEO), in order to obtain the SEE caused by direct ionization of heavy ions, the simulation needs to go through several modules in sequence, namely, GTRN, FLUX, TRANS, LETSPEC, and HUP. The specific function, input, and output of each module are detailed in the instructions for use of CREME96 [57]. The parameter selection of each module is shown in Table 8.6.

Among memory types, configuration memory refers to the storage unit used to configure system functions, but only a small part of the bits in the configuration memory is critical to the normal operation of the system. When these bits undergo unplanned changes, the system may not perform as intended. Such bits are called essential bits. For Xilinx Kintex UltraScale series FPGAs, Vivado can directly calculate the number and proportion of essential bits used in a certain design. Taking the heterogeneous CPU system as an example, the statistical results are shown in Table 8.7.

Table 8.5 Weibull fitting parameters of the SEU cross section per bit

Weibull parameters	Numerical values
σ_{sat}	$0.2911 \mu\text{m}^2$
L_{th}	$0.07 \text{ MeV cm}^2/\text{mg}$
W	$13 \text{ MeV cm}^2/\text{mg}$
S	1.5

Table 8.6 CREME variables

CREME modules	Description	Numerical values
GTNR	Around the earth	51.6 deg, 450 km
	Magnetic weather conditions	Quiet
FLUX	Atomicity	1~92
	Solar conditions	Smallest
TRANS	Shield	Aluminum, 100 mils
LETSPEC	Minimum energy	0.1 MeV/nuc
HUP	RPP	$x = y = \sqrt{\sigma_{\text{sat}}}$, $z = x/5$

Table 8.7 The number and proportion of essential bits in three CPUs

CPU	Number of essential bits	Proportion/%
Hummingbird E203	2,528,749	2.45
Cortex-M3	2,844,896	2.76
MicroAptiv	3,040,701	2.95

Table 8.8 SEE results for three CPUs

CPUs	Failure rates	R_{orbit} (SEE/(device/day))
Hummingbird E203	λ_{c1}	1.66333E - 02
Cortex-M3	λ_{c2}	1.87129E - 02
MicroAptiv	λ_{c3}	2.00008E - 02

According to the essential bits occupied by each CPU, the in-orbit SEU of each CPU can be evaluated by the HUP module, and the results are shown in Table 8.8. The obtained result R_{orbit} is λ_{c1} , λ_{c2} , and λ_{c3} , and its unit SEE/(device/day) represents the SEE that occurs per device per day.

Substituting λ_{c1} , λ_{c2} , and λ_{c3} into Eq. (8.12) and taking $\lambda_k = 10^{-2}\text{SEE}/(\text{device}/\text{day})$, we obtain the failure rate of heterogeneous redundant systems as a function of time. By substituting λ_{c1} into Eq. (8.13), we obtain the failure rate of an isomorphic redundant system composed of three Hummingbird E203 CPUs as a function of time. Similarly, by substituting λ_{c2} and λ_{c3} into Eq. (8.13), the failure rate of an isomorphic redundant system composed of three Cortex-M3 and three MicroAptiv CPUs can be obtained as a function of time. The above results are plotted to obtain Fig. 8.29.

By analyzing Fig. 8.29, it can be found that the overall failure rate of the heterogeneous redundant system is smaller than the three-mode isomorphic redundant system composed of Cortex-M3 and MicroAptiv, respectively. And as time goes by, it gradually approaches the failure rate of the isomorphic redundant system composed of the CPUs with the smallest failure rate, namely, Hummingbird E203. This result is consistent with Proposition 8.1. The overall failure rate of the heterogeneous redundant system eventually approaches $\lambda_{c1} + \lambda_k$, and this value is also the ultimate trend value of the overall failure rate of the isomorphic redundant system composed of three Hummingbird E203 CPUs. In this example, when $t \geq 1100\text{d}$, the

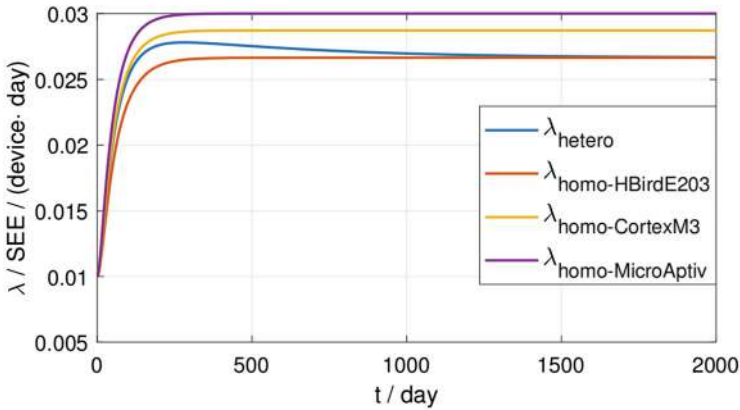


Fig. 8.29 SEE results of one heterogeneous redundant and three isomorphic redundant systems

difference in failure rate between the heterogeneous redundant system and the isomorphic redundant system composed of Hummingbird E203 is $<1\%$.

According to the above analysis and in terms of long-term effects, in particle radiation, the SER manifested by the heterogeneous redundant system is equivalent to the smallest SER of the isomorphic redundant system, and it is superior to the other two isomorphic redundant systems at every moment. Considering the advantages of heterogeneous redundant systems over isomorphic redundant systems in avoiding common-mode failures and reducing attack experience and reproducibility of attack results, we can conclude that heterogeneous redundant systems have better functional safety features.

If every instantaneous static architecture of dynamic architecture A is better than the static architecture of dynamic architecture B in SER evaluation, then without considering the differences in other factors such as scheduling strategies, feedback control response time, etc., dynamic architecture A will show better anti-SER characteristics than dynamic architecture B. In other words, the DHR architecture has better SER resistance than the dynamic isomorphic redundant architecture.

8.4 Summary of This Chapter

This chapter discusses the application prospects of ESS as a “stone from another mountain” in emerging fields such as wireless communication, AI, and chip design. Firstly, on the basis of reviewing the development paradigm of wireless communication and sorting out the ESS problems of wireless communication, the development vision of endogenously safe and secure wireless communication is put forward, including the ESS architecture, mechanisms, key technologies, and performance analysis of wireless communication. Secondly, on the basis of analyzing the common ESS problems, individual ESS problems, and GFS problems of artificial

intelligence, a preliminary design scheme for the endogenously safe and secure defense framework of AI is given. Finally, based on a preliminary introduction to the design ideas of endogenously safe and secure chips, the powerful anti-SEU ability of endogenously safe and secure chips is analyzed. Relevant research shows that the ESS architecture not only has an intrinsic empowering effect but also has a significant spillover effect, strong permeability, and extensive guiding significance.

Appendix

The calculation process of $\lambda_{\text{sys-he}}$ in Sect. 8.3.3 is as follows:

$P_i(t)$ is as follows:

$$\begin{aligned}
 P_0(t) &= e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} \\
 P_1(t) &= -e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} + e^{-(\lambda_{c2} + \lambda_{c3} + \lambda_k)t} \\
 P_2(t) &= -e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} + e^{-(\lambda_{c1} + \lambda_{c3} + \lambda_k)t} \\
 P_3(t) &= -e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} + e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_k)t} \\
 P_4(t) &= e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} - e^{-(\lambda_{c2} + \lambda_{c3} + \lambda_k)t} - e^{-(\lambda_{c1} + \lambda_{c3} + \lambda_k)t} + e^{-(\lambda_{c3} + \lambda_k)t} \\
 P_5(t) &= e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} - e^{-(\lambda_{c1} + \lambda_{c3} + \lambda_k)t} - e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_k)t} + e^{-(\lambda_{c1} + \lambda_k)t} \\
 P_6(t) &= e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k)t} - e^{-(\lambda_{c1} + \lambda_{c2} + \lambda_k)t} - e^{-(\lambda_{c2} + \lambda_{c3} + \lambda_k)t} + e^{-(\lambda_{c2} + \lambda_k)t} \\
 \left(\frac{dP_D}{dt}\right)^+ &= \frac{dP_D}{dt} = \lambda_k \cdot P_0 + \lambda_k \cdot P_1 + \lambda_k \cdot P_2 + \lambda_k \cdot P_3 + (\lambda_{c3} + \lambda_k) \cdot P_4 \\
 &\quad + (\lambda_{c1} + \lambda_k) \cdot P_5 + (\lambda_{c2} + \lambda_k) \cdot P_6
 \end{aligned}$$

Substitute the calculation result of $P_i(t)$ and let $A_1 = \lambda_{c1} + \lambda_{c2} + \lambda_{c3} + \lambda_k$, $B_1 = \lambda_{c2} + \lambda_{c3} + \lambda_k$, $B_2 = \lambda_{c1} + \lambda_{c3} + \lambda_k$, $B_3 = \lambda_{c1} + \lambda_{c2} + \lambda_k$, $C_1 = \lambda_{c3} + \lambda_k$, $C_2 = \lambda_{c1} + \lambda_k$, $C_3 = \lambda_{c2} + \lambda_k$

$$\begin{aligned}
 \left(\frac{dP_D}{dt}\right)^+ &= A_1 \cdot e^{-A_1 t} - B_1 \cdot e^{-B_1 t} - B_2 \cdot e^{-B_2 t} - B_3 \cdot e^{-B_3 t} + C_1 \cdot e^{-C_1 t} + C_2 \cdot e^{-C_2 t} \\
 &\quad + C_3 \cdot e^{-C_3 t}
 \end{aligned}$$

Because $P_D(0) = 0$, $P_D(t)$ is obtained as follows:

$$P_D(t) = -e^{-A_1 t} + e^{-B_1 t} + e^{-B_2 t} + e^{-B_3 t} - e^{-C_1 t} - e^{-C_2 t} - e^{-C_3 t} + 1$$

Therefore, from Eq. (8.12), the final result of $\lambda_{\text{sys-he}}$ can be obtained as Eq. (8.13).

References

1. Shannon CE. A mathematical theory of communication [J]. *Bell Syst Tech J*. 1948;27(3): 379–423.
2. Wang L, Xie S. Radio Spectrum management policy, regulation and technology [M]. Beijing: Electronic Industry Press; 2018.
3. You XH, Wang CX, Huang J, et al. Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts [J]. *Sci China Inf Sci*. 2020;64(1):1–74.
4. Dunkelman O, Keller N, Shamir A. A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony [J]. *J Cryptol*. 2014;27(4):824–49.
5. 3GPP. Security architecture and procedures for 5G system (release 15): 3GPP TS 33. 501 V15. 5.0 [S]. European Telecommunications Standards Institute. 2019.
6. Furqan HM, Hamamreh JM, Arslan H. New physical layer key generation dimensions: subcarrier indices/positions-based key generation [J]. *IEEE Commun Lett*. 2021;25(1):59–63.
7. Chen SL, Pang ZB, Wen H, et al. Automated labeling and learning for physical layer authentication against clone node and sybil attacks in industrial wireless edge networks [J]. *IEEE Trans Industr Inform*. 2020;17(3):2041–51.
8. Zhang CW, Yue J, Jiao LB, et al. A novel physical layer encryption algorithm for LoRa [J]. *IEEE Commun Lett*. 2021;25(8):2512–6.
9. Jiangxing W. Development paradigms of cyberspace endogenous safety and security [J]. *Scientia Sinica (Inf Sci)*. 2022;52(2):189–204.
10. Jin L, Hu XY, Lou YM, et al. Introduction to wireless endogenous security and safety: problems, attributes, structures and functions [J]. *China Commun*. 2021;18(9):88–99.
11. Chen C, Jensen MA. Secret key establishment using temporally and spatially correlated wireless channel coefficients [J]. *IEEE Trans Mob Comput*. 2010;10(2):205–15.
12. Wang X, Jin L, Huang K. Physical layer secret key capacity using correlated wireless channel samples [C]. Paper presented at: IEEE Global Communications Conference (GLOBECOM); 2016. p. 1–6.
13. Qin D, Ding Z. Exploiting multi-antenna nonreciprocal channels for shared secret key generation [J]. *IEEE Trans Inf Forensics Secur*. 2016;11(22):2693–705.
14. Zhihao Z, Wenyu L, Jianhua P, et al. Security performance analysis of cooperative transmission and cooperative interference mechanism in multi-layer heterogeneous cellular networks [J]. *Scientia Sinica (Inf Sci)*. 2016;1:33–48.
15. Yangming L, Liang J, Zhou Z, et al. Secret key generation scheme based on MIMO received signal spaces [J]. *Scientia Sinica (Inf Sci)*. 2017;47(3):362–73.
16. Jin L, Wang X, Lou YM, et al. Achieving one-time pad via endogenous secret key in wireless communication [C]. Paper presented at: IEEE International Conference on Communications, Chongqing, China; 2020. p. 1–6.
17. Liang YC, Chen J, Long R, et al. Reconfigurable intelligent surfaces for smart wireless environments: channel estimation, system design and applications in 6G networks [J]. *SCIENCE CHINA Inf Sci*. 2021;64(10):1–21.
18. Hu X, Jin L, Huang K, et al. Intelligent reflecting surface-assisted secret key generation with discrete phase shifts in static environment [J]. *IEEE Wirel Commun Lett*. 2021;10(9):1867–70.
19. Madhow U, Honig ML. MMSE interference suppression for direct-sequence spread-spectrum CDMA [J]. *IEEE Trans Commun*. 1994;42(12):3178–88.
20. Adem N, Hamdaoui B, Yavuz A. Pseudorandom time-hopping anti-jamming technique for mobile cognitive users [C]. Paper presented at: IEEE Globecom Workshops; 2015. p. 1–6.
21. Suxia G, Xiangyu L, Liang J, et al. An anti-jamming DOA estimation method with space hopping [J]. *Scientia Sinica (Inf Sci)*. 2016;46(7):899–912.
22. Hanawal MK, Abdel-Rahman MJ, Krunz M. Joint adaptation of frequency hopping and transmission rate for anti-jamming wireless systems [J]. *IEEE Trans Mob Comput*. 2016;15(9):2247–59.

23. Jin L, Lou YM, Xu XM, et al. Separating multi-stream signals based on space-time isomerism [C]. Paper presented at: 2020 International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing; 2020. p. 418–423.
24. Wei L, Bin C, Jibo W, et al. Secure communications via sending artificial noise by the receiver: ergodic secure region analysis [J]. *Signal Process.* 2012;28:1314–20.
25. Document of the State Council. Notice of the State Council on printing and distributing the development plan of a new generation of artificial intelligence [EB/OL]. 2017. http://www.gov.cn/zhengce/content/2017-07/20/content_5211996.htm
26. Naghibijouybari H, Neupane A, Qian Z, et al. Rendered insecure: GPU side channel attacks are practical [C]. Paper presented at: the ACM SIGSAC Conference; 2018. p. 2139–53.
27. Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks [J]. *Comput Sci.* 2013;4:1312.
28. Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples [J]. *arXiv preprint arXiv:1412.6572*; 2014.
29. Kurakin A, Goodfellow I, Bengio S. Adversarial machine learning at Scale [J]. *arXiv preprint arXiv:1611.01236*; 2016.
30. Madry A, Makelov A, Schmidt L, et al. Towards deep learning models resistant to adversarial attacks [J]. *arXiv preprint arXiv: 1706.06083*; 2017.
31. Moosavi-Dezfooli SM, Fawzi A, Frossard P. DeepFool: a simple and accurate method to fool deep neural networks [C]. Paper presented at: IEEE CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 2574–2582.
32. Carlini N, Wagner D. Towards evaluating the robustness of neural networks [C]. Paper presented at: 2017 IEEE Symposium on Security and Privacy (SP); 2017. p. 39–57.
33. Li Y, Li L, Wang L, et al. Nattack: learning the distributions of adversarial examples for an improved black-box attack on deep neural networks [C]. Paper presented at: International Conference on Machine Learning, PMLR; 2019. p. 3866–3876.
34. Ilyas, A, Santurkar, S, Tsipras, D, et al. Adversarial examples are not bugs, they are features [C]. Paper presented at: Proceedings of the 33rd International Conference on Neural Information Processing Systems; 2019. p. 125–136.
35. Liu Y, Chen X, Liu C, et al. Delving into transferable adversarial examples and black-box attacks [J]. *arXiv preprint arXiv: 1611.02770*; 2016.
36. Peng Q, Qin RX, Liu WL, et al. Heterogeneous architecture search approach within adversarial dynamic defense framework [C]. Paper presented at: The AAAI-22 Workshop on Adversarial Machine Learning and Beyond; 2022. p. 1–6.
37. Fu YG, Yu QX, Zhang Y, et al. Drawing robust scratch tickets: subnetworks with inborn robustness are found within randomly initialized networks [J]. *Adv Neural Inf Proces Syst.* 2021;34:13059–72.
38. Warren K, Weller R, Mendenhall M, et al. The contribution of nuclear reactions to heavy ion single event upset cross-section measurements in a high-density SEU hardened SRAM [J]. *IEEE Trans Nucl Sci.* 2005;52(6):2125–31.
39. Athalye A, Engstrom L, Ilyas A, et al. Synthesizing robust adversarial examples [C]. Paper presented at: International Conference on Machine Learning, PMLR; 2018. p. 284–293.
40. Peng Q, Liu WL, Qin RX, et al. Dynamic stochastic ensemble with adversarial robust lottery ticket subnetworks [J]. *arXiv preprint arXiv: 2210.02618*; 2022.
41. Texas Instruments. Answers to frequently asked questions about soft error rate [EB/OL]. 2022. <https://www.ti.com.cn/zh-cn/support-quality/faqs/soft-error-rate-faqs.html>
42. Quinn HM, Graham PS, Wirthlin MJ, et al. A test methodology for determining space readiness of Xilinx SRAM-based FPGA devices and designs [J]. *IEEE Trans Instrum Meas.* 2009;58(10): 3380–95.
43. Shifen G. Study on SEU occurred on board of several space shuttles [J]. *Chin J Space Sci.* 1998;18(3):8.
44. User Guide 116. Device reliability report v10.13 [Z]. Xilinx, Inc; 2020. p. 36.
45. SUITE V D. Ultrascale architecture soft error mitigation controller v3.1 [Z]. Xilinx, Inc; 2021.

46. Zhongming W, Zhibin Y, Hongxia G, et al. Static and dynamic tests of single-event effect in SRAM-based FPGA [J]. *At Energy Sci Technol*. 2011;45(12):1506–10.
47. Zhibin Y, Ruyu F, Hongxia G, et al. Acquisition and classification of static single-event upset cross section for SRAM-based FPGAs [J]. *High Power Laser Part Beams*. 2011;23(3):811–6.
48. Lesea A, Drimer S, Fabula J, et al. The Rosetta experiment: atmospheric soft error rate testing in differing technology FPGAs [J]. *IEEE Trans Device Mater Reliab*. 2005;5(3):317–28.
49. Yuanhang S, Li Y, Zhaozhao L, et al. Prototype design of intelligent transportation mimic security chip system [C]. Paper presented at: The 3rd Conference on Advanced Computing and Endogenous Security, Nanjing; 2020. p. 677–684.
50. Introduction to Markov modeling for reliability [EB/OL]. 2022. <https://www.mathpages.com/home/kmath232/kmath-232.html>
51. Engel J, Morgan KS, Wirthlin MJ, Graham PS. Predicting on-orbit static single event upset rates in Xilinx Virtex FPGAs [R]. Faculty Publications; 2006. p. 1–66.
52. Bo W, Jia W, Weixin L, et al. Single event effect detection and on-orbit upset rate estimation for Spaceborne ASIC chip [J]. *Semicond Technol*. 2019;44(9):728–34.
53. Adams JH, Barghouty AF, Mendenhall MH, et al. CREME: the 2011 revision of the cosmic ray effects on micro-electronics code [J]. *IEEE Trans Nucl Sci*. 2012;59(6):3141–7.
54. Tylka AJ, Adams JH, Boberg PR, et al. CREME96: a revision of the cosmic ray effects on micro-electronics code [J]. *IEEE Trans Nucl Sci*. 1997;44(6):2150–60.
55. Allen G. Virtex-4vq static SEU characterization summary [R]. Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration; 2008.
56. Berg M, Kim H, Phan A. Xilinx Kintex-Ultrascale field programmable gate array single event effects (SEE) heavy-ion test report [R]. Space R2 LLC, SSAI, NASA GSFC; 2019.
57. Sierawski B. How to run CREME96 [EB/OL]. 2010. <https://creme.isde.vanderbilt.edu/CREME-MC/help/how-to-run-creme96>