

Zero Trust Security Engineering

A Zero Trust approach to engineering security in your products



Mahesh Patil



Zero Trust Security Engineering

A Zero Trust approach to engineering security in your products



Mahesh Patil



Zero Trust Security Engineering

*A Zero Trust approach to
engineering security in your products*

Mahesh Patil



www.bpbonline.com

First Edition 2026

Copyright © BPB Publications, India

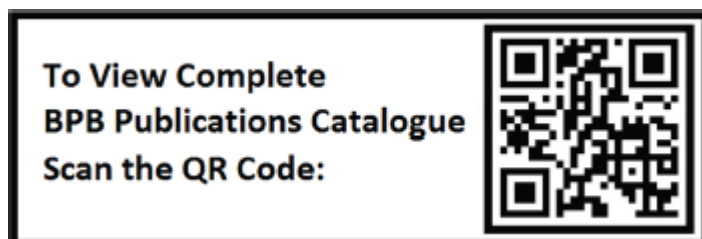
ISBN: 978-93-65894-370

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.



www.bpbonline.com

Dedicated to

*My father, who proved that a great teacher
does not just educate but transforms lives.*

*May his spirit of learning live in every
engineer and product leader who reads this.*

About the Author

Mahesh Patil has 23+ years of experience in software engineering and technology leadership and delivery. He has a background in software engineering and has served as principal architect, director of technology, and CTO. He specializes in cloud platforms, security, reliability, and digital transformation. He has diverse experience working across various technologies within regulated industries such as banking, wealth and asset management, insurance, and health; and has global working experience across the United States, Europe, and India. Currently leading financial services and health-focused cloud solutions, driving innovation by modernizing legacy systems and enabling cloud-native applications.

Passionate about product strategy, secure architectures, and resilient cloud-native solutions, Mahesh has a proven track record of building and scaling reliable SaaS products for B2B enterprises. His expertise spans product management, cloud technologies, cybersecurity, and high-availability and reliable system design, making him a sought-after leader in the tech industry.

An advocate for continuous learning and knowledge sharing, Mahesh actively engages in tech forums and industry discussions, contributing valuable insights on cloud computing, security best practices, system reliability, and modern software development frameworks. This book reflects his deep commitment to helping teams build secure, scalable, and dependable software solutions.

About the Reviewer

Surya Prakash Nalluri is a globally recognized cybersecurity executive, researcher, and innovator with more than two decades of experience spanning secure software engineering, application security, DevSecOps, and cyber risk governance. Beginning his career as a Java developer, he quickly advanced into security architecture—remediating complex vulnerabilities, designing a scalable CSRF protection framework for legacy banking systems, and helping launch early mobile thin-client applications during the first wave of mobile banking.

Now serving as a senior vice president at Citigroup, he leads enterprise-wide security programs, including the global **malicious code detection (MCD)** initiative, overseeing thousands of production code reviews to identify backdoors and insider-threat patterns. He manages geographically distributed teams, drives security transformation initiatives, and provides strategic oversight across architecture, governance, tool modernization, and operational excellence.

Surya is the inventor of multiple patented cybersecurity innovations, including the award-winning security bot automation framework (US11763007) and international patents in AI-enhanced data governance, quantum-resilient security, and federated security orchestration. He authored the widely adopted **Damn Vulnerable Thick Client Application (DVTA)** and OSSPolicyGuard, addressing open-source supply chain and protestware risks.

A prolific researcher, he has published peer-reviewed journal papers, SCOPUS-indexed articles, and book chapters in cybersecurity, AI, and privacy. He also contributes as an editorial board member and reviewer for multiple international journals in cybersecurity, digital forensics, and AI.

Surya plays an active leadership role in the global professional community—serving as IEEE Privacy Task Force Chair, contributing to ISACA

COBIT 2025, and judging major global awards including Globee, Stevie, and Webby. He is a dedicated mentor supporting emerging cybersecurity professionals.

Beyond his professional contributions, he enjoys chess, drawing, music, and reviewing technology publications.

Acknowledgement

This book would not have been possible without the unwavering support and encouragement from my mother, wife, son, and my sister. I want to express my deepest gratitude to them for helping me dedicate all the time required for this project and for keeping me motivated throughout.

A long journey that began with my first talk in 2019, and BPB Publications has played a massive part in bringing it to fruition. I am indebted to their guidance and expertise in bringing this book and dream to fulfillment. Their help in revising this book was invaluable, thanks to the participation and collaboration of reviewers, technical experts, and editors.

Various lessons from this book have come not just from my own research and experience, but also from the valuable contributions of my colleagues and co-workers. Many of the technologists, leaders, security, and compliance experts I have worked with in the tech industry have taught me a lot and provided constant feedback on my work.

Finally, I would like to thank you, the reader, for showing interest in my book and supporting its creation. Your encouragement has been invaluable.

Preface

In 2013, only 40% of the world was connected to the internet, and much of that data travelled in plaintext. Today, almost 70% of the global population is online, and encryption is now a standard practice. However, as our defenses have improved, so have the threats. Advancements in artificial intelligence, particularly generative AI, have reduced the barriers to sophisticated attacks. Emerging technologies such as quantum computing are challenging current protection methods. All in all, they make the traditional perimeter-based security models insufficient. Zero Trust security engineering challenges this outdated approach by adopting a never trust, always verify mindset.

In May 2021, the Colonial Pipeline Ransomware Attack led the U.S. President to declare a state of emergency. The attack led to vast fuel shortages across the country. Countless organizations have crippled their security, suffering devastating fines, penalties, and irreparable reputational damage. In 2025 alone, multiple brands were targeted by sophisticated ransomware attacks, causing significant disruptions and impacting their businesses, revenues, and customers.

Drawing inspiration from similar real-life instances, this book will provide you with all the knowledge you would need to build secure applications. You can apply these techniques regardless of whether your product is hosted on the cloud or on-premises. It will introduce you to the key concepts and requirements you need to know when building software products.

This book explores how to integrate Zero Trust principles into your software development lifecycle, network, and operations, enabling you to build secure, resilient, and scalable. Harnessing the knowledge in this book will position you as a more brilliant, more strategic, safety-driven, and more effective engineer or leader.

Chapter 1: Understanding Zero Trust - Security is everyone's job and primarily focuses on protecting the solution from unauthorized access and damage. This chapter will dive into why security is not a single-team responsibility and why the Zero Trust approach is the right way to approach security. It will also dive into the evolution and history of Zero Trust.

Chapter 2: Architecture and Pillars of Zero Trust - This chapter introduces the architecture behind the Zero Trust approach and the core pillars of the Zero Trust approach to build secure solutions in today's world. The book presents a simple framework that can be applied to any system, organisation, application, or product. This chapter dives into these core architecture principles of Zero Trust and the six core pillars of Zero Trust engineering that will be discussed in detail in the subsequent chapters.

Chapter 3: Secure Your Code, Build and Deploy - Whether you are a developer or a leader, code is a fundamental part of the security posture of all systems. So, let us learn to shift left security and integrate it into every stage of the development process, adopting practices when writing code.

Chapter 4: Secure Your Identities - Robust identity and access management solutions, moving beyond simple password-based systems to multi-factor authentication and role-based access control. This chapter will cover why this is important and some ways you can embed secure identities in your applications.

Chapter 5: Secure Your Network - Traditionally, organisations relied on a perimeter-based network architecture. The Zero Trust approach focuses on de-perimeterisation. This chapter will cover examples of why this approach is essential and ways to implement it.

Chapter 6: Secure Your Data - Even with sufficient security across all layers, compromises do happen, and you want to protect your crown jewels (and that is data). This chapter will cover how to adopt comprehensive data security management practices that are simple and easy to adopt.

Chapter 7: Manage Vulnerabilities - Establishing a continuous vulnerability management process in your software development and infrastructure landscape is supercritical. This is how you try to keep attackers at bay.

Chapter 8: Continuous Monitoring - A feedback loop is an essential part of sound Zero Trust security. With security implemented across all your layers, you would need appropriate monitoring to detect threats and respond to incidents when they happen, and to watch, learn, and improve your protections.

Chapter 9: Container Security - Since the launch of Kubernetes, containers have become ubiquitous with modern microservices-based architectures. It is thus vital that we review best practices and apply the same framework to container security. This chapter takes a case study of a real application and provides some practical tips for securing containers.

Chapter 10: Security in the Age of Artificial Intelligence - As artificial intelligence becomes embedded in organisations and people increasingly use GenAI tools, it is essential to understand the security implications of these tools and technologies. AI is a double-edged sword as it has reduced the barrier to entry for attackers but also helps improve defenses. This chapter will help you understand the implications of AI and be more aware of this ever-changing landscape.

Chapter 11: Dealing with Complexity - As you start applying security to your products and solutions, you will most likely be faced with complexity. This chapter talks about how to use a risk-based approach to strike the right balance between security and other non-functional requirements. It will also touch upon some techniques that can help you make good judgment calls.

Chapter 12: Policies, Frameworks and Compliance - Whether working in regulated industries or not, policies, frameworks and compliance are essential to organizational governance, risk management, and legal obligations. This chapter will emphasize the role of compliance in ensuring that organizations meet legal requirements, industry standards, and ethical expectations, preventing costly violations and reputational damage. It outlines various global and industry-specific regulatory requirements that organizations must navigate, including data protection laws (e.g., GDPR), financial regulations (e.g., AML), and environmental standards.

The book will cover various examples to emphasize the need to build secure, reliable systems and draw insights from industry best practices and experiences. After reading this book, you will be equipped with a simple

and powerful approach, frameworks and best practices that you can apply in building secure and reliable systems.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the *Coloured Images* of the book:

<https://rebrand.ly/43be64>

The code bundle for the book is also hosted on GitHub at
<https://github.com/bpbpublications/Zero-Trust-Security-Engineering>.
In case there's an update to the code, it will be updated on the existing
GitHub repository.

We have code bundles from our rich catalogue of books and videos
available at <https://github.com/bpbpublications>. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

At www.bpbonline.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks. You can check our social media handles below:



Instagram



Facebook



Linkedin



YouTube

Get in touch with us at: business@bpbonline.com for more details.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at business@bpbonline.com with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit www.bpbonline.com. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit www.bpbonline.com.

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



Table of Contents

1. Understanding Zero Trust

Introduction

Structure

Objectives

Importance of security

Evolution of cyberthreats

Complexities when protecting systems

Types of threat actors

Types of attacks

Emerging technologies and impact

Everyone has a part to play

Evolution and history of Zero Trust

Parallel story of origin

Conclusion

Points to remember

References

2. Architecture and Pillars of Zero Trust

Introduction

Structure

Objectives

Traditional approach

Advantages and disadvantages

Case study: Target

Zero Trust architecture principles

Terms and definitions

Never trust, always verify

Key objectives of security

Core pillars of Zero Trust

Secure your code, build and deploy

Secure your identities and access

Secure your network

Secure your data and APIs

Manage vulnerabilities

Continuously monitor and improve

Conclusion

Points to remember

References

3. Secure Your Code, Build and Deploy

Introduction

Structure

Objectives

Case study: SolarWinds hack

Detailed timeline

Learnings

Case study: MOVEit vulnerability

Detailed timeline

Learnings

Other notable examples

XZ Utils backdoor

British Airways supply chain attack

Core concepts

Secure requirements and design

- Threat modeling*
- OWASP top 10*
- Compliance requirements*
 - Common pitfalls and solutions*
- Secure coding
 - Secure coding checklist*
- Secure code review*
 - Infrastructure as code and checklists*
- Secure build
 - Common pitfalls and solutions*
- Secure scan and test
- Secure deploy
 - Common pitfalls and solutions*
- Continuous integration and deployment monitoring
 - Sample metrics*
- Tools and plugins
 - Common pitfalls and solutions*
- Automation and security*
 - Guidelines and checklists
 - For developers and engineers*
 - For infrastructure engineers*
 - For leaders*
- Conclusion
- Points to remember
- References

4. Secure Your Identities

- Introduction
- Structure
- Objectives

Case study: Uber

Detailed timeline

Learnings

Case study: Plex

Case study: Cash App

Case study: Target

Strong authentication

Multi-factor authentication

Real-world relevance

Common pitfalls and solutions

Access controls

Least privilege access

Real-world relevance

Role-based access control

Real-world relevance

Attribute-based access control

Real-world relevance

Segregation of duties

Real-world relevance

Common pitfalls and solutions

Securing privileged access

Privileged access management

Short-lived credentials

Secure remote access

Incident response and recovery

Real-world relevance

Common pitfalls and solutions

Secure federated identities

Single sign-on

Standards and protocols

Common pitfalls and solutions

Real-world relevance

Identity governance and compliance

Identity lifecycle management

Access reviews

Compliance

Identity governance and administration

Real-world relevance

Common pitfalls and solutions

Securing APIs and systems

Zero Trust model for API and microservices

Access keys

Real-world relevance

Access tokens

Real-world relevance

Mutual authentication

Real-world relevance

OAuth 2.0 and API security best practices

Real-world relevance

API and system security options

Policy design

Continuous monitoring and response

Emerging technologies

Passwordless authentication

Cloud-based access control

AI and machine learning integration

Behavioral biometric

Decentralized identity

Edge computing and IoT

Regulatory change

Tools, plugins and technologies

Guidelines and checklists

For developers and engineers

For infrastructure engineers

For leaders

Conclusion

Points to remember

References

5. Secure Your Network

Introduction

Structure

Objectives

Case study: NotPetya

Case study: Target

Case study: Sony

Case study: TJX companies' breach

Core concepts

Secure network design and segmentation

Network segmentation and isolation

Real-world relevance

Micro-segmentation strategies

Real-world relevance

Firewalls

Real-world relevance

Virtual private networks

Real-world relevance

Zero Trust Network Access

Real-world relevance

Identity and access

Device trust assessment

Real-world relevance

Secure communication

- Secure communication protocols*
 - Secure routing protocols*
- Real-world relevance*
 - Domain name system security*
 - Web application firewalls*
- Real-world relevance*
 - API gateways*
- Real-world relevance*
 - Network policies*
 - Service mesh*
- Real-world relevance*
 - Network operations
 - Emerging technologies
 - Artificial intelligence and machine learning*
 - Remote browser isolation*
- Quantum-safe networking*
 - Edge computing and network security*
- Guidelines and checklists
 - For developers and engineers*
 - For infrastructure engineers*
 - For leaders*
- Conclusion
- Points to remember
- References

6. Secure Your Data

- Introduction
- Structure
- Objectives
- Case study: Marriott
 - Case study: Yahoo breach*

Case study: LastPass

Case study: H&M

Data security requirements and design

Data classification and governance

Real-world relevance

Regulatory compliance

Real-world relevance

Data privacy

Real-world relevance

Data protection at rest and in transit

Common pitfalls and solutions

Cryptography and encryption

Federal Information Processing Standards

Real-world relevance

Key management

Real-world relevance

Tokenization and data masking

Tokenization

Real-world relevance

Data masking

Real-world relevance

Transport Layer Security

Real-world relevance

Mutual Transport Layer Security

Real-world relevance

Digital certificates

Public key infrastructure

CA hierarchy

Real-world relevance

Common pitfalls and solutions

Access control

- Secure data lifecycle*
- Real-world relevance*
- Common pitfalls and solutions*
- Data governance*
- Data governance framework*
 - Secure archival and deletion
 - Common pitfalls and solutions*
 - Datastore security
 - Customer managed keys for encryption*
 - Field-level encryption*
- Real-world relevance*
 - Continuous monitoring*
- Emerging technologies
- Guidelines and checklists
 - For developers and engineers*
 - For infrastructure engineers*
 - For leaders*
- Conclusion
- Points to remember
- References

7. Manage Vulnerabilities

- Introduction
- Structure
- Objectives
- Case study: Capital One
 - Detailed timeline*
 - Learnings*
- Case study: Equifax
 - Detailed timeline*

Learnings

Other notable examples

Core concepts

Vulnerability management lifecycle

Real-world relevance

Zero-day vulnerability

Discover

Asset discovery and inventory management

Configuration management

Real-world relevance

Software composition analysis

Open-source software

Static application security testing

Dynamic application security testing

Network discovery

Prioritize and assess

Asset criticality

Common Vulnerability Scoring System

Business impact analysis

Remediate

Validate

Report

Monitor and improve

Cloud Security Posture Management

Cloud Workload Protection Platform

Honeypots

Emerging technologies

Software bill of materials

Integration of incident response

AI-based prioritization

AI-assisted scanning and updates

Continuous assessment and threat intelligence
Cloud Native Application Protection Platform
Guidelines and checklists
For developers and engineers
For infrastructure engineers
For leaders
Conclusion
Points to remember
References

8. Continuous Monitoring

Introduction
Structure
Objectives
Case study: Colonial Pipelines
Detailed timeline
Learnings
Other notable examples
Core concepts
Monitoring strategy
Monitoring infrastructure
Technology integration
Analysis and reporting
Compliance and audit
Security operations center
Incident response
Roles and responsibilities
Data classification
Team management
Security awareness and learning
Backup and recovery

Backups

Recovery and resilience testing

Forensic analysis

Process automation

Real-world relevance

Challenges and anti-patterns

Emerging technologies

Guidelines and checklists

For developers and engineers

For infrastructure engineers

For leaders

Conclusion

Points to remember

References

9. Container Security

Introduction

Structure

Objectives

Overview of containers

Brief history

Containers and container platforms

Security of containers

Use case introduction

Applying the Zero Trust approach

Secure your code

Secure your identities

Secure your network

Secure your data

Vulnerability management

Continuous monitoring

Conclusion

Points to remember

References

10. Security in the Age of Artificial Intelligence

Introduction

Structure

Objectives

Brief history of AI

Emerging threats due to AI

Deepfakes and synthetic media

AI-driven phishing and social engineering

Autonomous AI agents and unchecked access

AI-enhanced malware and automated attacks

Zero-day and one-day attacks

Identity and access attacks

Economic and political manipulation

Complexity and cost of protection

AI as a defense tool

Real-time threat detection

Deep network monitoring

Physical security monitoring

Phishing detection

Endpoint security

Alert prioritization

Automated and adaptive response

Continuous learning and awareness

AI-specific threats and mitigations

Regulatory and ethical considerations

Checklist for building secure AI systems

Conclusion

Points to remember

References

11. Dealing with Complexity

Introduction

Structure

Objectives

Security vs. other requirements

Striking the right balance

Human-centric approach

Risk-based approach

Risk assessment

Risk prioritization

Business goals

Architecture optimization

Phased implementation

Automation

Conclusion

Points to remember

References

12. Policies, Frameworks and Compliance

Introduction

Structure

Objectives

Compliance overview

Compliance and regulatory organizations

Real-world relevance

Compliance regulations

Data protection

Consumer duty and protection

Corporate governance and accounting

Anti-money laundering

Financial

Medical and pharma

Real-world relevance

Standards and frameworks

Emerging trends and technologies

Conclusion

Points to remember

References

Index

CHAPTER 1

Understanding Zero Trust

Introduction

Building robust cybersecurity measures is imperative in the current digital age. As you will see in this chapter, there are threat actors everywhere, with the proliferation of sophisticated cyber threats and vulnerabilities. Organizations must fortify their defenses to protect sensitive information and maintain operational integrity. Security has also evolved drastically from tapping phone lines to highly sophisticated supply chain and phishing attacks to network breaches. Modern tools and technologies, such as **artificial intelligence (AI)** and **machine learning (ML)** and quantum computing advances, are opening new frontiers in security protection and reducing the barrier to sophisticated attacks.

Zero Trust, a security model that challenges the traditional perimeter-based approach, emerges as a pivotal strategy in addressing these challenges. This book goes beyond the standard Zero Trust security approach within the network. It applies this to the entire software development lifecycle and operations for anyone engineering solutions on modern technologies.

Structure

The following topics are covered in the chapter:

- Importance of security
- Complexities when protecting systems
- Emerging technologies and impact
- Evolution and history of Zero Trust

Objectives

This chapter provides a background on the importance of security and the complexities involved in protecting systems. Emphasizing the growing complexities in safeguarding systems, we will dissect the multifaceted nature of emerging threats, from ransomware to insider threats, and the pivotal role of Zero Trust in mitigating these risks. We will also look at emerging technologies and threats, enabling attackers and organizations to protect their systems. Lastly, we will dive into the evolution and history of Zero Trust, its core inspiration, and how it differs from traditional approaches to security.

Importance of security

In March 2022, *Pegasus Airlines*^[1] (a Turkish airline) was notified by *SafetyDetectives*, a cybersecurity team, that they had left a large amount of data unprotected online. It was sensitive flight data that was linked to their flight system software. This software helped pilots manage in-flight processes and contained flight charts, navigation materials, crew's **personally identifiable information (PII)**, and software source code. The data amounted to 23 million files. In this situation, no malicious intent was detected, and it was a plain example of employee negligence and human error. Exposing their crew's (including pilots') personal data and confidential information on flights could have caused disasters if an attacker had gained access to it.

If you think that outcomes cannot be too disastrous, here is another one. In May 2021, *Colonial Pipeline Ransomware Attack*^[2] (American Oil Pipeline system) suffered a ransomware attack. It was considered the most significant cyberattack on an oil infrastructure target in the history of the United States, leading to a regional emergency declaration for 17 states and Washington,

D.C., to keep fuel supply lines open. The company had to pay the amount that was asked by the hacker group (75 bitcoin or \$4.4 million). A direct impact was fuel shortages at *Douglas Airport*, requiring American Airlines to change flight schedules temporarily, and at least two flights (to Honolulu and London) had fuel stops or plane changes added to their schedules for four days. Over the days, this led to fuel shortages across multiple states.

U.S. President *Joe Biden* declared a state of emergency on May 9, 2021. The attackers used a disused VPN account to breach the perimeter and enter the Colonial Pipelines network. Further analysis of the incident showed a lack of appropriate identity and access management controls as well as network segmentation.

We will talk about many examples and crises throughout the book, but it boils down to the fact that many of them could have been averted. What happened at *Pegasus Airlines* may not have led to financial penalties, but it did cause reputational damage. In the case of Colonial Pipeline, it was not just the ransomware but also the disruption across the country. A lot of times, these issues boil down to problems such as weak passwords, misconfigurations and human errors, supply chain vulnerabilities, and insider threats, to name a few. All of these might be overwhelming, but rest assured, we will dive deeper into these as we unpack the various principles in this book. The point is that many such instances could have been averted by following certain security principles, which is the book's focus.

The information systems, digital software, and products are a common thread across these two occurrences and other attacks. Have you wondered what the earliest form of known security attack is (electronic)? Here is a sneak peek into the history.

Evolution of cyberthreats

The telegraph is the earliest known electronic device. It had a simple weakness that could be abused; anyone who wanted to snoop could tap into the wire with cable cutters to listen in on your conversations, and even the telegraph operator (see [Figure 1.1](#)) might also leak the contents of your message. One of the early examples of telegraph eavesdropping involved the US government routinely listening in on telegraph lines for intelligence and national security purposes. During the civil war, intercepted messages^[3] were

often handed over to *Abraham Lincoln* himself. The following figure shows human operators in the Bell System Telephone Switchboard; they talk to the caller and connect them to the appropriate circuit to communicate with the receiver:



Figure 1.1: Human operators route and listen in on telephone calls^[1]

The telegraph evolved into the telephone, the radiotelegraph, the **Advanced Research Projects Agency Network (ARPANET)**, and the internet as we know it today. Today, almost every organization is digital and nearly everything runs on software. Most people use software to control the behaviors of services offered to customers. As most industries go digital, the impacts of security compromises can be devastating. Here are some examples across a few of the key sectors:

- **Healthcare:**

- **Date:** June 2024
- **Country:** United Kingdom
- **Summary:** A ransomware cyberattack on Synnovis led to the disruption of services offered by major hospitals around London.
- **Impact:** A ransomware attack on Synnovis, in June 2024 severely disrupted patient care across major hospitals in London, delaying thousands of procedures and disrupting services at a few major hospitals across southeast London^[4]. Synnovis processes blood tests

on behalf of several NHS organizations, many of which were using the Synnovis Library information management system, which consolidated multiple systems and went live in October 2023. The incident had a major impact on the delivery services, with blood transfusions being particularly affected. About 6,199 acute outpatient appointments and 1,491 elective procedures got postponed from 3 June onwards.^[5]

- **Root cause:** A Russian cybercrime group injected malware into Synnovis's IT system, locked the entire computer system and demanded a \$50 million ransom. Stolen data included sensitive patient details such as names, dates of birth, NHS numbers, and descriptions of blood tests.^[5]

- **Transport and logistics:**

- **Date:** November 2023
- **Country:** Australia
- **Summary:** Suspension of container operations and employee data theft due to a breach affecting major terminals.^[6]
- **Impact:** Three-day suspension of operations, leading to 30,000 container backlogs. DP World Australia experienced a cyberattack in November 2023, leading to a three-day suspension of operations and the theft of employee personal data. The breach, detected on November 10, affected container terminals in Melbourne, Sydney, Brisbane, and Fremantle, disrupting freight movements. No customer data was compromised, and no ransomware was found. The company cleared a backlog of over 30,000 containers by November 20.
- **Root cause:** According to multiple cybersecurity analysts, the port operator failed to fix a critical IT vulnerability called *CitrixBleed* when it was hit.

- **Finance:**

- **Date:** September 2017
- **Country:** USA
- **Summary:** 147 million customers' data was exfiltrated for months by hackers.

- **Impact:** One of the well-known breaches within financial services was of *Equifax*. During September 2017, more than 40% of the population of America was potentially impacted by the Equifax data breach. The data that was compromised included names, dates of birth, social security numbers, driver's license numbers, and credit card numbers.

Due to the highly sensitive nature of PII and compromised financial information, Equifax was fined \$700 million for the breach.^[7]

- **Root cause:**
 - The company failed to patch a well-known vulnerability (CVE-2017-5638) in its open-source developing framework, *Apache Struts*. At the time of the breach, the patch for CVE-2017-5638 had been available for six months.
 - Equifax failed to segment its ecosystem, so the attackers could seamlessly access multiple servers after gaining access through the web portal breach.
 - The hackers found usernames and passwords sorted in plain text, which they used to escalate privileges and gain deeper access.

Here is a summary of publicly reported cyber incidents across industries^[6]:

Publicly reported cyber incidents by industry - 1 July 2023 - 30 June 2024

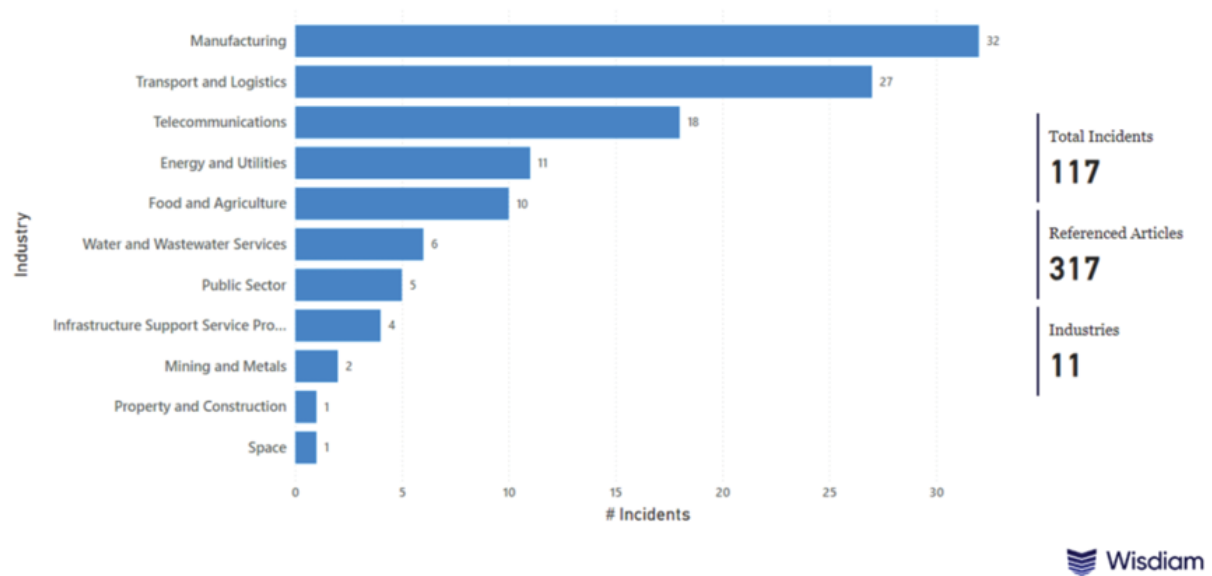


Figure 1.2: Publicly reported cyber incidents 2024

As per *Statista*^[8], the finance sector was the most targeted by attackers. And rightly so; everybody needs financial services where people save money, trade, and transact. Attacking and compromising a bank can have far-reaching consequences. Let us look at the following figure to note the sectoral distribution of cyberattacks:

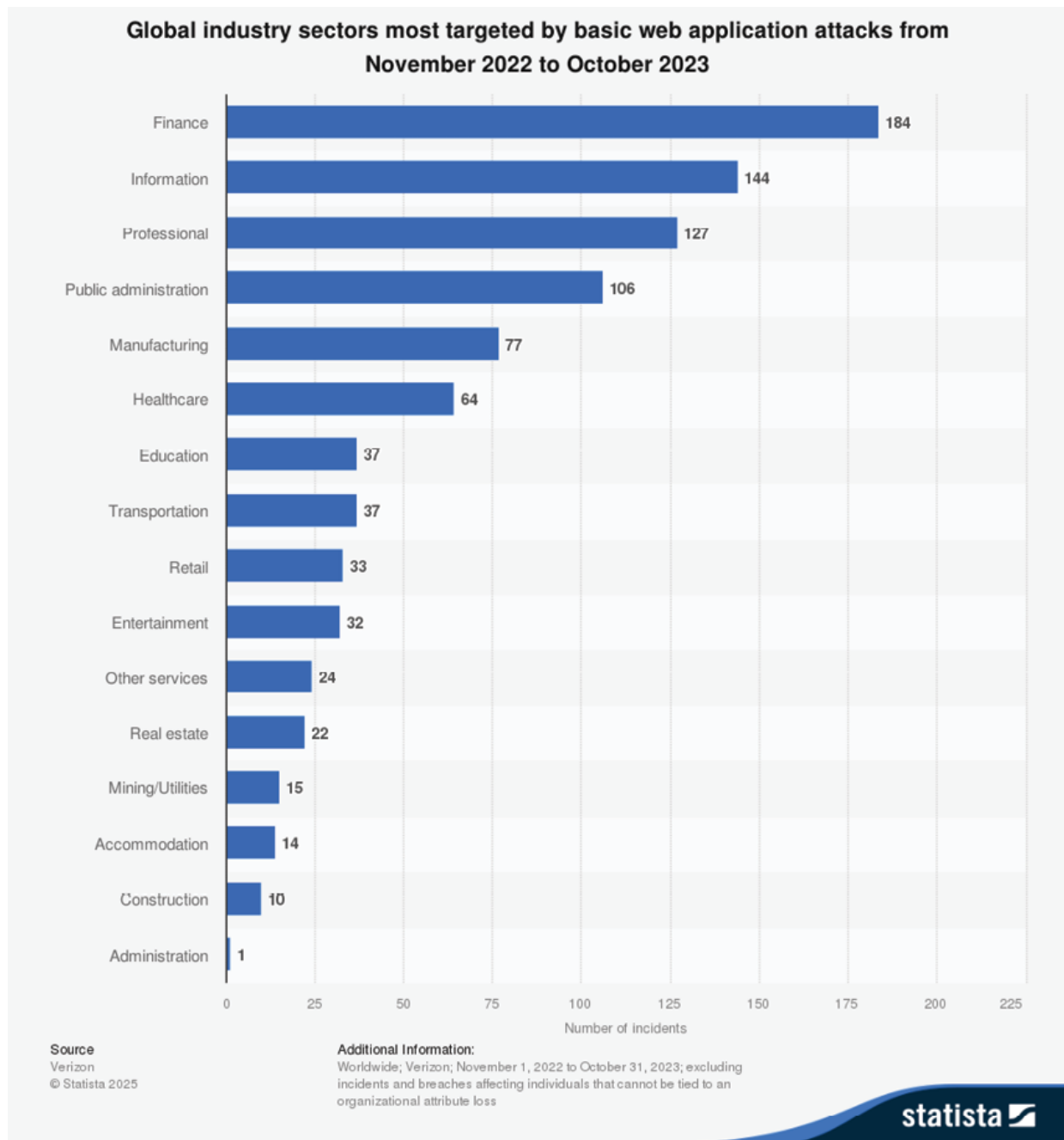


Figure 1.3: Global industry sectors targeted by attacks

These are a handful of examples; we will cover more such examples throughout the book, aiming to give real-life insights into the approach that you should take to prevent the likelihood of these attacks. These examples highlight some of the common effects of these cyberattacks and leaks. These can be summarized in the following table ^[9]:

Type	Description

Revenue loss	Regardless of whether the attack affects all or partial systems, most affected businesses must shut down parts of their operations and systems while the attack is being resolved. In some cases, even a core system such as billing infrastructure, like in the case of Colonial Pipeline, could affect operations. In many cases, a ransom payment is also required to recover services or data. All of this implies revenue losses.
Reputation damage	Every organization is responsible for revealing breaches' details and informing affected customers. Once such a data breach occurs, their reputation is tarnished. For example, the Equifax attack shows some glaring limitations in its security posture. Customers would take such attacks seriously and lose faith in the brand. It can take a long time to build that reputation back again.
Loss of customers	With reputation damage, there is also a loss of customers, both new and existing. Most existing customers will seek alternatives during the outage or lockdown of services. However, in the long term, customers will choose a different organization to do business with.
Productivity loss	When the attack happens, most organizations and their teams are under stress to solve the mishap and bring services back to normal. As a result, organizations and their teams lose valuable time that could have been spent on building the business. This can also be due to the shutdown of systems that the employees or teams need to use for their day-to-day operations.
Legal implications	In most regulated industries, there are also fines and legal penalties for organizations. For instance, Equifax was fined \$700 million by the Federal Trade Commission. ^[7]
Service disruptions	Most security incidents impact service availability. Depending on the industry, this could delay healthcare services (e.g. Sinnovys outage), delivery of commodities such as fuel (e.g. colonial pipeline) or payments and fund transfers (in case of banks). Some of these could be so critical that it could impact people's lives and well-being.

Table 1.1: Effects of cyberattacks

To summarize, the impact of security outages and data breaches can be far-reaching. They could start regionally and spread across countries and continents. It is thus super important for organizations to do their best to prevent such breaches and outages.

Now that you have seen the common types and effects of the attack, let us understand a bit more about who the attackers are, what their motivation is, and why security is so complex.

Complexities when protecting systems

In this world and age, protecting systems has gotten extremely hard due to the increasing number and types of threat actors and their access to funding, tools, and technologies. There are myriad types of attacks these threat actors can launch, and they typically do so over a long period to avoid being noticed. Additionally, emerging tools have made the jobs of attackers even easier.

Types of threat actors

The term you would often hear in the cybersecurity context is threat actors. Threat actors are individuals or groups carrying out malicious cyberspace activities, targeting digital systems, networks, and data. Hackers is another term that is often used. Still, hackers typically use their skill and knowledge to overcome a problem or uncover a vulnerability and may or may not have malicious intent. Threat actors, on the contrary, always have malicious intent. There are various types of hackers: white hat (typically ethical and work on improving cybersecurity, conduct penetration testing), black hat (also called cybercriminals), and gray hat (operating in a moral gray area, sometimes offering to fix issues for a fee).

So, let us start by looking at different types of threat actors^[10] that most organizations face and their motivations:

- **Cybercriminals:**

- **Details:** This category of threat actors is criminals, either individuals or groups, who use technology to commit malicious acts. They typically abuse a vulnerability in the digital systems or networks, which we will cover later in the book.
- **Motivation:** Mostly financial gain.
- **Methods:** Phishing, ransomware, malware, and social engineering.
- **Impact:** Data theft, financial losses, and disruption of service.
- **Target(s):** Businesses, individuals, and financial institutions.
- **Challenge:** These actors can be sophisticated individuals or groups, but do not usually have deep pockets like State-sponsored attackers.

- **Hactivist:**

- **Details:** Hactivist combines the terms hacking and activism and uses

cyberattacks to promote political or social agendas. They usually want to drive justice, transparency and social causes, focus on exposing perceived wrongdoings, and protest powerful or government policies.

- **Motivation:** Political, social, or ideological causes.
- **Method(s):** Website defacement, DDoS attacks, data leaks.
- **Impact:** Reputational damage, service disruption, and exposure of sensitive information.
- **Targets:** Organizations perceived as unethical or opposing their beliefs.
- **Challenge:** Since an ideology or social agenda drives them, they are hard to convince to change their minds.
- **Nation-state sponsored attackers:**
 - **Details:** They are highly skilled individuals backed by government entities, conducting sophisticated operations against other nations and organizations. They have significant access to resources and technology and typically employ techniques not detected for long.
 - **Motivation:** Support their nation or state's interests, such as espionage, sabotage, or destabilizing a country or state. They can also be motivated by financial gain, political agendas (such as during elections, etc.), or learning more about other country's drivers and infrastructure.
 - **Method(s):** Carry out planned and executed attacks that go undetected for a long period, sophisticated custom malware, and social engineering.
 - **Target(s):** Government institutions, critical infrastructure, rival nations or states.
 - **Impact:** National security threats, intellectual property theft, infrastructure or service disruption, espionage, and intelligence.
 - **Challenge:** Significant sponsorship and resources and access to talent for sophisticated attacks.
- **Insider threats:**
 - **Details:** These originate from inside the organization and involve individuals with access to the systems either when they are employed

or just left employment and still have retained access. Insider threats can also be unintentional and either due to carelessness or lack of awareness, in which case the individual exposes data or details to an outsider without understanding the implications.

- **Motivation:** Varies (revenge, financial gain, accidental).
- **Method(s):** Abuse of privileged access, data exfiltration.
- **Impact:** Data breaches, financial losses, operational disruption, reputational damage.
- **Target(s):** Their own organizations (past or current).
- **Challenge:** Although these may not be as sophisticated as the other categories and mostly work individually, they can do significant damage, primarily because of the amount of access they may have.

Aside from these, there are a few other categories, namely, cyber terrorists, organized crime groups, and thrill seekers; but their motivations and impacts are not too different from the ones discussed previously.

With this knowledge about the various threat actors and their motivations, let us discuss why security is so difficult due to the variety of attacks and the impact of emerging technologies.

Types of attacks

As seen in the previous section, the earliest known communication mechanism, the *Telegraph* had a weakness that could be abused. Anyone who wanted to snoop could tap into the wire with cable cutters to listen in on your conversations (also called a **man-in-the-middle (MITM)** attack), or the telegraph operator (now called the insider attack) might leak the contents of your message. These days, with the adoption of technology and especially with the availability of AI, attacks have gotten sophisticated, and generating an attack is much easier. Some common types of attacks are listed as follows:

- **Phishing:** Cybercriminals use fake emails, text messages, or websites to trick customers or employees into sharing sensitive information by impersonating trusted identities. Typically, phishing is a gateway to other attacks to obtain credentials and access systems or plant malware. As per IBM X-Force, phishing is the primary infection vector in 41% of

cybersecurity incidents. Also, 94% of malware is delivered via email, per Panda.^[1] Here is an example of a phishing attack:

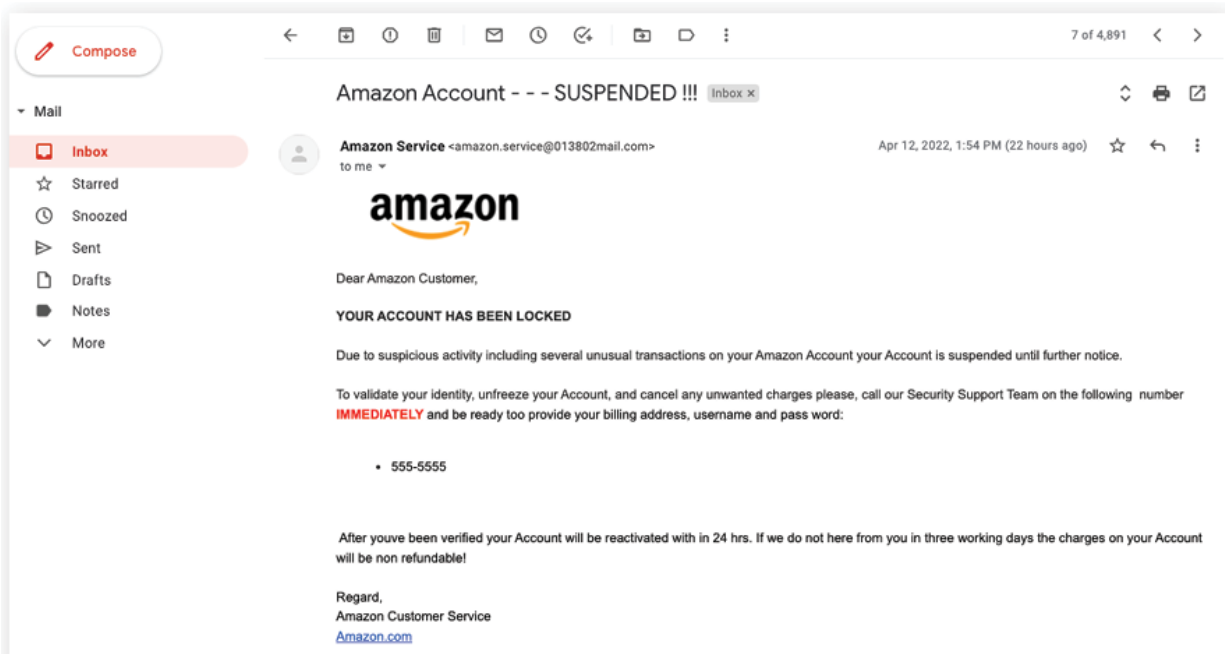


Figure 1.4: Sample phishing email^[2]

- **Malware:** The term represents any malicious software designed to disrupt, damage or gain unauthorized access to data or systems. Usually distributed in the form of files or code, it tries to achieve one of the following:
 - Access of an infected system to a remote attacker.
 - Send spam from an infected system to other targets.
 - Investigate the infected user or systems' local network.
 - Extract and send sensitive data from the infected system.

Malware can take multiple forms. Here are some of the common ones:

- Adware is a software that displays ads or banners but can sometimes cause unauthorized access and disrupt users.
- Botnets (robot networks) are networks of infected computers under the control of a single attacking party using command-and-control servers. Typically used in **distributed denial-of-service (DDoS)** attacks.

- Spyware is malware that collects information about the system and communicates with the attacker, such as key loggers.
- Viruses typically copy themselves on a computer or network to corrupt or delete data. Still, these are attached to a program and are only activated when a user opens the program.
- Worm self-replicating virus exploits vulnerabilities to spread across computers, unlike viruses that do not attach to existing files or programs and can go unnoticed longer.
- Cryptojacking is malicious crypto-mining when criminals hack into a business and install software.

The mobile malware statistics in [Figure 1.5](#) shows that nowadays, smartphones are the primary way consumers use digital media, and mobile malware has become quite common. Due to the growth and usage of these devices, it is now much easier to scam or send a phishing text email. People can also have a false sense of security when using phones and opening links sent to them.^[12]

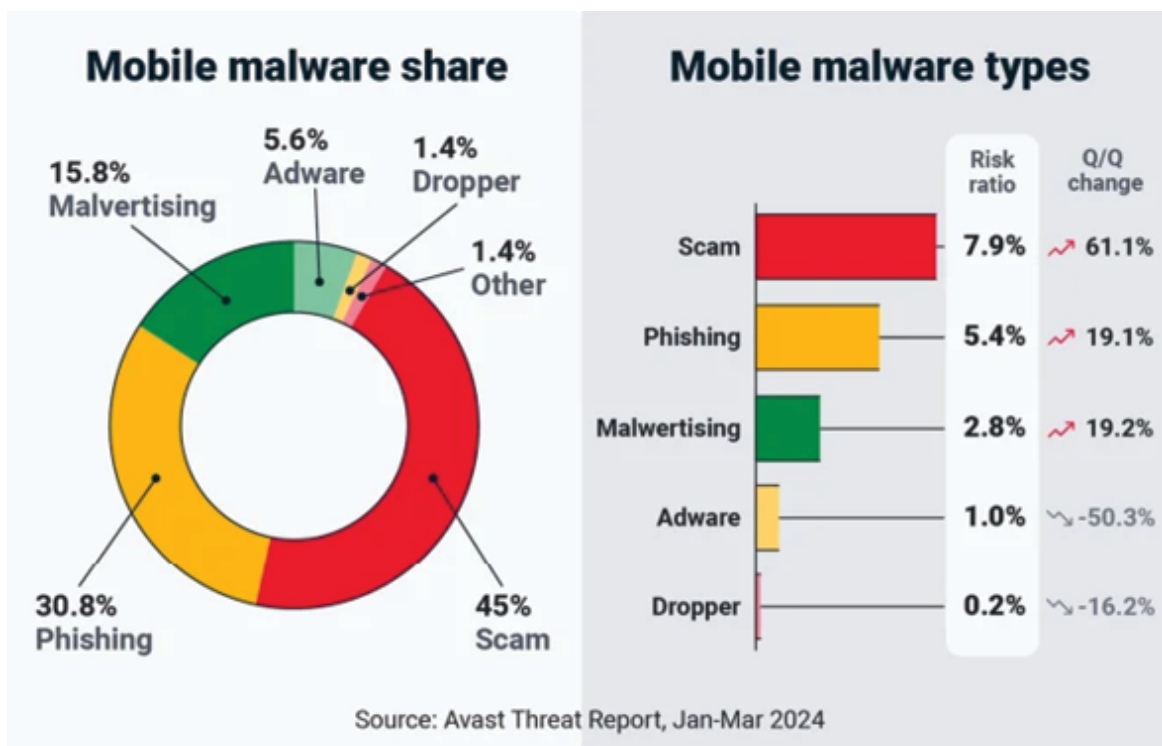


Figure 1.5: Mobile malware statistics for the first quarter of 2024 ^[3]

Malware is so common that in 2023 alone, there were 6 billion attacks, approximately 190,000 malware attacks per second.^[12]

- **Ransomware:** It is a type of malware that encrypts a victim's data and demands a ransom to decrypt it. Ransomware attacks often start with phishing emails. In 2024, 54% of organizations were hit by ransomware. 99% of the organizations hit by ransomware were able to identify the root cause of the attack, with exploited vulnerabilities as the commonly identified starting point for the second year running. Also, ransom payments have increased significantly, with a median payment of \$2,000,000 (a 5X increase on the \$400,000 reported in 2023) and a mean payment of \$3,960,917 (a 2.6X increase on the \$1,542,330 reported in 2023). Ransomware thus remains a major threat to organizations of all sizes around the globe.^[13]
- **DDoS attacks:** DDos attacks overwhelm a system with resources, making it partly or fully inaccessible to users. DDoS attacks can disrupt access to the system, such as online banking, health systems and trading platforms. Attackers typically create traffic from different sources, typically from resources they have already compromised, leading to high network traffic, website slowness and inaccessibility. This typically leads to service disruptions with increased error rates. During the Ukraine war, DDoS has been used to paralyze essential services.
- **Supply chain attacks:** These attacks involve compromising a third-party software, data source, and software provider. In this case, typically, a threat actor will gain access to third-party software or data that an organization uses. Today, many organizations rely on third-party software and services, be it payroll, human resources, finance, security, or payment systems. In addition, even when organizations build their software solutions, they leverage many external libraries and software. If any of these software or libraries were compromised, it could go undetected for a long time as they come from a legitimate supplier, and the organization using them has no reason to suspect it as an attack. In 2017, a malware attack targeted Ukraine's government and infrastructure and spread to other countries via a supply chain attack on a tax accounting software called *MeDoc*. As per *ReversingLabs*, 2023 saw a 28% increase from the year before in the total number of

malicious packages uploaded to open-source repositories.^[14]

- **Insider threats:** An insider threat is when a member of an organization with access to systems and data causes a data breach, fraud, or theft. Insider threats take various forms, such as an employee or a member of the organization giving away sensitive information knowingly or accidentally, someone abusing a security shortcoming to gain a badge or access via another employee, or a third party with access to systems or networks. Insider threats can be due to negligence (someone ignoring a policy violation, say piggybacking at the security entrance), accidental (mistyping or revealing sensitive information where it should not be), or threats (by a malicious insider or someone holding a person hostage).

As per *Cybersecurity Insiders*, 48% of organizations report that insider attacks have become more frequent in 2024, and 51% have experienced six or more attacks in the past year. The key drivers are complex IT environments, adoption of new technologies, and inadequate security measures.^[15]

- **MITM attacks:** As the name suggests, this attack typically involves a third party between two communicating parties. This is akin to someone intercepting telegraph and radio conversations in the past. These days, the attack involves cybercriminals intercepting communication between two parties to steal data over communication channels. The goal is to steal personal data, login credentials, account and credit card numbers. Most attackers use a combination of eavesdropping (intercepting traffic close to endpoints or redirecting or passing through a node), spoofing (an attacker impersonates voice or messages), tricking users (to accept fake digital certificates), and breaking SSL/TLS connections (using SSL stripping or TLS downgrade attacks). Per *JumpCloud*^[16], MITM accounted for 19% of successful cyberattacks in 2024.

Emerging technologies and impact

We have already seen examples of misconfiguration previously, and cloud computing's widespread adoption has led to misconfigured cloud storage becoming a common source of data breaches. Cloud computing is only expanding its footprint globally, and cloud adoption among enterprise

organizations is over 94% as per *RightScale*^[17]. However, cloud adoption also brings a lot of complexities; there is a widening visibility gap across cloud operations and infrastructure; as per *CloudZero*^[17], the most significant threats in the public cloud include misconfiguration (68%), unauthorized access (58%), insecure interfaces (52%), hijacking of accounts (50%). Although the cloud brings a lot of security advancements, it is important to understand that security on the cloud is the customer's responsibility, the cloud provider is only responsible for the security of the cloud, and the security of applications running on the cloud is still the customer's responsibility.

The advent of **Internet of Things (IoT)** environments involves a network of many smaller devices and sensors. Typically, these devices are harder to upgrade and maintain, making them susceptible to attacks such as MITM. IoT has dramatically expanded the attack surface, with billions of often poorly secured devices providing network entry points. These IoT devices frequently lack proper security protocols and regular updates, creating persistent vulnerabilities.

One more recent technological advance has been in AI and ML. This space has evolved dramatically since the 1950s; the most significant breakthroughs have occurred in the past decade with the rise of deep learning and neural networks. In addition, the biggest revolutionary shift was the public release of *ChatGPT* by *OpenAI* in November 2022. The release introduced the world to **large language models (LLMs)** and sparked the era of generative AI. Trained on vast amounts of data, AI can now create human-like text, convert text to images, generate code, and even create music and videos.

This rapid advancement presents additional complexity in cybersecurity, as sophisticated AI models can now automate the discovery of software vulnerabilities, generate convincing phishing emails, and create deepfakes that bypass traditional authentication systems. In [Chapter 10, Security in the Age of Artificial Intelligence](#), we will dive into the specific impacts of AI and preventative measures for those building and using AI. However, these advancements have democratized the use of AI for everyone, including attackers, bringing down the barrier of creating sophisticated attacks. *Group-IB* experts^[18] discovered a 70% increase in public posts offering zero-day exploits for sale. As per *JumpCloud*^[16], phishing attacks have increased by

over 1,000% as a result of generative AI; bad actors are increasingly using AI models like *WormGPT* and *FraudGPT* to launch APT spear-phishing attacks. AI has helped hackers create more sophisticated malware that evolves on the fly to evade security systems.

Cryptocurrencies have made it easier for attackers to collect untraceable ransoms, while blockchain technology has ironically made some supply chain attacks more sophisticated.

Extended reality (XR) technologies, including **augmented reality (AR)** and **virtual reality (VR)**, are also becoming prevalent. They raise new privacy and security concerns by collecting extensive personal and biometric data.

Quantum computing, although not yet available for mass adoption, also threatens to break current encryption methods, potentially rendering sensitive data vulnerable once quantum computers reach sufficient power. Once this becomes mainstream, many security protocols, procedures and frameworks will need a huge uplift.

As people spend more and more time on digital devices and virtual environments, security is only getting more difficult to execute. Let us now understand who is ultimately responsible for maintaining a good security posture.

Everyone has a part to play

Insider threat is the most significant type of threat organizations face today. Most employees have a lot more access than an outsider would have, and at times, motivations to cause harm or damage. With the evolving technology landscape, people are also falling prey to, at times, simple attacks (such as phishing and social engineering). Imagine an organization is a vast ship, and any security vulnerabilities are like holes in the boat. If the holes are not fixed, the ship will sink over time. The job of everyone on the ship is to identify, report and fix every single hole on the ship to ensure its safety. Security of any organization is thus as strong as the weakest link. It is thus everyone's responsibility to ensure the safety of their organization. If every person is vigilant and aware, it goes a long way to ensure the organization's safety. Many of the examples covered in this book, and most of the attacks like phishing, insider attacks, and exfiltration, could be averted if the people

building or protecting the systems were to be more vigilant and aware. That is the reason why security is everyone's responsibility. It is not a select few executives or a security team operating in a silo; building and running a truly secure system requires everyone involved to take responsibility and do their part, no matter how small. In the next section, let us look at how Zero Trust fits here.

Evolution and history of Zero Trust

The origins of Zero Trust^[19] are mostly associated with security within networking, originating in 2004, based on the ideas presented by the *Jericho Forum*. The forum later merged into what is now called the *Open Group Security Forum*. This started as a group of like-minded CISOs^[20] wrestling with the limitations of the dominant and unquestioned philosophy of securing all resources by putting them on a secure network behind a security perimeter. *Jericho Forum* promoted the concept of de-perimeterization. Traditional security relied on the concept of perimeterization, basically, there was a boundary drawn between the corporate network and the outside network. Everything inside the corporate network (LAN) was trusted; systems could typically talk to each other without any authentication, authorization, or encryption. *Figure 1.6* shows the various layers of a traditional perimeter network with two firewalls separating external and internal networks:

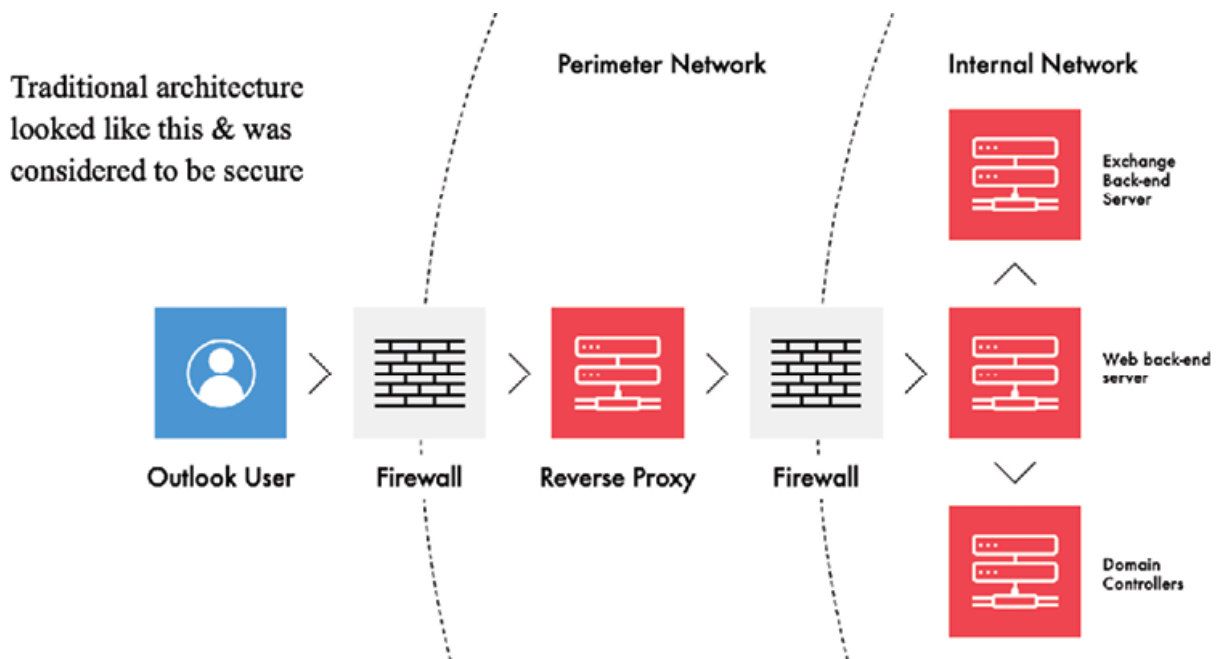


Figure 1.6: Traditional perimeter-based architecture

We will cover this topic in more detail later. However, the most important flaw in this model was the growing number of devices connected 24x7 to corporate networks, so the idea of a trusted boundary became outdated and risky. It was necessary to introduce key controls to secure the network. A lone device connected to the corporate network could quickly go unnoticed and cause much damage. This is precisely what went wrong with *Target* in 2013. The attackers first found a weak device that an HVAC contractor Target had used. After compromising this device, they signed into a contractor system hosted by Target. They uploaded executable files, which would further help them gain remote access to Target servers, finally compromising the confidential data of millions of customers.^[21]

Zero Trust was officially introduced in 2010 by forrester analyst *John Kindervag*. In 2011, Google was the first company to release a full implementation of this approach with BeyondCorp.^[19]

In 2018, the seven core principles of Zero Trust were established, and the **National Institute of Standards and Technology (NIST)**^[22] released the first version of SP 800-207. In 2019, Gartner introduced the term **Zero Trust Network Access (ZTNA)** and the secure access service edge networking model.^[19]

In Jan 2020, the COVID-19 pandemic disrupted many people's lives

globally, and social distancing and remote work became the norm. Remote and hybrid work options became common, and although this was a positive change for many people, it expanded the attack surface as people were not only in controlled environments anymore, working from homes, cafes, airports or wherever they could find a Wi-Fi connection. As a result, since 2021, there has been more emphasis on adopting this approach. There is also a parallel story, though not as popular, which distills this idea further.

Parallel story of origin

There is, however, a parallel story of trust in software, existing much earlier than the previously shared origin story, and it involves none other than the creator of Unix, *Ken Thompson*. In August 1984, Thompson was awarded the *ACM Turing Award* for his work on the Unix operating system, and he also authored a paper titled *Reflections of trusting trust*.^[23] In the paper, he reveals how he could have planted a bug or a *Trojan Horse*, allowing him to log in to any Unix system as a user. He summed it up for us: *You cannot trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No source-level verification or scrutiny will protect you from using untrusted code. This would be an example of a supply chain attack, one of today's imminent threats.*

While not labelled as Zero Trust, Thompson's 1984 essay laid the conceptual groundwork for the idea of never implicitly trusting code. Look around you; you will hardly find anything that does not run on code (of some kind). You must have multiple IoT devices at home, thermostats, kitchen appliances, home security systems, pacemakers, smart speakers, and smart TVs. We are so dependent on our smartphones with all the apps that run on them. On the streets there are multiple cameras and traffic lights, autonomous cars, and the list goes on. Modern-day hospitals and airports all need technology.

The best case in point was the *CrowdStrike* outage on 19 July 2024. People woke up to hear on the radio that every country from Australia to the USA was affected: trains, airlines, hospitals, banks, hotels, manufacturing, stock markets, broadcasting, gas stations, retail stores, and most other businesses were struggling. By an estimate, 8.5 million systems crashed and could not properly restart. This has been called the most significant outage in the history of information technology. The outage disrupted daily life,

businesses, and governments around the world. The estimated damage of this crisis was expected to be at least \$10 billion.^[24]

The root cause appeared to be the release of an update to CrowdStrike's Falcon Sensor software. This software has protected Windows since version 10 and any servers (database, web, etc.) that run on those operating systems. Although antivirus and malware protection are good, CrowdStrike's solution has kernel access. Falcon Sensor software uses sensor content that steers and defines its threat-detection engine's capabilities. This content is meant to be updated frequently to protect from emerging malware and is delivered as files. One of these files delivered on 19th July was not properly validated due to an issue with its validation software. The software loaded and parsed the new content and was confused by the broken template instance, resulting in an out-of-bounds memory read, triggering an exception within CrowdStrike's Windows driver-level code, which would bring down the whole box. Upon reboot, it would start up and crash all over again. The rest is history, causing a global impact.^[25]

This outage was a genuine mistake in the testing and validation process and not a criminal or malicious act. However, it is a stark reminder of the widespread impact of software. Someone with malicious intent can potentially cause a similar impact using supply chain attacks if they have sufficient access.

Moreover, most of these software systems are built using many different layers of software systems; let us look at a simple program that will process resumes:

```
from pdfminer.high_level
import extract_text as pdf_extract_text
import docx2txt
import nltk
import re
import subprocess
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
... rest of the implementation to process resumes.
```

Notice the number of import statements in the code and the number of downloads before you start processing a single resume. Any of these imported libraries could introduce a vulnerability that could send your data to an external system or URL. Imagine if such a program were processing confidential data. This is the simplest example and only one type of threat. In a world where everything runs on code, various threat actors with lots of money can do a lot of damage, let alone causing accidental issues such as what happened at *CrowdStrike* due to a mistake.

In such a world, how do you secure your systems? You build with Zero Trust. Next, we will cover its meaning.

Conclusion

Through this chapter, you were introduced to Zero Trust and its importance. In the next chapter, we will cover the key principles and pillars of Zero Trust based architecture and explore the cornerstones of Zero Trust engineering from securing your code, data, and network to managing vulnerabilities and monitoring.

Points to remember

- No matter what industry you work in, healthcare, transport and logistics, and finance, security is extremely important.
- Systems today involve a lot of complexities. Equally, there are various types of threat actors: cybercriminals, hackers, nation-state-sponsored attackers, and insider threats trying to access confidential information.
- They do this through various attacks such as phishing, malware, ransomware, DDoS attacks, and MITM attacks, to name a few.
- Emerging technologies such as AI, IoT, VR, and quantum computing are making it easier for perpetrators to gain access and making security even harder.
- Security is thus everyone's responsibility; it is only by being vigilant and aware that we can build secure systems.
- Build systems with Zero Trust since traditional approaches are no longer

sufficient.

References

1. <https://www.syteca.com/en/blog/real-life-examples-insider-threat-caused-breaches>
2. https://en.wikipedia.org/wiki/Colonial_Pipeline_ransomware_attack
3. <https://community.spiceworks.com/t/security-history-a-timeline-of-early-electronic-eavesdropping/580924>
4. <https://www.digitalhealth.net/2024/06/critical-incident-at-synnovis-disrupts-pathology-services-across-se-london/>
5. <https://www.bankinfosecurity.com/synnovis-attack-halts-8000-nhs-patient-procedures-so-far-a-25770>
6. <https://wisdiam.com/publications/recent-cyber-attacks-transport-logistics-sector/>
7. <https://www.ftc.gov/news-events/news/press-releases/2019/07/equifax-pay-575-million-part-settlement-ftc-cfpb-states-related-2017-data-breach>
8. <https://www.statista.com/statistics/221293/cyber-crime-target-industries/>
9. <https://copycei.com/the-consequences-of-cyber-attacks-and-their-impact-on-cybersecurity/>
10. <https://www.recordedfuture.com/threat-intelligence-101/threat-actors/threat-actor-types>
11. <https://www.cobalt.io/blog/cybersecurity-statistics-2024>
12. <https://www.avg.com/en/signal/malware-statistics>
13. <https://assets.sophos.com/X24WTUEQ/at/9brgj5n44hqvgsp5f5bqcp/s/sophos-state-of-ransomware-2024-wp.pdf>
14. <https://www.reversinglabs.com/blog/the-state-of-software-supply-chain-security-2024-key-takeaways>

15. <https://www.cybersecurity-insiders.com/portfolio/2024-insider-threat-report-gurukul/>
16. <https://www.cloudzero.com/blog/cloud-computing-statistics/>
17. <https://jumpcloud.com/blog/cyber-attack-statistics-trends>
18. <https://www.helpnetsecurity.com/2024/03/01/hi-tech-crime-trends-2023-2024/>
19. <https://www.cryptomathic.com/blog/the-evolution-of-zero-trust-security>
20. <https://www.microsoft.com/en-us/security/blog/2020/10/28/back-to-the-future-what-the-jericho-forum-taught-us-about-modern-security/>
21. <https://www.sipa.columbia.edu/sites/default/files/2022-11/Target%20Final.pdf>
22. <https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf>
23. https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf
24. https://en.wikipedia.org/wiki/2024_CrowdStrike-related_IT_outages
25. https://www.theregister.com/2024/07/24/crowdstrike_validator_failure/

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



CHAPTER 2

Architecture and Pillars of Zero Trust

Introduction

The previous chapter focused on the importance of security within organizations, the various threat actors, emerging technologies and threats faced by organizations, the role everyone plays in ensuring the security of the organization, and the brief origins and history of Zero Trust. We will now dive into the core architecture principles behind the Zero Trust model and the *never trust, always verify* approach to ensure every digital interaction undergoes continuous validation, irrespective of its origin. This approach mandates strict access controls, micro-segmentation, and real-time threat detection, bolstering a robust security posture across digital platforms. As technological advancements progress alongside evolving security vulnerabilities, Zero Trust has emerged as a crucial framework in safeguarding digital ecosystems against sophisticated cyber threats.

Structure

The following topics are covered in the chapter:

- Traditional approach

- Zero Trust architecture principles
- Key objectives of security
- Core pillars of Zero Trust

Objectives

This chapter delves into the architecture of the Zero Trust model and highlights its indispensable role in securing digital environments. It outlines the core principles necessary for building secure solutions in the modern world. We will cover some case studies to emphasize the importance. Through extensive experience in deploying secure cloud infrastructures and using varied security frameworks, the authors propose a straightforward framework that can be adapted to any system, organization, application, or product. The upcoming chapters will detail the pillars and provide a simple yet effective approach to engineering security in your products and solutions.

Traditional approach

In the last chapter, we covered the two origins of Zero Trust. The first is what Forrester analyst *John Kindervag* introduced, and *Google* adopted in implementing the *BeyondCorp* security model, which the Open Group Security Forum initially pioneered. It was later formalized as a cybersecurity framework by the **National Institute of Standards and Technology (NIST)** in its **NIST Special Publication (SP) 800-207**. NIST is a highly respected non-regulatory agency within the U.S. Department of Commerce that develops and promotes standards and best practices for organizations and government agencies. Their mission is *to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life*¹. Although NIST is a US-based organization, its frameworks and standards have a global impact. **NIST Cybersecurity Framework (CSF)**, **Federal Information Processing Standards (FIPS)**, encryption and hashing algorithms (such as SHA series) are adopted and used by many organizations globally.

This initial section will discuss the salient points and core principles of NIST 800-207. We will then extend these further based on the paper by *Ken Thompson on Reflections on Trusting Trust*.

Let us start by looking at the traditional security model. As shown in [Figure 2.1](#), it was based on defining an internal network and a perimeter network that separated the internal from the external (internet). Typically, these three layers (internal, perimeter, and external) were separated by firewalls, as shown in the following figure. This approach was based on the idea that the **demilitarized zone (DMZ)** or perimeter network was a gateway to the public internet for all internal services.

Typically, publicly available services such as email, web servers, FTP (file) servers, and **Domain Name System (DNS)** servers were placed in the DMZ, allowing a clear separation of external from internal systems. Connectivity between these layers was meticulously controlled using the two firewalls to ensure systems could only talk to other systems on specific ports and IP addresses. However, a common pattern was to use a reverse proxy server, as shown in the figure. This is like a proxy server, an intermediary, but used the other way around. Instead of providing a service to internal users wanting to access an external network, it provides indirect access to internal resources (usually the Internet) through an external network.^[2]

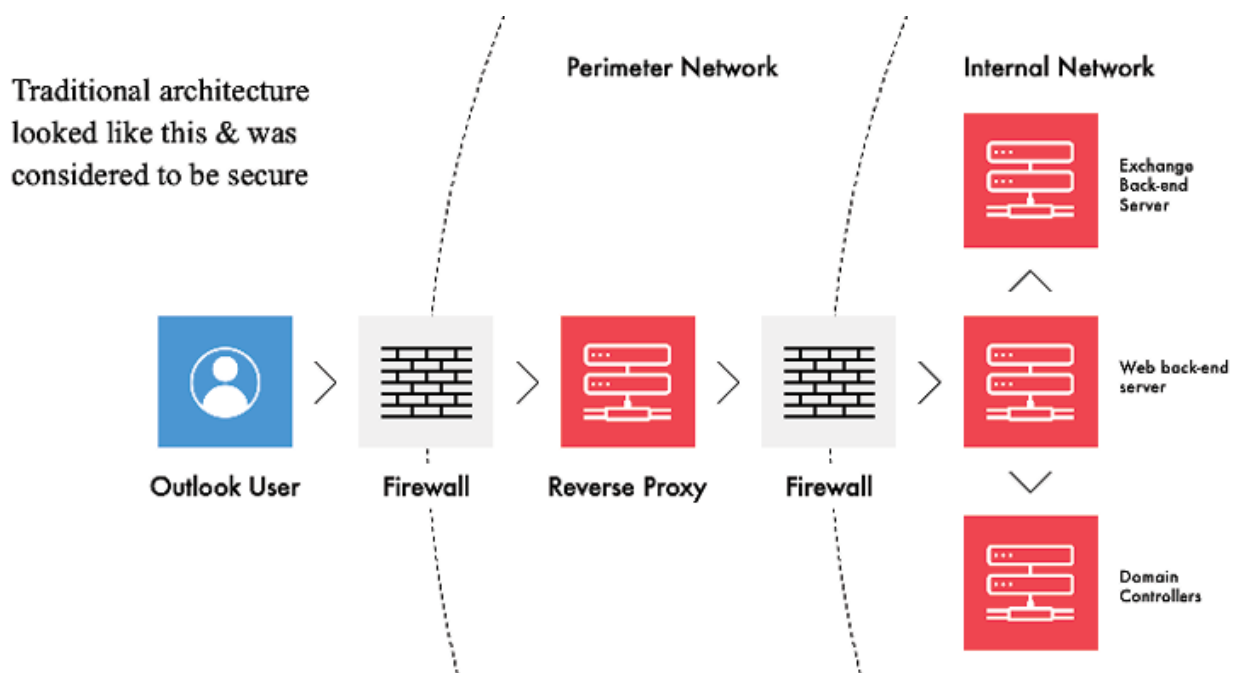


Figure 2.1: Traditional network architecture

Advantages and disadvantages

For a long time, this model worked as it had some clear advantages, mentioned as follows:

- **Simplicity:** Implementing security within network devices was much more manageable by clearly demarcating the perimeters. Users within the organization can still access and share content easily, while external users can gain access through the DMZ (perimeter network). This approach also simplifies troubleshooting for internal and external facing services.
- **Protection:** Although a bit expensive to implement due to the use of appliances, it helped protect the internal resources from attackers such as spoofing IP addresses and DDoS attacks (firewalls can block certain types of traffic, such as **Internet Control Message Protocol (ICMP)** and ping, used for generating these).
- **Reduced risk:** Organizations could limit the resources they must publicly expose. This would also prevent a mistake or misconfiguration from quickly exposing an internal service publicly, making it susceptible to attacks.
- **Enhanced and easy traffic management due to separation:** Using firewalls allows for concentrating on the relevant traffic to each network and transparent policies for what traffic is and is not allowed at each level.

This traditional model relied on the following key assumptions:

- **Clear trust boundary:** Every resource and device can be easily classified as internal or external. An internal user trying to access services could be fully trusted and did not require specific checks, authentication, or authorization. Any external user would, however, must go through more rigorous checks as they could not be trusted.
- **Fewer external and more internal services:** A significant focus is on protecting internal resources. There was an implicit expectation that the services in the perimeter network were limited, and any new services were expected to be hosted internally.
- **Technology change was not rapid:** The DMZ network approach has been discussed since the late 1900s and was popular until the 2010s. During this period, the underlying assumption was that technological change was not rapid. The adoption of digital technology itself was not high, and attackers

were not sophisticated. Hence, the DMZ approach, which focused on more static perimeters, made sense.

- **Lateral movement is uncommon:** The DMZ or perimeter model assumed little movement of resources and users within the network. The model was built for static servers or resources and a consistent set of users connecting to them.

In this model, the internal network was wide open; there were usually no major access controls on the systems, and communication between the users and the services or resources was over unencrypted channels. There was an overreliance on protecting what was exposed externally in the perimeter network and not enough on the internal network. This meant any internal network traffic was susceptible to the same attacks, and there was limited protection against it.

The traditional approach also worked for a while and was quite popular. The perimeter network model is not entirely obsolete, and resource segmentation, which we will discuss in [Chapter 5, Secure Your Network](#), is still used today. However, the approach to having a wide-open internal network had a few significant flaws. Refer to the following figure:

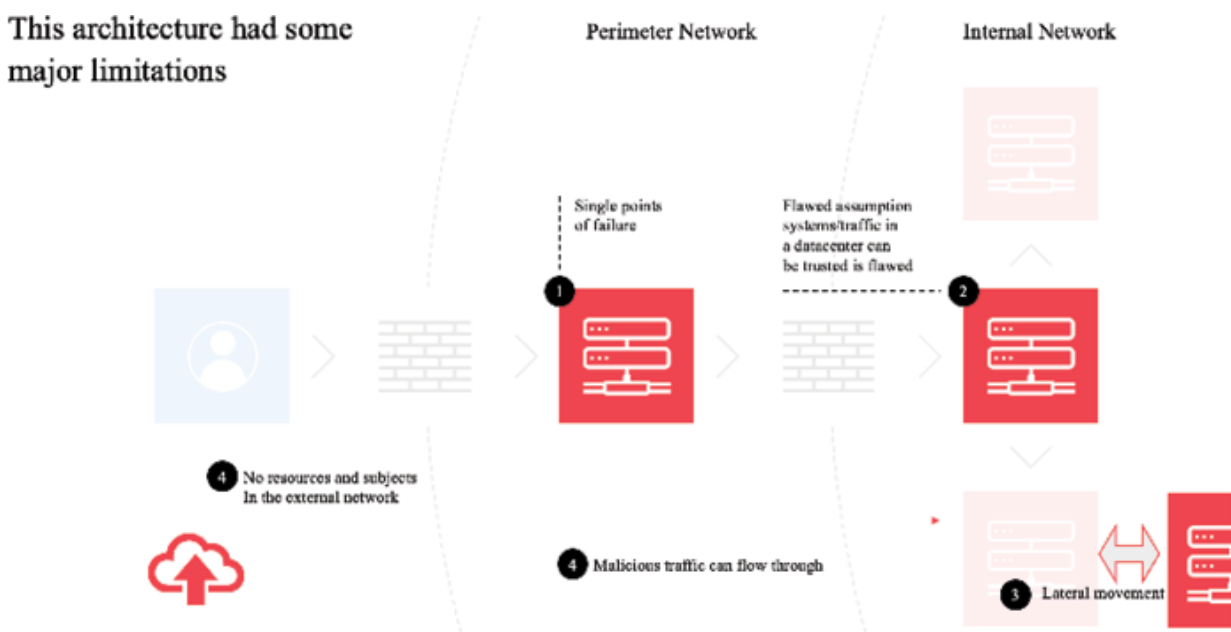


Figure 2.2: Limitations of traditional network architecture

The limitations of a traditional network architecture are as follows:

- **Single points of failure:** This model included two firewalls and reverse proxies to guard the network. If any of these devices were compromised, an intruder could freely move and gain access to confidential data and information. All the attackers had to do was find a way to break the fortress, compromise the services or proxies in the DMZ, and get past the firewalls since these same single points of failure were susceptible to misconfigurations.
- **Flawed assumption that traffic inside the data center can be trusted:** This assumption does not account for either insider threats, someone with malicious intent, or someone inadvertently doing something that could reveal confidential access externally. Blocking insider attacks. Due to a lack of encryption controls, data exchange with internal systems was also susceptible to MITM attacks.
- **Lateral movement:** There was also insufficient consideration to protect users' movement internally. If an outsider or insider gains access to a resource/service, they can use this access to explore other devices, networks, operating systems, and network hierarchies. There were no controls to prevent such movement of users, so an attacker could gain more insight into the devices and controls and potentially bypass them over time, gaining access to confidential information.
- **Resources and subjects outside the perimeter:** The traditional model was also not built with remote workers and external cloud services. Post-COVID-19, there has been a huge surge in people working remotely. This meant users of internal systems who used to be in the office and on internal networks could now work from any location, home, garage, or cafe. The model did not account for this surge of external devices trying to access internal resources. Equally, the model also did not account for resources that were deployed externally, especially cloud services or third-party services that the organization hosted on an external provider.
- **False sense of security.** All in all, the model provided a false sense of security. The proxy, which functions as a relay between the non-trusted (external) and trusted (internal) networks, is typically enabled by a web application firewall. However, it could not limit malicious traffic with growing resources, users, and traffic coming from different directions.

Case study: Target

Let us look at a case study to better understand Zero Trust:

- **Date:** December 2013
- **Country:** USA
- **Summary:** Target's **point of sale (POS)** machines, by which customers pay in store, had been compromised, and confidential data was leaked.
- **Impact:** The attackers made off with 70 million customers' PII and 40 million customers' credit card information. Following the attack, Target faced numerous lawsuits from different parties. Target announced in 2015 that the breach cost the company \$162 million in expenses related to the breach. It paid \$18.5 million to settle a multi-state lawsuit, \$10 million to settle a federal class-action lawsuit, \$39 million in response to a separate class-action lawsuit brought by several U.S. banks, and \$67 million to Visa alone. Target was estimated to have lost \$1 billion from the attack, only a tiny fraction of which cyberattack insurance covered. CEO *Gregg Steinhafel* and *Beth Jacob*, the company's chief information officer, resigned.^[4]
- **Root cause:** The attackers first found a weak device that an HVAC contractor firm that Target used, then moved laterally to gain access to this data.
- **The details of the attack are as follows:**

In late 2013, Target, the third-largest US retailer and a top 50 company in the Fortune 500, operating nearly 2,000 stores in the U.S. and Canada, in the middle of the holiday season, announced its POS machines, by which customers pay in-store, had been compromised. The attack occurred during the peak holiday shopping season between November 27 and December 15, 2013, and the following steps tell us how the attackers carried out this exploit:^[4]

1. **Research (weak link):** Target uses third-party contractor firms to run its stores and operations like many other companies. One of these attackers, *Fazio Mechanical Services*, an HVAC contractor, was using a free antimalware service with limited information security procedures.^[4]
2. **Initial access:** Using phishing emails, the attackers infected Fazio's

systems with *Citadel*, a widespread malware that can steal login information stored in computers' web browsers.

3. **Initial compromise:** This malware earned them access to credentials they signed into a contractor-facing billing portal (called *Ariba* ^[5]) hosted by Target. The portal was a web application where Fazio could upload legitimate documents, such as invoices.^[4]
4. **Exploitation:** Since Target did not block executable files on this solution, attackers further exploited the vulnerability by uploading a *web shell* named after a legitimate file, in this case, a PHP file named **xmlrpc.php** ^[5]. The web shell comprised a script with operating system commands that, once uploaded, gave the attackers remote access to Target's servers. The attackers then located Target's secure servers containing credit card numbers and their holders' personal information using Active Directory. First, the attackers would need domain admin privileges. The attackers took advantage of single sign-on, a standard design allowing administrators to log in to their user accounts once and automatically log in to applications. Single sign-on relies on hashes (which are part of the Windows **New Technology LAN Manager (NTLM)** authentication protocol), encoded output generated from passwords, and is stored on servers as long as administrator accounts are logged in. Unfortunately, only the hashes, not cleartext passwords, were needed to log in as an administrator. Thus, the attackers harvested the hashes and gained administrator privileges. With these privileges, they could create a separate domain admin account to maintain their network access even after the original account's password was inevitably reset. They even named this new account after a legitimate IT application to avoid suspicion.^[4]
5. **Lateral movement:** After mapping their targets, the attackers used a series of connected servers as a tunnel to bypass security measures (such as firewalls) and infiltrate the database. However, they found no credit card information when they arrived, as Target complied with **Payment Card Industry (PCI)** Security Standards, which required merchants to dispose of card numbers after purchases rather than storing them in a database. Instead, the attackers turned to the memory on POS machines.^[4] They launched a **random access**

memory (RAM) scraping, also called a POS attack, to steal customer credit card information by intruding into the RAM of the sales terminal.

6. **Actions and breach:** The attackers installed *Kaptoxa*, a type of malware that can disguise itself as legitimate antivirus software and break POS machines' firewalls on 40,000 of the 50,000 POS machines. Kaptoxa then stored customers' credit card information in machine memory as transactions were made; ultimately, the credit card information of 40 million customers was exfiltrated to an attacker-controlled server. This stolen data was sent to a controlled FTP account using an internal Windows FTP client.^[4]

Lessons

The lessons from the attack are as follows:

- **Identities and access:** The Target breach underscored the importance of robust authentication mechanisms. The attackers used multiple levels of credentials to breach the perimeter. Firstly, they used credentials for the billing system that they gained access to via a phishing attack. Next, they could also use local credentials to browse the active directory. Lastly, they could also create new admin credentials using stolen credentials. This emphasizes the importance of protecting credentials, stronger authentication protocols and access management.

One technique, **multi-factor authentication (MFA)**, requires multiple factors besides username and password, and it has recently emerged as a crucial tool in enhancing security. Use MFA on sensitive systems to reduce risks associated with credential stealing. MFA alone does not necessarily eliminate all the risks related to credential stealing, due to emerging attacks such as MFA fatigue or push bombing, where attackers repeatedly send MFA requests to users who can inadvertently approve fraudulent requests. Enforcing strong access controls to reduce user access to browse users, create credentials, etc., can also prevent attacks and slow down attackers.

- **Data security:** The attackers who gained database access could not retrieve confidential data as the data storage met PCI compliance policy. The specific PCI DSS requirement^[5] is *Do not store sensitive authentication data after authorization (even if encrypted). If sensitive*

authentication data is received, render all data unrecoverable upon completion of the authorization process. This is one of the strong security measures that slowed down the attackers, albeit not for long.

However, the attackers could use the POS systems to extract the card information. This brings us to the issue with magnetic strip cards. The data on the magnetic stripe cards is static and easy to copy and replicate. Instead, credit card chips and PINs are more secure than magnetic stripe cards as the chip and PIN cards generate a unique cryptographic code for each transaction, making it harder to replicate. The Target breach accelerated the adoption of more secure payment technologies, particularly chip and PIN cards, also triggered by the **Europay, Mastercard, Visa (EMV)** liability shift that would hold merchants liable for credit card fraud if they did not upgrade to EMV-compliant payment terminals.

Chip and PIN cards offer several security benefits over traditional magnetic stripe cards, mentioned as follows:

- Unique cryptographic codes are generated for each transaction. As a result, static card details can no longer be easily extracted.
- Difficulty in replicating or cloning the card as the generated codes are unique and temporary (per transaction).
- A requirement for a PIN as part of the transaction adds an extra layer of authentication. This is akin to MFA in the authentication process.

These features make it significantly harder for attackers to use stolen card data for fraudulent transactions, reducing the potential impact of data breaches involving payment card information.

In short, not storing or encrypting sensitive customer data and employing short-lived tokens or codes for crucial transactions improves data security and limits attackers.

- **Network segregation and segmentation:** Once the attackers accessed the server, proper network segmentation could have slowed them down. Although Target had firewalls, the attackers could create a series of tunnels to travel across the network. Adequate network segmentation could have contained the breach and prevented the attackers from

moving freely within the network. Network segregation involves isolating different parts of the network to control access and limit the potential spread of threats. The Target breach highlighted that internal traffic alone cannot be trusted or tested as secure. Organizations should implement the following controls:

- Separate networks for different departments or functions.
 - Isolate networks for sensitive data and systems from the rest of the infrastructure and networks.
 - Implement micro-segmentation and restrict access between network segments based on their specific requirements.
 - These measures can significantly reduce the potential impact of a breach and limit an attacker's ability to move laterally within the network, slowing them down and reducing the overall risk to an organization.
- **Vulnerability management:** The Target breach started with a weak link in one of the third-party contractors. This highlights the importance of comprehensive vulnerability management, especially concerning the supply chain, third-party access, and risk management. This means understanding and minimizing vulnerabilities posed by network and data access by vendors and the greater supply chain, not just their corporate network and data, and managing and securing connections with third-party vendors, especially those with network access.

Organizations must extend the scope of their cybersecurity programs to include the following:

- Careful vetting of vendors with network access.
- Implementation of strict access and least privilege access controls for external parties.
- Monitoring and auditing of third-party access to systems and data.
- Regular assessments of third-party/vendor security practices.
- Continuous evaluation of the security posture of the entire supply chain.
- Regular security audits of vendor connections and practices.

- Clear security requirements and expectations in vendor contracts.
- **Continuous monitoring and security awareness:** The Target breach demonstrated the critical importance of ongoing vigilance, security monitoring, and cybersecurity awareness among everyone involved. Enhanced security monitoring and intrusion detection systems could have alerted Target to suspicious activity earlier, potentially mitigating the extent of the damage. Target could have monitored user lists for additions, escalation of privileges, and abnormal queries to Active Directory/**Lightweight Directory Access Protocol (LDAP)**^[5].

Target did not notify customers of the breach until December 19, four days after identifying it. In his first interview after the breach, *Gregg Steinhafel*, who served as Chairman, President, and CEO, justified the delay. He explained that the first day *was about making our environment safe and secure. We worked very hard on that. And by 6:00 at night, our environment was safe and secure. We eliminated the malware and the access points. And, so, we were very confident that coming into Monday, guests could come to Target and shop confidently with no risk.* However, the drive to enable customers to continue shopping during the holiday season would prove shortsighted, as even after Target removed the immediate cyber threat, many customers' confidence in the company would be slow to rebound.^[4]

It shows how corporate crisis management programs must emphasize response and recovery as key elements and effective communication with customers and external stakeholders, such as the press, regulators, and law enforcement.

Let us look at the key measures that can help detect and respond to such threats proactively:

- Implementing robust monitoring systems can help detect and respond to threats more quickly.
- Real-time analysis of network traffic and system logs.
- Use of advanced intrusion detection and prevention systems.
- Implementation of **security information and event management (SIEM)** solutions.
- Regular vulnerability scans and penetration testing.

Fostering a culture of security awareness throughout the organization is equally crucial, including with third parties; this requires:

- Regular security training for all employees.
- Simulated phishing exercises are used to test and improve awareness; in Target's case, the third-party provider could have benefited from such awareness exercises.
- Clear communication of security policies and best practices.
- Encourage the reporting of suspicious activities or potential security incidents.

Zero Trust architecture principles

So, what are the core principles of a Zero Trust architecture? We will discuss the details of Zero Trust networks separately in [Chapter 5, Secure Your Network](#). In this chapter, we will focus on high-level inspiration and architectural principles. These principles have been broadly adopted from the NIST.SP.800-207 publication^[3]. Some principles have been consolidated for simplicity.

In addition, as discussed in the previous chapter, there is more than just securing the network. The *Reflections on Trusting Trust* paper by Ken Thompson raises additional questions about trusting code, the libraries we use, and the code we write. Some of the key takeaways from that paper include:

- No amount of source-level verification or scrutiny will protect you from using untrusted code.
- Trusting software goes beyond trusting the source code; it requires trust in the compiler, the tools used to build the compiler and the source code, and potentially all dependent tools and libraries.
- As the level of the programming gets lower, these bugs will become more complex and more challenging to detect. A well-installed microcode bug will be almost impossible to detect.

Terms and definitions

The following are the key terms and their definitions used in this subsection:

S. no.	Term	Definition
1	Resource	Any service or data source that a user or a device wants to access.
2	User	A human user, malicious or not, who is trying to access a resource.
3	Application (app)	Applications are any software systems that are trying to access resources. Resources can also be applications; however, this section typically treats applications as actors trying to access a resource.
4	Device	The user typically uses an asset that can connect to a network to gain access or perform an operation, such as a desktop, laptop, mobile device, or printer.
5	Policy	Policy can mean multiple things; Zero Trust can be considered a policy. However, in the context of Zero Trust, a set of policies is typically applied to every device or user. In that context, policy is a set of conditions that would be evaluated before granting access to a resource.

Table 2.1: Terms and definitions in Zero Trust

Some of the following principles have been extended further to include trust beyond the network:

- **Verify everything:** The core principle of Zero Trust is to assume the network is hostile and threats exist around you. This implies organizations should not automatically trust anything inside or outside their perimeters and instead must verify anything and everything trying to connect to their systems before granting access. External and internal threats always exist, so you do not differentiate between internal and external networks.^[3]

This rule can include the entire landscape in which an organization and its users operate. So, it is not just hostile networks but people building software, providing it, and using it. Your competitors and state actors may be interested in your actions and want to use the information for their benefit. You also assume Zero Trust in the data and software you create or receive from others and verify it before its usage.

- **Implication:** Assume Zero Trust in the network, software, data and all actors (internal and external).
- **Secure all flows:** This is an extension of the first principle. Since you assume everything is hostile, you must secure every flow and communication channel. Access requests from internal and external

networks must meet the same security requirements. Network locality is not sufficient for deciding trust, so even if an attacker gains access, they would be subject to similar controls. All communication should happen securely, confidentiality and integrity should be protected, and source authentication should be provided.^[3]

In addition to network flows, this same principle can be applied to other information and data flows. For instance, software provided by an external third party to you also should be subject to the same level of security as software built internally and delivered to a different environment.

- **Implication:** Every device, user, app, and flow is authenticated and authorized.
- **Least privilege access:** Beyond evaluating trust in the requestor of the resource, access should be granted with the least privileges and for the task's duration. This means that access is only provided for a shorter duration for the specific transaction and may not occur directly before initiating a session or performing a transaction with a resource. Also, authentication and authorization to one resource will not automatically grant access to a different resource. If the user needs to access a second resource, they must authenticate and authorize again to initiate a session.^[3]
 - **Implication:** Access is least privileged and time limited.
- **Dynamic policies:** Since much more complex requirements now govern access, Zero Trust introduces the concept of dynamic policies. These requirements must be dynamic and calculated from as many sources as possible (such as device compliance, geolocation, and user behavior analytics) before granting access. Every organization should protect its resources by defining its resources, who its members are (ability to authenticate users from a federated community), and what access those members need.^[3]
 - **Implication:** A Zero Trust solution requires dynamic policies that adapt based on user activity, behaviors and are applied in real-time.
- **Monitor and measure security posture:** Organizations embracing Zero

Trust must evaluate resource requests and establish continuous diagnostics and mitigation to monitor the granted access state of devices and applications. This intel will be helpful to patch and update policies and vulnerabilities. This process allows differentiating assets (users, devices, apps) that are subverted (have vulnerabilities) from those in a more secure state, helping build insight into the current security posture of the organizations across all owned and associated assets.^[3]

- **Implication:** A robust monitoring and reporting system should be in place to provide actionable data.
- **Continuously improve security posture:** Once you have enough information being monitored and measured, every organization should constantly use the information collected about the current state of assets, network infrastructure, and communications to improve the security posture. Updates can include refining the policies applied and improving security protocols on resources. Continual monitoring with possible reauthentication and reauthorization occurs throughout user transactions, as defined and enforced by policy (e.g., time-based, new resource requested, resource modification, anomalous subject activity detected) that strives to achieve a balance of security, availability, usability, and cost-efficiency.^[3]

Continuous improvement is one of the critical parts of the overall process that often gets overlooked. If an organization does not monitor and improve its security posture over time, vulnerabilities can creep up, and attackers can exploit them to deliver a successful attack and compromise data. Improving security posture also applies to software development processes, third-party review and assurance. Another essential part of continuous improvement is training employees and making them aware of new and emerging threats and how to counter them.

- **Implication:** Stay vigilant and improve continuously.

Never trust, always verify

Zero Trust can be distilled down to a single phrase *never to trust, always verify*, dynamically and continuously. [Figure 2.3](#) shows how these principles

could be applied to a typical flow. It would include the following key steps:

1. A user, device or application tries to connect to a resource. We will always assume the user or the network they come from is hostile, and we will verify everything. The user could be an external or an application trying to access an internal resource or an internal user trying to do the same.
2. Regardless of their location, we will secure every information flow. The user might be trying to access a service deployed internally, or it can be a third party providing a service or software to the organization. Regardless of the location of the source, we will use the most secure means of communication. The details of the secure modes are covered in later chapters, but some examples include using encryption of the data and a secure authentication mechanism.
3. The request is verified, and access is granted by applying the least privilege principle per session. So, if a user is trying to access a service and only requires the ability to read part of the system or data, they are only provided access to that specific dataset and not the rest. Various approaches exist, such as role-based and attribute-based access controls. In addition, grant access for a limited duration that is sufficient for the user must complete their action.
4. Dynamic policy determines user access, examining the user and their behavioral and environmental attributes. Suppose a user tries connecting from two locations within a short duration. In that case, they may be subject to different levels of due diligence than someone trying to access from the exact location.
5. The organization will monitor and measure usage and current security posture over time as users, devices, and apps continue to access resources. The security posture shows the users, the resources they access, and their current security compliance (e.g., based on the software versions they use).
6. Finally, organizations can use the data and insights gathered from security posture monitoring to improve their security posture continuously. Continuous improvements help close any security vulnerabilities or loopholes over time.

Refer to the following figure:

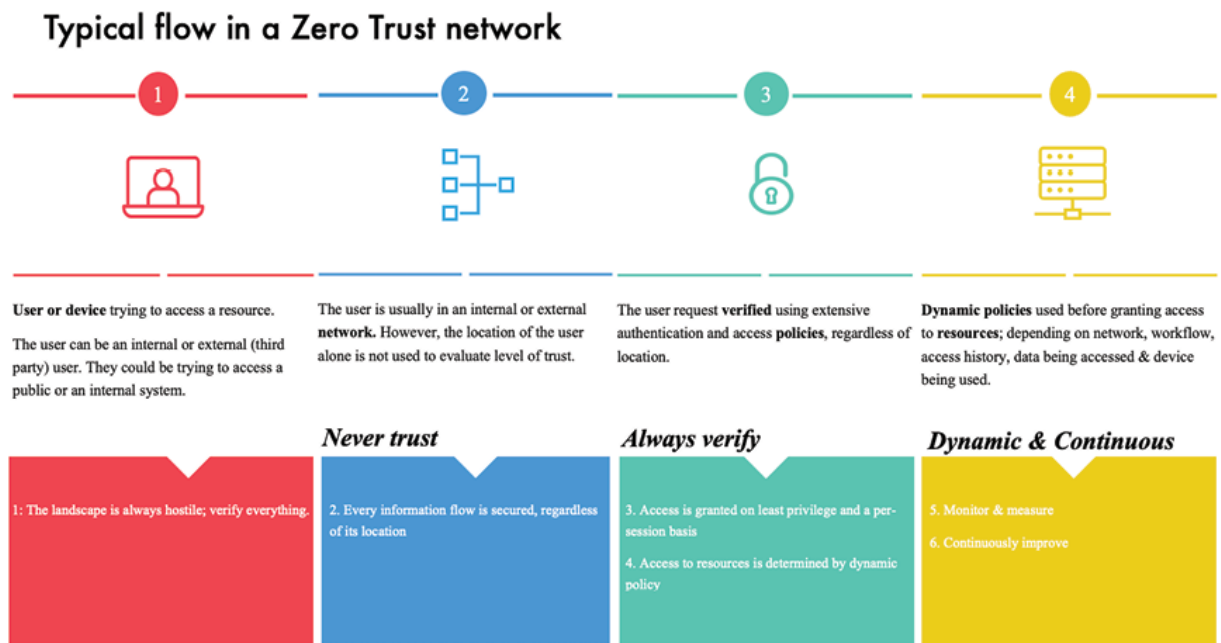


Figure 2.3: Zero Trust typical flow

Key objectives of security

Before we cover the core pillars of security, we should also look at some of the core objectives of security. One of the simplest ways to understand the key objectives of security is shown in [Figure 2.4](#), called the **CIA triad**. It is an information security model designed to protect sensitive information from data breaches and compromises. The model is composed of **confidentiality, integrity, and availability**. It provides a comprehensive framework for safeguarding sensitive data and maintaining robust security practices. By addressing these three fundamental principles, organizations can effectively identify vulnerabilities, implement appropriate security measures, and create a multi-layered defense strategy against evolving cyber threats. The CIA triad not only guides the development of security policies but also helps evaluate incident response plans and assess the effectiveness of existing security controls, making it an indispensable tool in the modern cybersecurity landscape. This model will also help you understand the key drivers behind the pillars we cover in the next section. Refer to the following figure:



Figure 2.4: CIA triad model

- **Confidentiality:** It is the first element and usually applies to confidential information such as customers' personal information, contact details, bank accounts, and health information. Confidentiality ensures that information is accessible only to authorized individuals, preventing unauthorized access and potential data breaches. Confidentiality extends beyond personal data to sensitive information, such as an organization's intellectual property, corporate records, trade secrets, and any information that should not be available in the public domain. There could also be multiple sensitivity levels, and data may have to be segregated based on a classification; we will cover classification in more detail later.
- **Integrity:** It focuses on maintaining the accuracy and consistency of data throughout its lifecycle, ensuring that information remains trustworthy and unaltered. For instance, on a banking app, the key information they have on the record about you as a customer and the payments, bank account details, payments, and balances are all critical. Every bank needs to ensure this information is accurate and consistent. Any incorrect data could lead to the inability of end users to access their funds or receive the money they need. The bank must also ensure that as transactions happen daily, the information is updated, and the calculations, such as interests and balances, are performed accurately.

and consistently. This can also drive technology choices based on the integrity requirements of the system in question.

- **Availability:** It guarantees that data and systems are accessible to authorized users (customers or internal) when needed, supporting uninterrupted business operations. Regarding health or medical information, data availability could mean life and death scenarios, as was clear from the *WannaCry* ransomware attack in 2017 that hit hospitals and primary care organizations in the UK and led to cancelations of emergency operations and routine appointments. Thus, organizations must ensure they can tolerate power outages and disasters like floods and protect themselves from cybercriminals. In many ransomware attacks, the attackers encrypt sensitive data and ask for ransoms to provide access to it, causing outages. In most cases, availability requires sufficient redundancy in the systems (copies of data, multiple servers running the application), regular backups, and testing of recovery procedures.

Core pillars of Zero Trust

Now that you understand the core principles of Zero Trust architecture let us examine how you apply them. The Zero Trust approach to implementing secure systems is comprised of six core pillars. These six pillars help simplify your approach to applying security in daily life. They apply to every organization and following them will help you build and operate secure systems and improve the safety, reliability, and security of the organization's security posture.

Figure 2.5 is a good way to visualize these pillars is using an ancient fortress standing resolutely against time and threats. Castle architecture dates to Europe's 9th and 10th centuries, and its primary function was defense. The architecture comprised a fortified tower that was the last line of defense and the lord's living quarters of the king. Around the tower were moats, drawbridges, and portcullises that provided multiple layers of protection.

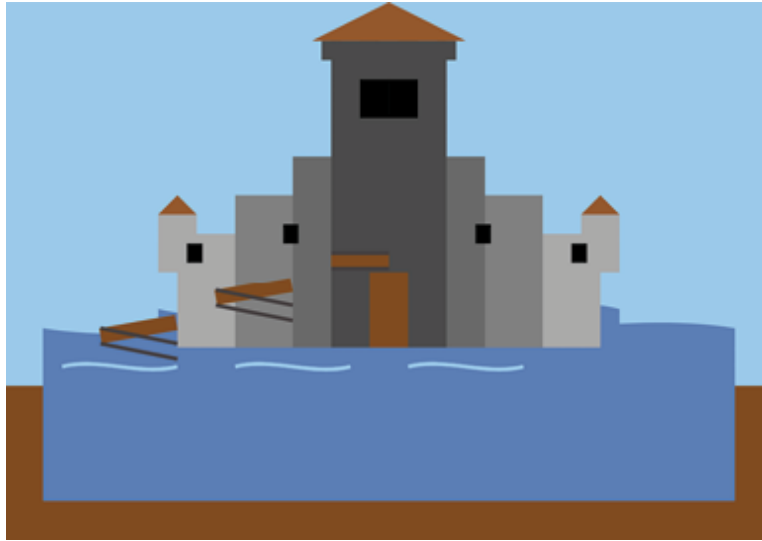


Figure 2.5: Ancient fortress with multiple layers of defense

The castle shows multiple layers of such a defense system. The system is made of three layers, mentioned as follows:

- **Outer layer:** It includes a wide outer moat, first drawbridge, outer defensive wall with guard towers, arrow slits, and windows for surveillance.
- **Middle layer:** Secondary moat, second drawbridge, taller middle wall, and additional defensive position.
- **Inner layer:** Inner keep with the highest walls, final drawbridge, central tower with battlements, and a fortified entrance.

The pillars of a Zero Trust architecture we cover in this book also offer a similar defense in-depth approach akin to moat-based protection for castles, given as follows:

- **Outer layer:** Includes securing the code, build and deploy, and securing identities and access.
- **Middle layer:** Secure your network and data.
- **Inner layer:** Managing vulnerabilities, continuous monitoring and improvements.

Figure 2.6 shows these different layers of security applied to security. Similar to the castle architecture of moats and other defenses, it also relies on multiple layers as depicted by the rings. Each ring providing distinct yet interconnected protection.

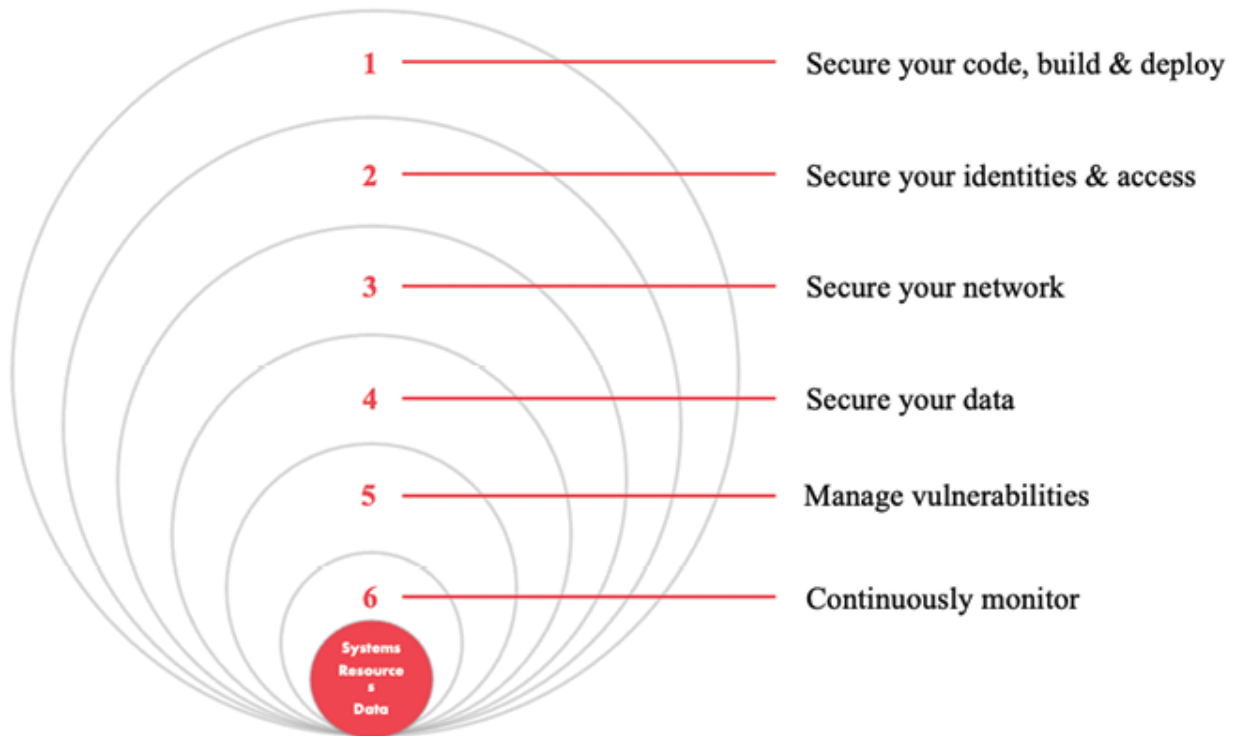


Figure 2.6: Zero Trust layers model of defense

At its heart lies the keep, where the crown jewels rest; this is the sensitive data and resources. Surrounding this inner sanctum are various layers of protection. Just as no single wall could guarantee the fortress's safety, Zero Trust security acknowledges that protection comes through layers, each reinforcing the others, creating a defense that is both flexible and formidable. Like peeling an onion, each layer of security reveals another, forcing potential adversaries to overcome multiple independent challenges, ensuring that a breach of one layer does not compromise the entire system. These layers of security, or pillars as we reference in this book, are listed as follows:

1. Secure your code, build and deploy.
2. Secure your identities.
3. Secure your network.
4. Secure your data and APIs.
5. Manage vulnerabilities.
6. Continuous monitoring.

Let us look at each of them in detail.

Secure your code, build and deploy

Ken Thompson's paper clarifies that you cannot trust code you did not create. However, even when people write their software, they rely on multiple external libraries, third-party systems and software. Hardly any system is built from scratch without any external dependencies. Consider a modern self-driving car: its codebase might leverage TensorFlow for neural networks processing camera feeds, **robot operating system (ROS)** to integrate various components to build a complex system, and AUTOSAR-compliant components for essential vehicle control. Each dependency can introduce a potential vulnerability. Imagine a compromised computer vision library misidentifying stop signs or a tampered vehicle dynamics module affecting braking behavior. For instance, *Tesla's Autopilot*^[6] relies on PyTorch for perception tasks, while some companies integrate proprietary systems with open-source components for path planning. This intricate set of dependencies means securing just the in-house code is insufficient.

Modern autonomous vehicle manufacturers must implement rigorous supply chain security measures. These include cryptographically signing all software components, maintaining detailed dependency graphs (including transitive dependencies), continuously monitoring upstream vulnerabilities, and enforcing strict version control with reproducible builds. Do not worry if these measures do not make sense; we will discuss them later.

Companies like *GitHub*, *GitLab* and many others have developed sophisticated CI/CD solutions that compile code and validate each software component against safety-critical requirements. These automated pipelines typically involve many checks to ensure that even seemingly innocent library updates do not compromise the safety of the system being built.

There are four high-level steps to the entire code deployment process, and Zero Trust security has to be integrated into each of the following steps:

1. Code is where engineers write code and unit test their software.
2. Build involves coding an artefact being built or constructed based on the code.
3. Scan, test, and verify the generated artefact(s) or software deliverables.
4. Deploy is the last step, where the software is released for general consumption.

Secure your identities and access

The second pillar is securing **identity and access management (IAM)**. In today's world, this is the most important aspect of protecting systems; some might say this is the outermost layer or the moat that will protect your confidential data and systems. Let us take two examples where identity and access control are important: autonomous vehicles and banking systems. Both present their challenges around safety and security.

Imagine a self-driving taxi service like *Waymo* or *Cruise*: the vehicle must simultaneously manage multiple identity layers, from the AI system's authentication to cloud services for navigation and traffic data to passenger verification for ride access. We need to be able to stop intruders from breaking into these autonomous vehicles and stop unauthorized passengers from using the service.

In the banking world, most banking apps now require secure authentication methods, including biometric verification via phone geolocation. Large transfers typically also require additional modes of authentication and supporting evidence. Some security measures also examine the user's behavior and recent activity before making decisions and granting access.

This pillar of security comprises multiple areas:

- Single sign-on and identity verification.
- MFA.
- Mutual authentication for applications.
- Least privilege access controls, such as **role-based access control (RBAC)**, **attribute-based access control (ABAC)**, and **policy-based access control (PBAC)**.
- Short-lived access.

Secure your network

Although the concept of Zero Trust applies mostly to network and access controls around it, within this book, the network security area will focus purely on networking security controls. This is because the other areas are covered under other pillars, such as identity and access, data and monitoring.

Many within the workforce work remotely, and people travel all the time, working from home, cafes, and airports. This requires a hyper-vigilant

immune system, where every data packet is treated as potentially hostile, regardless of origin. Applying this to smart or autonomous vehicles, a car is parked in its owner's garage, connecting to its home Wi-Fi, and should be subject to the same Zero Trust scrutiny as if it were parked in a hostile territory.

The following are the key areas of network security:

- Network design and segmentation approaches.
- Network access control techniques using firewalls.
- Secure communication using transport level security.
- Threat detection and prevention to monitor network threats and intrusions.

Secure your data and APIs

Data is the most important commodity in today's digital landscape. Securing data and APIs is paramount for organizations to protect sensitive information and maintain trust with their users. A comprehensive approach to data and API security encompasses several critical components. It is focused on protecting sensitive information, both at rest and in transit. It employs various measures to ensure the confidentiality, integrity, and availability of data, including:

- Encryption of data at rest to protect data stored inside databases and object stores.
- Encryption of data in transit when the data is being transmitted over communication channels, mostly to prevent interception or tampering by intermediaries.
- Access controls limit access to users who need access for the duration they need. This is mostly covered under the identity and access pillar.
- Backup and recovery of data in case of breaches and disasters.

Organizations must implement security measures and create a multi-layered defense strategy to protect their valuable data and APIs. This requires the following key controls and systems to be in place:

- Data encryption is the foundation, ensuring that information remains confidential and tamper-proof during transit and at rest.
- Key management systems play a crucial role in safeguarding encryption

keys, allowing for secure generation, storage, and rotation of these vital assets.

- **Public key infrastructure (PKI)** provides a robust framework for authenticating users, devices, and services through the issuance and management of digital certificates.
- Secrets management is essential for securely storing and controlling access to sensitive credentials, API keys, and other confidential information, preventing unauthorized access and potential breaches.
- Backups of critical data and regular testing of recovery procedures.

Manage vulnerabilities

The first four pillars are focused on measures during the design and implementation of solutions. The next two pillars are more about finding issues, remediating, and being reactive. However, they form a critical part of implementing a Zero Trust architecture.

Vulnerability is a weakness in a system that an attacker can exploit. In Target's case, this was the ability to upload executables from their billing software. Modern-day systems are very complex, with multiple servers and applications running complex architectures. If left unchecked, it is easy for a vulnerability to go unnoticed for a long period, enough for an attacker to utilize and cause harm.

Traditionally, vulnerability management was focused on scanning servers and patching outdated systems. Modern vulnerability management extends beyond this traditional scan-and-patch cycle; organizations now employ continuous security testing, chaos engineering principles, and automated remediation. For instance, cloud-native systems might automatically spawn patched container instances while gracefully terminating vulnerable ones, similar to how our bodies replace compromised or dead cells. The key challenge lies in balancing security with availability: patching a bank's transaction processing system vulnerability or an autonomous vehicle's software requires careful orchestration to prevent service disruption while maintaining safety. This necessitates sophisticated orchestration systems, automated deployment and release processes, and coordinating patches across complex distributed systems while maintaining integrity and uptime.

Vulnerability management is comprised of a 4-step process:

1. Discover vulnerabilities through scanning, network discovery, etc.
2. Prioritize and assess the risk of the vulnerabilities to focus on the most critical ones.
3. Remediation of the prioritized vulnerabilities.
4. Verification of the remediation to ensure it has resolved the underlying problem.

Continuously monitor and improve

The last pillar is the feedback loop of the Zero Trust model. Once you have secured all that you can, and also identified and managed your vulnerabilities, it is essential to analyze how the process is performing, get data about potential threats or compromises, learn from it, and improve. Continuous monitoring is like the dashboard of your smart car or smart device. It shows you the system's health, its performance, and where you might need to improve or change. This area typically spans development and security operations teams.

Modern security operations constantly monitor multiple data streams for potential threats. They receive data from across the organization's network and infrastructure estate. The data set typically includes policy violations, vulnerabilities, suspected compromises, and attacks. The role of the operations team is to review these, work towards resolving them, and improve processes. **Security operations centers (SOC)** span millions of telemetry signals for many large organizations. A good continuous monitoring system will correlate seemingly unrelated events, an unusual authentication pattern in Asia, abnormal sensor data in Europe, and suspicious API calls in North America, to identify potential attack patterns. Like a bank's fraud detection system that monitors transactions 24/7 to flag fraudulent transactions, modern SOCs employ AI-driven SIEM platforms that can detect subtle anomalies.

Continuous monitoring is typically made up of the following:

- Tools and infrastructure implementation with log aggregation, real-time analytics and threat intelligence platforms, and SIEM platforms.
- Monitoring strategy for protective monitoring use cases, alert setup and tuning, response automation, metrics, and KPIs.

- Detection capabilities include correlation, pattern recognition, and anomaly detection.
- Incident response, including playbook development triage procedures. Investigation workflows, containment strategies, and recovery procedures.

Conclusion

This chapter introduced the core principles of a Zero Trust architecture as formalized by NIST SP 800-207. We also covered the broader framework this book will focus on, based on the six key pillars of securing code, identities, network, and data: managing vulnerabilities and monitoring. In the next chapter, we will discuss the first principle of securing your code, build, and deployment processes.

Points to remember

Zero Trust architecture (formalized by NIST) relies on the core concept of *never trust, always verify*, and is made of the following key principles:

- The landscape is always hostile; verify everything.
- Every information flow is secured, regardless of its location.
- Grant access on least privilege and a per-session basis.
- Determine access to resources by a dynamic policy.
- Monitor and measure the integrity and security posture.
- Continuously improve security posture.

The book will focus on implementing a broader framework based on the following key pillars:

- Secure your code, build and deploy.
- Secure your identities and access.
- Secure your network.
- Secure your data.
- Manage vulnerabilities.
- Continuous monitoring.

References

1. <https://www.nist.gov/about-nist>
2. <https://techtarget.com/searchsecurity/definition/DMZ>
3. <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-207.pdf>
4. <https://www.sipa.columbia.edu/sites/default/files/2022-11/Target%20Final.pdf>
5. <https://aroundcyber.wordpress.com/wp-content/uploads/2014/09/aorato-target-report.pdf>
6. <https://www.ultralytics.com/glossary/pytorch>

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



CHAPTER 3

Secure Your Code, Build and Deploy

Introduction

Developing, validating, and deploying secure software is critical for every engineer and leader in technology companies today. It is also fraught with a lot of challenges. At the heart of this endeavor is the principle captured by *Ken Thompson* in his pivotal paper, *Reflections on Trusting Trust* ^[1]. Thompson underscores a fundamental truth: *You can't trust code you did not create*. In today's complex digital landscape, however, this notion must be extended to acknowledge that you cannot even trust the code you created yourself. This assertion is not merely theoretical but underscored by modern software development's reality, where code integrity is constantly at risk.

The sophistication of many of the recent supply chain attacks exemplifies this vulnerability. These attacks have become increasingly subtle and devastating, capable of infiltrating and compromising even the most secure systems by manipulating the code and tools trusted within the build and deployment processes. Such threats highlight the imperative for robust security practices throughout the software development lifecycle, which is the premise of this chapter.

Structure

The following topics are covered in the chapter:

- Case study: SolarWinds hack
- Case study: MOVEit vulnerability
- Core concepts
- Secure requirements and design
- Secure coding
- Secure build
- Secure scan and test
- Secure deploy
- Continuous integration and deployment monitoring
- Tools and plugins
- Guidelines and checklists

Objectives

This chapter discusses strategies for building a more secure code creation, testing, and deployment environment. This chapter illustrates the need for vigilance and innovation to maintain security during the requirements and design phases and protect the code that the code engineers write. We then look at securing the build and deployment tools and processes. We will explore methodologies to safeguard against threats and ensure the trustworthiness of software development and deployment systems.

Case study: SolarWinds hack

Let us look at a case study for better understanding:

- **Overview:**
 - **Date:** March 2020
 - **Countries:** Multiple
 - **Summary:** The attack is one of the most comprehensive global

supply chain attacks affecting organizations. Cybersecurity firm *FireEye* was the first to announce that it had fallen victim to a sophisticated hacker attack attributed to a backdoor installed via *SolarWinds Orion* software. SolarWinds said the backdoor created by attackers was distributed via infected software updates and delivered to as many as 18,000 organizations worldwide. ^[2]

- **Impact:** More than 18,000 SolarWinds customers had applied the SUNBURST update, allowing the remote access trojan to infect all customer systems and networks. Among notable victims, the US departments of health, treasury, and state were affected by this attack. Private companies like *FireEye*, *Intel*, *Cisco*, and *Microsoft* are affected by this malware. ^[4]

A lot of personal and sensitive information was accessed during the SolarWinds cyberattack. Many hacked networks could still be unsecured or even modified for easier entry. Although hackers have been flushed out of most networks, primarily the government, it is impossible to be 100% sure they have been flushed out of all. ^[3]

The estimated total costs of the cleanup, including the costs from each government entity affected and those affected in the private sector, is around \$100 billion. ^[3]

- **Root cause(s):** The attackers accessed the build system and launched a supply chain attack. A **dynamic link library (DLL)** is a digitally signed component of the Orion software framework containing a backdoor that communicates via **Hypertext Transfer Protocol (HTTP)** to third-party servers. Once the update is installed, the legitimate executable will load the malicious DLL and enable the sharing of confidential information. ^[2] Some of the leading causes are:
 - Insufficient build pipeline security and weak access controls.
 - Lack of adequate review of artifacts.
 - Limited monitoring of the build process.

In the *Learnings* section, we will cover some ways this could be prevented, such as robust build, code, and deploy processes, build system hardening, access monitoring, and least privilege access.

Detailed timeline

Most of us would remember 2020 as the peak of the COVID-19 pandemic, stay-at-home orders, lockdowns, and news of near and dear ones suffering. It was quite a dark period in mankind's history in recent years. Many changes came out of this period; some, like hybrid work, were positive. At the same time, the adoption of digital solutions increased manifold. This meant attackers now had more access and a much larger estate and companies to go after. State threat actors are one of the hardest to protect against. In 2020, the state-sanctioned hacker groups within Russia carried out a significant attack. The main goal of state actors is to gather valuable intel from state departments, crippling economies, and operations that are essential to maintain security and stability in a nation-state.^[2] Cyberattacks are much safer than spies on the ground, as they are hard to trace, much cheaper, and can be highly sophisticated and more straightforward to scale.

SolarWinds is an American company headquartered in *Austin, Texas*. The company provides SaaS solutions for IT infrastructure, network, and infrastructure monitoring. Since it provides software for IT infrastructure management, its software typically has complete access to customer data, logs, and workflow details. Some customers include large organizations such as *Amazon, Morgan Stanley, Walmart, and McDonald*. SolarWinds offers an IT performance management and monitoring system called *Orion*. Customers use the Orion platform worldwide; over 30,000 public and private organizations use the Orion network management system.^[4] To enhance its effectiveness, Orion has access to customer system performance logs and data, making it a lucrative target for hackers. Orion was at the core of this supply chain attack.

In early December 2020, the cybersecurity firm FireEye announced that it had fallen victim to a sophisticated hacker attack. The perpetrators had stolen, among other things, penetration testing tools that the company uses to test its customers' IT security. However, it was clear they were not the only victim; thousands of other organizations had also been compromised via backdoors injected into SolarWinds Orion software.^[2]

Hackers discovered that many high-profile organizations used SolarWinds' IT tools. The customer list published on their website had more than 300,000 customers worldwide (although not all were impacted by the attack),

including all branches of the US military, central US federal government agencies and US Fortune 500 companies. Orion network management software for large enterprise-class networks was often configured with extensive administrative rights.^[2]

Since September 2019, hackers have had access to SolarWinds' internal network. They planted a backdoor into its software update infrastructure, containing SUNSPOT malware by compromising the build system and creating a temporary update file. When SolarWinds' software developers built a finished product from the source code, SUNSPOT replaced a single file with the hackers' copy, to which they could add extra code. This allowed the hackers access to SolarWinds' Orion platform software and to inject a backdoor. SUNSPOT monitored SolarWinds' Microsoft Visual Studio development tools, which a range of companies use to develop software. The backdoor gave hackers access and impersonated users and accounts of victim organizations. The malware could also access system files and blend in with legitimate SolarWinds activity without detection, even by antivirus software.^[4]

The first infected update was digitally signed on 24 March 2020 and distributed to SolarWinds customers on 26 March, whereupon thousands of private companies and government agencies worldwide began to download the infected software updates, unaware of the injected backdoor.^[2]

In early April 2020, the first SUNBURST backdoors started to communicate with the hacker-controlled servers after an initial dormant period of up to two weeks.^[2] SUNBURST was designed to imitate legitimate network traffic but send hidden information to the hackers, such as the identity of organizations that had downloaded the backdoor. Between March and December 2020, the hackers used the custom SUNBURST backdoor to target specific victims. Only a tiny fraction of the approx. 18,000 compromised organizations saw follow-on hacking activity. However, the selected victims were attacked with an extensive arsenal of custom malware and tailored operations.

One of the victims, FireEye, had its systems compromised, and a custom loader called TEARDROP delivered a configured version of a pen testing tool, Cobalt Strike, enabling hackers to access information about FireEye customers and their penetration testing tools.

When the hackers gained access to Microsoft, they exploited the SUNBURST backdoor to access Microsoft's source code repositories via an internal account. However, according to Microsoft, it does not rely on the secrecy of source code for the security of its product, thus limiting the effect of the hackers' insight.^[2]

There is a lot of uncertainty regarding how deep this attack goes and what information was accessed. However, the focus of the attack was espionage and information gathering. While many companies and departments reported signs of data breaches and malware, it is still hard to determine what information was accessed and stolen, if any. The total cost of the cleanup, including the fees from each government entity affected and those affected in the private sector, is predicted to be \$100 billion, though true costs are unknown.^[3]

Since the hack was discovered, SolarWinds recommended that customers update their existing Orion platform and release patches for the malware and other potential vulnerabilities discovered.^[4] Following is the sequence of the rest of the events:

- **September 2019:** Threat actors gain unauthorized access to the SolarWinds network.
- **October 2019:** Threat actors test initial code injection into Orion.
- **February 20, 2020:** A malicious code known as SUNBURST was injected into Orion.
- **March 26, 2020:** SolarWinds unknowingly starts sending out Orion software updates with hacked code.

Learnings

The lessons from the attack are as follows:

- **Robust code, build, and deploy processes:** Attackers compromised the SolarWinds Orion platform's software build and code-signing infrastructure. The source code of the affected library was directly modified to include malicious backdoor code, which was compiled, signed, and delivered through the existing software patch release management system. There are claims that their software update server credentials were exposed in a public GitHub repo^[7], which highlights the importance of secret management and scanning. Although this has

not been ascertained as the root cause, it highlights the importance of having robust code, build, and deploy processes, including hardening the build system, monitoring access, signing artifacts, and least privilege access.

- **Need for software bill of materials (SBOM):** Following this attack, the Biden administration mandated the need for SBOMs by an executive order issued in May 2021. SBOM means a formal record containing the details and supply chain relationships of various components used in building software. Software developers and vendors often create products by assembling existing open-source and commercial software components. The SBOM enumerates these components in a product. It is analogous to a list of ingredients on food packaging. An SBOM is helpful to those who develop or manufacture, select, purchase, and operate software. Developers often use available open-source and third-party software components to create a product; an SBOM allows the builder to ensure those components are up to date and respond quickly to new vulnerabilities. Buyers can use an SBOM to perform vulnerability or license analysis, both of which can be used to evaluate risk in a product. ^[8] Since modern software applications no longer rely on a monolithic stack of discrete components, developers now build applications out of many components that can come from many sources. Any one of the components that make up an application could represent a risk if there is an unpatched vulnerability. As such, it is critical for developers, organizations they work for, and end users who consume applications to be aware of all the components that make up an application using standards such as **System Package Data Exchange (SPDX)**. ^[4]
- **Centralized logging:** A survey of impacted authorities and companies revealed that many organizations did not have the most basic logging processes. Should these organizations fall victim to a large-scale cyberattack, it would prove very difficult to investigate the incident and provide post-breach clean-up, as was the case with the SolarWinds hack. ^[2] This topic, including log aggregation, storage and anomaly detection, will be discussed further in [Chapter 8](#), *Continuous Monitoring*.

Case study: MOVEit vulnerability

Let us look at another case study for better understanding:

- **Date:** May 2023
- **Countries:** Multiple
- **Summary:** Many high-profile companies worldwide (government, finance, media, aviation, and healthcare organizations) had exorbitant confidential information stolen due to software weaknesses. Internet-facing MOVEit Transfer web applications were infected with a web shell named LEMURLOOT, which was then used to steal data from underlying MOVEit Transfer databases.^[12]
- **Impact:** Many MOVEit clients were affected. One of the providers of payroll services, Zellis, was affected, and the attackers claimed they had confidential data for eight UK organizations, including the BBC. The data included home addresses, National Insurance numbers, and bank details.^[12]
- **Root cause(s):** A SQL injection vulnerability in MOVEit Transfer and MOVEit Cloud was the root cause of the attack. The vulnerability allowed attackers to access MOVEit Transfer's database from its web application without authenticating. The attacker could then execute SQL statements that alter or delete entries in the database and infer information about the structure and contents of the database.^[12]

Detailed timeline

MOVEit is managed file transfer software produced by *Ipswitch Inc.* (now part of an American company, *Progress Software*) that encrypts files and uses file transfer protocols such as FTP or SFTP to transfer data. It also provides automation services, analytics, and failover options. For guest users, MOVEit provides one-time use scenarios. Guest users can send, view, download, replay, and receive packages. The attackers used guest access to elevate permissions due to the SQL injection vulnerability.^[12]

On May 31, *Progress Software* posted a notification alerting customers of a critical vulnerability (**Common Vulnerabilities and Exposures (CVE)-2023-34362**) in their MOVEit Transfer product. CVE-2023-34362 is

an SQLi vulnerability that enables threat actors to elevate privileges, view and download data from the database server, and potentially allow the theft of Azure system settings and the associated key and containers. ^[13]

A bug in MOVEit's code meant it incorrectly extracted headers that ended with X-siLock-Transaction, allowing requests to be sent to the guest access flow. It would then allow attackers to generate a CSRF token to create a session. Using this token can generate SQL injection of a list of SQL statements. One can then create a **JSON Web Token (JWT)** to obtain sysadmin **application programming interface (API)** access tokens. With sysadmin access, one can manipulate the database to retrieve/upload/update data and delete files. Further attackers could use the payload from the comments field in the database (inserted using SQL injection), upload a file, and achieve **remote code execution (RCE)** by abusing deserialization. For more details, refer to the 14th link in the *References* section.

Attackers used this vulnerability to drop a web shell named LEMURLOOT. LEMURLOOT was a C# web shell targeting MOVEit Transfer, using a hard-coded password to execute commands like downloading files, extracting Azure settings, and managing users with hard-coded passwords. It retrieved storage details and metadata via SQL queries, compressing and returning data to the client. ^[15]

Ransomware group *Clop* believed to be Russian-based, exploited this vulnerability. The MOVEit team worked with industry experts to investigate. In June, Progress tested and deployed a patch to MOVEit Cloud and made it available for MOVEit Transfer. On July 5, 2023, Progress Software introduced a new service pack system for all MOVEit products for more regular updates. ^[16]

Learnings

Some of the key learnings from this case study are as follows:

- **Adopt secure coding practices:** Since the vulnerabilities exploited SQL injection flaws, organizations must enforce secure coding practices, including input validation, escaping, and parameterized queries, to reduce susceptibility to similar attacks. SQL injection is a critical web application security vulnerability that the **Open Web Application Security Project (OWASP)** has consistently ranked as one

of the top threats. These vulnerabilities should be part of regular design and development practices.

- **Secure build and deployment processes:** Integrate automated security tools in **continuous integration/continuous deployment (CI/CD)** pipelines to scan for vulnerabilities in dependencies and configurations. Use **static application security testing/dynamic application security testing (SAST/DAST)** tools to identify issues early in the build process.
- **Proactive patch management:** Organizations should prioritize applying security patches as soon as they become available. The MOVEit vulnerabilities demonstrate how quickly attackers can exploit known vulnerabilities.
- **Continuous monitoring:** Implement robust logging and monitoring systems to detect unusual activities. In the MOVEit case, some exploitation attempts dated back to July 2021, highlighting the importance of long-term log retention and analysis.
- **Incident response preparedness:** Develop and regularly update incident response plans. The rapid exploitation of MOVEit vulnerabilities underscores the need for quick and effective response capabilities.
- **Third-party risk management:** Regularly assess and monitor the security posture of third-party software and services. The MOVEit vulnerabilities affected numerous organizations, emphasizing the importance of understanding supply chain risks.

Other notable examples

Let us look at some other notable examples in the following subsections.

XZ Utils backdoor

In February 2024, a Microsoft engineer named *Andres Freund* spotted a vulnerability in *XZ Utils*, a general-purpose data compression library and command line shipped with Linux operating systems. The attacker introduced the backdoor, allowing the attacker to execute commands remotely. Although XZ Utils dates to 2009, the vulnerability was introduced in 2023 by a second maintainer. Andres spotted this when investigating performance issues due to high SSH connections. The CVE was assigned a

CVSS score of 10, the highest possible score. The incident highlighted the importance of vigilance when using open-source software.

British Airways supply chain attack

In September 2018, *British Airways* announced a breach that resulted in the theft of customer data, including names, email addresses, and credit card details. The breach resulted from a sophisticated supply chain attack (like SolarWinds); the attacker (linked to *Magecart group*—a web skimming syndicate) introduced a script on the BA website that would allow them to capture customer details, including card details, and send them to an external location.

Core concepts

As you can see from these case studies, a secure software development lifecycle that prioritizes security from requirements through deployment is integral to ensuring security. *Figure 3.1* shows various areas of a software development lifecycle that need to be secured. This includes the following key lifecycle areas, which we will discuss in detail:

1. **Secure requirements and design:** These are techniques for ensuring that security is considered in your requirements and design.
2. **Secure code:** Looks at how every software engineer (dev or infra) can adapt to ensure the security of their code.
3. **Secure build:** It focuses on the security of the build phase.
4. **Secure scan and test:** It is a subset of build, but mainly verifies the software quality and security.
5. **Secure deploy:** This involves releasing software securely.

1. Secure code, build and deploy

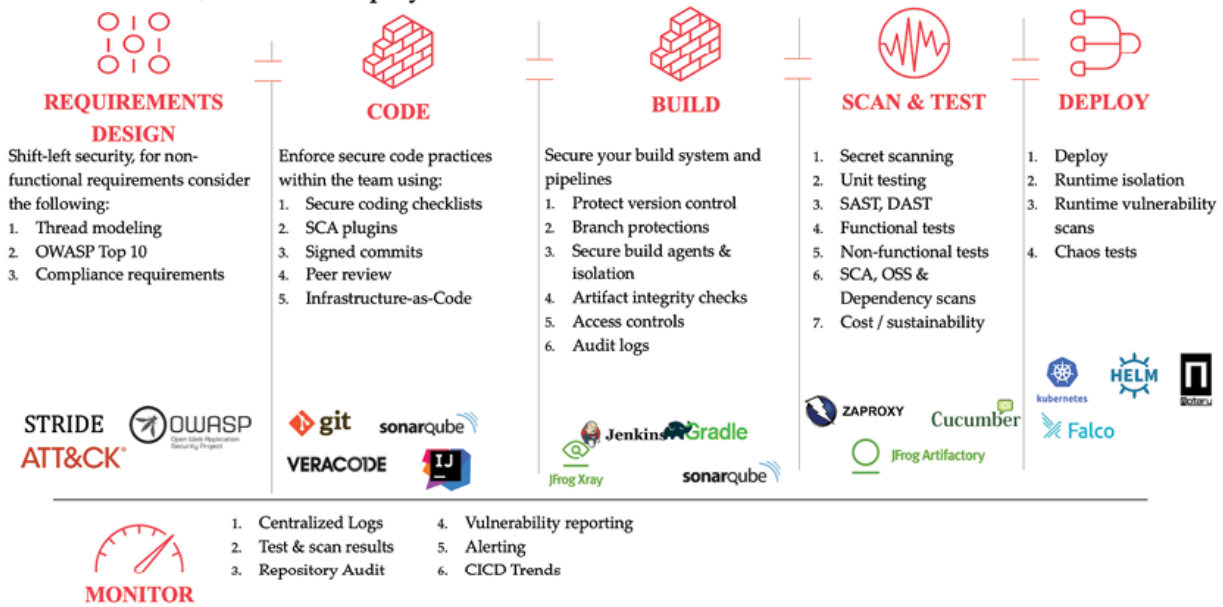


Figure 3.1: Core concepts of secure your code

Secure requirements and design

Requirements and design are the first stage of software development, where the objectives, functionalities, and constraints are identified and documented. These requirements are a foundation for creating design specifications that outline how the software will achieve its goals. As a result, this is an important place to identify security inputs to building software. Let us discuss the key techniques that could be used to build security into the product.

Threat modeling

Threat modeling is an important activity that is the foundation of secure development. Creating a security blueprint for your application before development begins is best. Just as a building architect considers structural vulnerabilities in a building's design, developers must anticipate potential security weaknesses in their applications. There are many different approaches to threat modeling, such as STRIDE, OWASP top 10, MITRE ATT&CK, and covered here:

- **Spoofing, tampering, repudiation, information disclosure, denial of**

service, and elevation of privilege (STRIDE): The STRIDE model (shown in [Figure 3.2](#)) is an effective methodology that has evolved to help identify boundaries, events, and entities by applying them to a data flow diagram. It focuses on the six threats that can go wrong in an application. ^[9]

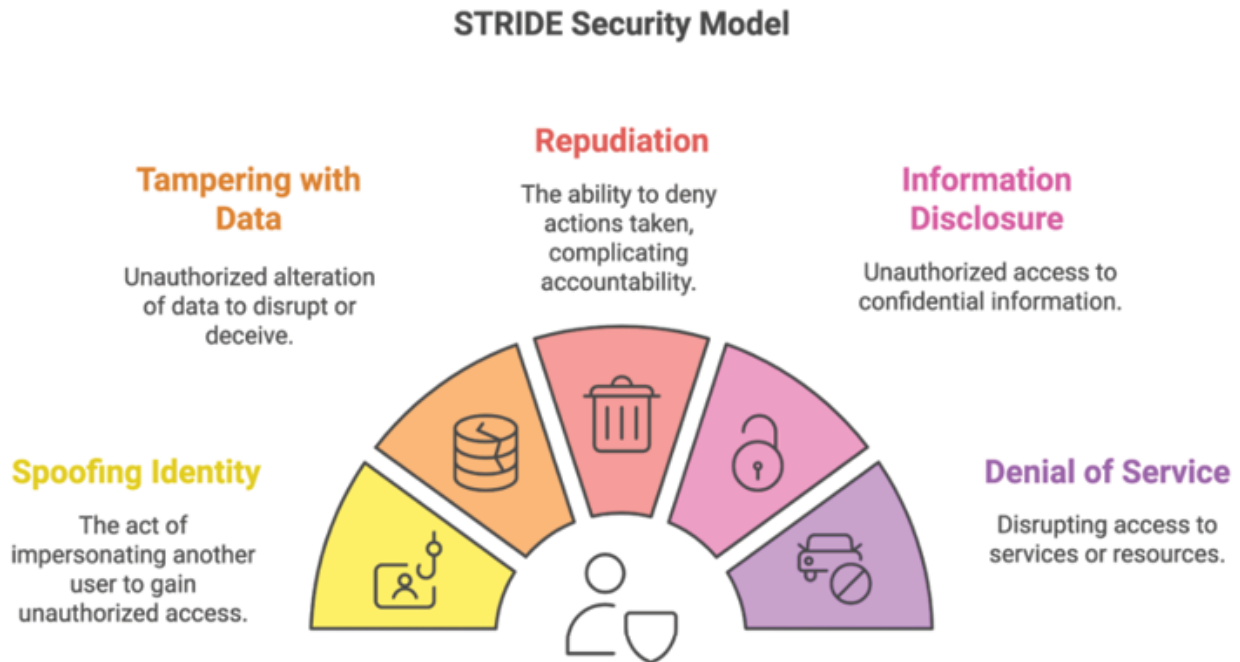


Figure 3.2: The STRIDE model ^[9]

[Table 3.1](#) shows the key threats, what they are, and some examples of how attackers can exploit them:

Threat	Details	Exploits
Spoofing	(Authentication) Impersonating users or systems to steal data or gain access. E.g. Phishing attacks, session hijacking, social engineering and DNS spoofing	An attacker pretending to be a bank's website using a similar domain or email address to trick users into sharing confidential data.
Tampering	(Integrity) Modifying data or code to redirect a user or access and steal data E.g. MITM, SQL injection, file integrity attacks	Manipulating API responses to change transaction amounts, sending users to an external website.
Repudiation	(Non-repudiation) Adopting measures that help in denying malicious actions E.g. Log forging	A user claiming they did not authorize a transaction, hiding unauthorized access by tampering with audit logs.

Information disclosure	(Confidentiality) Exposing data E.g. Insider data breaches	Accidentally exposing customer records through misconfigured APIs.
Denial-of-service (DoS)	(Availability) Overwhelming systems or parts of systems with requests to render them unusable E.g. DDoS, application DoS	Flooding a login system or website with requests to cause downtime.
Elevation of privilege	(Authorization) Gaining unauthorized access and elevating privileges causing harm E.g., Privilege escalation, social engineering, Kernel exploits	A regular user accessing admin functions.

Table 3.1: STRIDE model

- **Real-world relevance:** The STRIDE model helps effectively find the threats inherent in the system's design and implements contingencies to cover those gaps. The STRIDE threat model ensures that software products maintain the CIA triad (mentioned in the previous chapter).

The best way to apply the model (depicted in [Figure 3.3](#)) is by regularly performing threat modeling exercises, given as follows:

1. **Scope:** Define and agree on the scope of the exercise, document the architecture and flows, and identify key assets and actors.
2. **Identify threats:** Use STRIDE or one of the other models. Categorize each threat into the appropriate bucket as advised by the model.
3. **Assess risks:** Based on their likelihood and business impact, define the threat actors for the risks, and identify the source of the threat (such as coding, configuration, build, and deployment).
4. **Mitigate:** The threats by implementing security controls (such as MFA, training, encryption, firewalls, and data reconciliations), prevention measures, detection and response based on alerts, or redundancies such as backups and scaling (for **distributed denial-of-service (DDoS)**).
5. **Review and update:** Document the decisions, capture learnings, and test and validate the mitigations. Update the model with every new feature release, security incident, or architectural or infrastructure change.

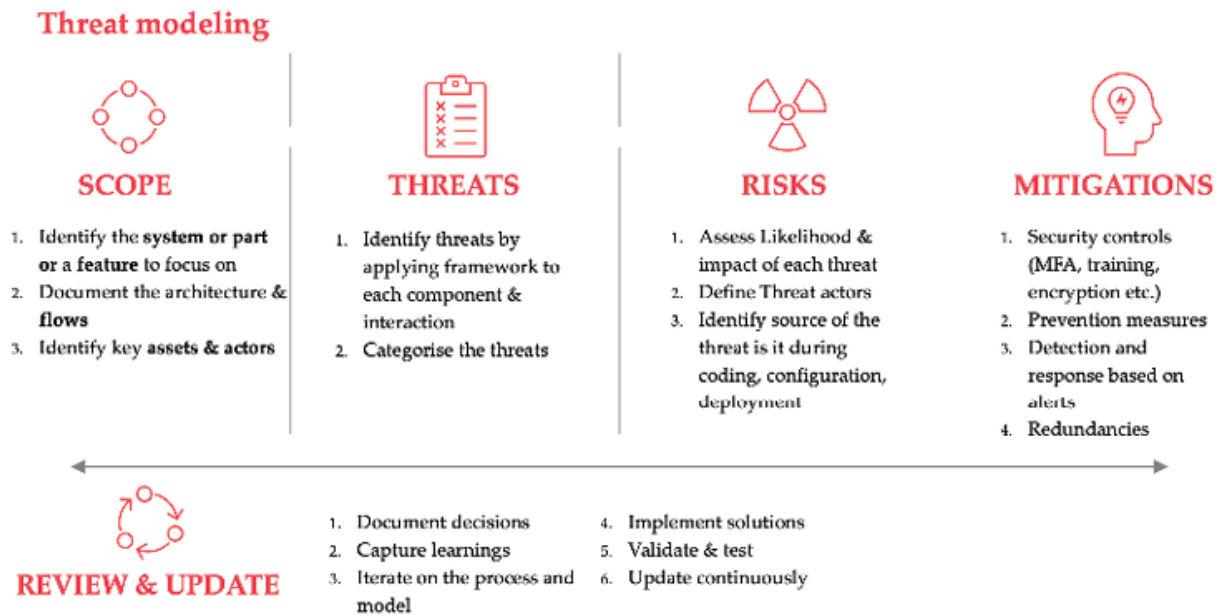


Figure 3.3: Applying the STRIDE model

Refer to the OWASP threat modeling process example and the 17th link in the *References* section. Start by drawing a simple architecture flow of the application's scope, focusing only on its scope areas. Then, identify various threats and mitigations, like the example listed here. [Table 3.2](#) shows a sample output from applying the model:

Type	Description	Security control
Spoofing	The attacker steals user credentials and impersonates them	MFA
Tampering	The attacker modifies an audit log entry	Immutable audit logs
Repudiation	The user claims they did not perform a transaction or payment	Detailed logging, capture location details, send notifications post transaction
Information disclosure	Attacker steals sensitive data	Encryption in transit and at rest
Denial of service	Attacker floods the login with invalid requests	Rate limiting, use CAPTCHAs
Elevation of privilege	User exploits SQL injection to gain admin rights	RBAC, server-side checks, data validation and sanitization, parameterized queries

Table 3.2: Applying the STRIDE model (example)

- **MITRE ATT&CK:** MITRE **adversarial tactics, techniques, and common knowledge (ATT&CK)** is a knowledge base of adversarial

techniques based on real-world observations. It helps organizations understand the behavior and actions of cyber adversaries by mapping out attack stages, from initial access to data exfiltration. The framework is beneficial for improving threat detection, incident response, and threat hunting by offering a structured way to identify and track adversary behaviors across various stages of an attack. Organizations can use MITRE ATT&CK to assess their security posture by identifying gaps in their defense mechanisms, developing detection rules, and improving their incident response playbooks. Security teams can leverage ATT&CK to simulate adversary techniques. It also serves as a common language for communication within and between security teams, enhancing collaboration and strategic defense planning.

- **Real-world relevance:** Let us say you were to apply MITRE ATT&ACK to detect and respond to a ransomware attack like the Colonial Pipeline example we covered in [Chapter 1, Understanding Zero Trust](#). Here is how you can do it:
 - The team references the MITRE ATT&ACK framework to map out potential tactics and techniques. Looking through the initial access, the team should identify relevant techniques, such as phishing, exploiting public-facing applications, and exploitation or privilege escalation.
 - The team then implements additional monitoring and detection techniques, such as configuring alerts, logging, etc.
 - Investigation and incident response continue per the tactics and techniques the MITRE ATT&CK framework provides for real-world attacks.
- **CVSS: The Common Vulnerability Scoring System (CVSS)** is a standardized framework used to assess and quantify the severity of security vulnerabilities, providing a numerical score that helps organizations prioritize risks during threat modeling. CVSS evaluates vulnerabilities based on several metrics, including exploitability, impact on confidentiality, integrity, availability, and user interaction required for an attack. This scoring system allows security teams to objectively assess the risks associated with various vulnerabilities and prioritize

them according to their potential impact on the organization's assets. In threat modeling, CVSS helps identify high-risk vulnerabilities that require immediate attention and guides the allocation of resources toward mitigation efforts. By integrating CVSS into the threat modeling process, organizations can enhance their ability to proactively identify and address critical threats, improving overall security posture and risk management. This will also be discussed later in the vulnerability management topic, but for now, we understand this is part of the requirements and design process.

- **Real-world relevance:** Whenever a vulnerability is identified within the system, one needs to look at the CVE score associated with it, typically a number between 0 and 10. A score of 9 or higher would be considered severe. However, it is also applying the context of the application or asset to the score. If the application or asset is not critical or does not store any confidential data and is completely isolated, it could be prioritized lower than other systems that might be.

OWASP top 10

The **Open Worldwide Application Security Project (OWASP)** is a nonprofit foundation that works to improve software security. OWASP top is the most popular standard awareness document for developers and web application security. It lists the most critical security risks web applications can face. Since many applications currently involve a web-based interface or API, these are essential in development. OWASP has also updated the framework with ongoing technology and threat landscape changes. The framework will be updated in 2025, but the previous version covered the following threats and associated mitigations. [Table 3.3](#) shows the current vulnerabilities and mitigations:

Vulnerability	Description, mitigations, case studies

Vulnerability	Description, mitigations, case studies
A01 broken access control	Occurs when users can access resources or perform actions outside their intended permissions. Attackers can exploit this to view or modify data they should not have access to. To prevent, one should: • Implement proper role-based access control (RBAC). • Enforce server-side access control, not just client-side checks. • Use least privilege principles for users and applications. Real-life example(s): An access control flaw in <i>Facebook</i> allowed attackers to exploit the <i>View As</i> feature to gain access to other accounts; 50-90 million accounts were accessed by attackers.
A02 cryptographic failures	Improper implementation of cryptographic algorithms leads to sensitive data exposure or weak encryption schemes. Examples include using outdated algorithms (e.g., MD5, SHA-1) or poor key management. • Use modern cryptographic algorithms (AES-256, RSA-2048) • Ensure proper key management, stored separately from data • Implement encryption of sensitive data at rest and transit We will cover this in more detail under Chapter 6, Secure Your Data. Real-life example(s): Chapter 6 case study: Marriott, sensitive data was not encrypted.
A03 injection	Flaws, such as SQL injection, occur when untrusted data is sent to an interpreter as part of a command or query, allowing attackers to manipulate queries or commands to the backend systems. • Use prepared statements with parameterized queries for databases. • Validate and sanitize all inputs to prevent malicious data. • Utilize object-relational mapping (ORM) libraries Real-life example(s): Chapter 3 case study: MOVEit, SQL injection vulnerability affecting organizations globally.
A04 insecure design	Flaws in the design phase of an application's development. It includes missing security controls or weak architecture that enables attacks (e.g., failure to use secure communication channels). Use secure design patterns and principles. Threat modelling during the design phase to identify risks.
A05 security misconfiguration	Applications, servers, or databases are not securely configured, leaving them vulnerable to attacks. This can include unnecessary services running or default credentials being used and exposing private services publicly. • Disable unnecessary services and ports. • Use automated tools to audit configurations and enforce secure settings. Real-life example(s): Chapter 7, Manage Vulnerabilities case study: Capital One. The breach was caused by the misconfiguration of the AWS S3 bucket.

Vulnerability	Description, mitigations, case studies
A06 vulnerable and outdated component	Using outdated libraries, frameworks, or components containing known vulnerabilities can expose applications to attackers, specifically third-party dependencies. • Continuously monitor and patch software dependencies. • Use automated dependency management tools (e.g., Snyk, Dependabot) to alert you to outdated components. • Regularly audit and remove unused libraries. Real-life example(s): <i>Chapter 7, Manage Vulnerabilities</i> case study: Equifax. The breach was caused by Apache Struts vulnerability.
A07 identification and authentication failures	Failures in authentication mechanisms, such as weak password policies, poor session management, or failure to implement MFA, allow attackers to bypass login systems. • Enforce strong password policies (e.g., length, complexity). • Use MFA to strengthen access control. • Implement secure session management, with proper session expiration and protection against session fixation.
A08 software and data integrity failures	Software, firmware, or data integrity is compromised using insecure update mechanisms or failing to validate the integrity of critical software updates. • Ensure software updates are signed and verified through secure mechanisms. • Use hash-based validation to ensure data integrity. • Use content security policies (CSP) to protect from tampered scripts and data. Real-life example(s): <i>Chapter 3</i> case study: SolarWinds was a widespread supply chain attack.
A09 security logging and monitoring failures	Insufficient logging and monitoring prevent organizations from detecting attacks or responding to security incidents. Attackers can exploit this to carry out malicious activities without detection. • Log all critical events, including authentication, access control changes, and data modifications. • Use centralized log management and analysis systems (ELK etc). • Set up alerts for abnormal activities to enable prompt response. Real-life example(s): <i>Chapter 3</i> case study: SolarWinds. The breach also lacked centralized logging and monitoring solutions for many affected organizations.

Vulnerability	Description, mitigations, case studies
A10 server-side request forgery (SRF)	An attacker can make requests from the server to internal or external resources. This is particularly dangerous when the server is trusted to request internal services or systems. • Access controls on all critical components. • Network-level firewalls and request filtering to block internal IP addresses or services from external requests. • Whitelisting for allowed URLs or resources that can be accessed. • Validate and sanitize all URLs before allowing them to be used for server requests. Real-life example(s): Chapter 7, case study Capital One. The attacker also exploited the SSRF vulnerability to gain AWS credentials.

Table 3.3: OWASP top 10

You can apply these techniques in the real world, as follows:

- Familiarize yourself with the top 10 vulnerabilities.
- Integrate all the mitigations to these vulnerabilities into your development lifecycle. This means evaluating every requirement and every feature.
- Perform threat modeling as part of the design to understand further requirements.
- Implement appropriate security controls: data encryption, secure authentication, and session management, data validations, sufficient logging, etc.
- Test for these vulnerabilities and penetration testing.
- Review, evaluate, and respond to security incidents.

Let us look at some examples of how some of these controls can be applied:

- **A01 broken access control:** Sometimes, developers miss out on enforcing server-side access control and assume client-side checks are sufficient. However, it is equally essential to ensure that every data or request from the server side or your API is validated for access. You must ensure the user trying to read/write data can do the same before allowing the operations. If this is not done, an attacker can bypass the front end and invoke the APIs/server-side directly to escalate their privilege and make changes they are not authorized to.

- **A03 injection (SQL injection):** The following Python code takes a parameter and dynamically searches. The first example (labeled as vulnerable) concatenates user input directly into the query, making it vulnerable to SQL injection, as the attacker can append additional clauses to the statement. The second example (labeled as corrected) uses a parameterized query:

```
from django.db import connection
cursor = connection.cursor()
def get_user_data_bad(username): # vulnerable
    query = "SELECT * FROM users WHERE username = '" + username + "'"
    cursor.execute(query)
def get_user_data_good(username): # corrected
    cursor.execute("SELECT * FROM users WHERE username = ?", (username))
```

Most programming languages have prepared statement support that should be used instead of dynamically constructing the query using strings; refer to the 18th link for Java in the *References* section.

Compliance requirements

The last source of requirements and design inputs is compliance requirements from within the organization and outside. Many requirements will be industry and jurisdiction-specific and mandated by regulatory bodies, standards organizations, or internal governance policies. For example, financial institutions must comply with regulations like the **Payment Card Industry Data Security Standard (PCI DSS)** for payment security or GDPR for data protection. In contrast, payment institutions must address standards such as PSD2 for secure digital transactions PCI DSS. Similarly, telecom providers often align with frameworks like ISO 27001 for information security management.

These compliance mandates influence systems' design, from data encryption and user authentication to audit logging and access controls. Adhering to these requirements not only ensures legal and ethical operation but also enhances trust and security for stakeholders. Integrating these considerations early in the design phase helps streamline development and reduces the risk

of costly rework or regulatory penalties later. *Chapter 12, Policies, Frameworks and Compliance*, focuses on policy frameworks, where we will dive deeper into this topic.

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Business impact to prioritize:** Depending on the team's expertise, you may come up with many threats; weighing them based on their business impact and likelihood is important. If not, you might create and build many features that may not be of real value.
- **Collaboration is key:** The main aim of the process is collaboration and joint learning of the team. Include developers, architects, security professionals and business stakeholders so they can all contribute.
- **Documentation:** The key learnings, decisions, threats, mitigations and assumptions should be documented for audit purposes. Document sufficient evidence but do not overshoot; it should be what you can maintain long-term.
- **Keep the process simple:** Use a simple, clear and concise approach to reviewing flows, scoring and prioritization. Do not add too much complexity, process, or extensive documentation that cannot be maintained over time.
- **Clear scope boundary and start small:** Examining a system can sometimes be overwhelming, as it might raise many threats and questions. It might be helpful to break down the system and focus on certain features.
- **Continuous improvement mindset:** Threat modelling is most effective when performed regularly as the application or system evolves with changing technology and threat landscape.

There is no one-size-fits-all solution; try different tools and frameworks for specific needs.

Secure coding

In today's world, where everything runs on software, secure coding is a

critical practice in Zero Trust engineering, which plays a vital role in protecting systems, data, and users from a wide array of cyber threats. By implementing secure coding principles, developers can significantly reduce vulnerabilities in their software, making it more resilient against attacks covered in the previous section. This proactive approach enhances applications' security posture and helps organizations comply with regulatory requirements and industry standards. Moreover, secure coding practices can improve software quality, reduce maintenance costs, and enhance user trust.

In an era when cyberattacks are becoming increasingly sophisticated and frequent, secure coding forms the foundation of robust cybersecurity strategies. It is essential for safeguarding sensitive information and maintaining the integrity of digital systems, and it is also the easiest to implement.

Secure coding checklist

The best starting point for secure coding is secure coding checklists. OWASP secure coding practices^[10]. In addition, OWASP also includes a cheat sheet series that has helpful guidance ^[11]. You will also find similar checklists for most programming languages. *Table 3.4* is a summary of the checklist; this is not meant to be exhaustive, so please familiarize yourself with the OWASP secure coding practice and other checklists for the specific programming languages you use. We will discuss some of these checklists in the later chapters.

Technique	Details
Input validation	Never trust user input: validate the length, format, and content of all incoming data using whitelisting and sanitize inputs.
Output encoding	Encode outputs to protect against XSS and sanitize outputs of operating system commands and untrusted data when querying SQL, Lightweight Directory Access Protocol (LDAP) .
Authentication	Implement strong authentication and password policies with MFA support. Secure password storage using industry-standard hashing.
Authorization	Enforce RBAC with the principle of least privilege for all resources
Session management	Use secure session handling with appropriate timeouts and secure cookie settings.

Technique	Details
Data protection	Encrypt sensitive data both at rest and in transit using strong, standardized algorithms. Protect all cached or temporary copies of sensitive data stored on the server from unauthorized access and purge those temporary working files as soon as they are no longer required. Do not store passwords, connection strings or other sensitive information in clear text or any non-cryptographically secure manner on the client side. Remove comments in user-accessible production code that may reveal the backend system or other sensitive information. Do not include sensitive information in HTTP GET request parameters. Disable auto-complete, client-side caching on forms and pages expected to contain sensitive information, including authentication. The application should support the removal of sensitive data when that data is no longer required.
Error handling	Implement custom error handling that does not leak sensitive information to users (like name, DOB, etc.). Do not disclose sensitive information in error responses, including system details, session identifiers or account information.
Logging and monitoring	Log security events securely with appropriate details for incident investigation without capturing sensitive data. Log all access control failures, validation errors, authentication attempts. All logging controls should be implemented on a trusted system.
API security	Secure all API endpoints with proper authentication, rate limiting, and input validation.
Database security	Use strongly typed parameterized prepared statement queries to prevent SQL injection attacks. Ensure that variables are strongly typed. Connection strings should not be hard coded but stored in config in a separate trusted system and encrypted. Close the connection as soon as possible.
File operation	Require authentication before upload. Do not save files in the same web context as the application. Never send an absolute file path to the client. Do not trust uploaded files: Validate file uploads, restrict file types, and scan for malware.
Memory management	Implement proper memory handling to prevent buffer overflows and memory leaks Check that the buffer is as large as specified. Specifically, close resources; do not rely on garbage collection.

Technique	Details
Secure dependencies	Use only verified dependencies from trusted sources with no known vulnerabilities.
Secure communications	Force Hypertext Transfer Protocol Secure (HTTPS) and implement the proper certificate validation. Transport layer security (TLS) certificates should be valid and have the correct domain name, not be expired, and be installed with intermediate certificates when required. Utilize TLS connections for all content requiring authenticated access and for all other sensitive information. Failed TLS connections should not fall back to an insecure connection.

Table 3.4: Secure coding checklist

Although familiarizing yourself with these secure coding guidelines and principles is important, it is also good to have automated prompts where possible. This is where many of the tools that offer **integrated development environment (IDE)** integrations can help. See the *Tools and Plugins* section later for a list of these tools.

Secure code review

What is better than one head (or human) working on a problem? It is two heads. This is one of the main ideas behind pair programming that **extreme programming (XP)** leverages heavily. However, this idea can be extended to any Agile software development lifecycle. We do this through code reviews, specifically secure code reviews. It is difficult for a single person to build the software and look for injection flaws, insecure configurations, and improper access controls, reducing the likelihood of security breaches in their code. Equally, the principle of **separation of duties (SoD)** is vital to ensure impartiality; developers who wrote the code should not be the sole reviewers, minimizing the risk of oversight or bias.

Secure peer code reviews (encouraging members from outside the team wherever possible) are critical in identifying vulnerabilities and ensuring software integrity before deployment. They help detect issues, such as secure code reviews, which can be conducted manually or with automated tools like SAST solutions, which scan code for known vulnerabilities. A well-structured code review involves understanding the application's threat model, focusing on high-risk areas, and validating adherence to security best

practices such as input validation, proper authentication, and secure cryptography. Combining manual expertise with automated tools ensures thorough coverage, fostering a robust security posture and delivering software that users can trust.

Secure code reviews should use the same checklists we covered above to validate that the code does what it needs in a safe and secure manner. [Table 3.5](#) provides a sample checklist for code reviews:

Category	Checklist item
Code quality	Is the code clean, readable, and maintainable?
	Are variable and function names meaningful and consistent?
	Is code free of unused variables, imports, and commented-out code?
	Are coding standards and style guides followed (e.g., indentation, line length)?
Functionality	Does the code meet the requirements and specifications?
	Are all edge cases and error conditions handled appropriately?
	Are there any logical errors or unexpected behaviors?
Security	Is input properly validated and sanitized?
	Are sensitive data and secrets (e.g., passwords, API keys) securely handled?
	Are secure methods used for cryptographic operations?
	Are potential vulnerabilities (OWASP top 10), such as SQL injection or XSS, addressed?
	Are authentication and authorization mechanisms implemented correctly?
Performance	Is the code optimized for performance without premature optimization?
	Are resource-intensive operations minimized or properly managed?
	Are there any potential memory leaks or inefficient loops?
Error handling and logging	Are errors and exceptions handled adequately with clear messages?
	Are errors and exceptions logged appropriately to aid in troubleshooting?
	Is logging implemented appropriately to avoid sensitive data exposure?
Dependencies	Are dependencies up-to-date and free of known vulnerabilities?
	Is there unnecessary or redundant use of external libraries?
Test coverage	Are unit tests written for all new code and changes?
	Are test cases comprehensive, covering both positive and negative scenarios?

Category	Checklist item
	Do all tests pass successfully?
Documentation	Are comments clear and helpful where needed?
	Are public methods and APIs documented with clear explanations?
	Is there a README or other relevant documentation for the code?
Compliance and standards	Does the code comply with relevant legal, regulatory, or organizational standards?
	Are secure defaults implemented for configuration settings?
Collaboration and workflow	Are commit messages clear and meaningful? Is the work linked to appropriate work items in tracking tools such as JIRA?
	Is the pull request description detailed, explaining the purpose of the changes?
	Are changes scoped appropriately, avoiding unrelated modifications?

Table 3.5: Secure code review checklist

Infrastructure as code and checklists

Infrastructure as code (IaC) allows for infrastructure provisioning and management using machine-readable configuration files instead of manual processes. Nowadays, all cloud providers (AWS, Azure, Google) support infrastructure as code using tools such as Terraform, Ansible, a native configuration management solution, and programming languages. Regardless of your chosen solution, IaC has multiple benefits over manual provisioning as it enables consistency, scalability, better audit and control, and automation across environments and parity. However, insecure IaC practices can lead to vulnerabilities like misconfigured resources, excessive permissions, or unprotected data exposure. Securing IaC involves integrating security practices throughout the development lifecycle, leveraging tools for static analysis, enforcing compliance with standards, and maintaining least privilege access. [Table 3.6](#) is a checklist for the IaC code. We will cover some of the tools later in the chapter.

Category	Checklist item
General practices	Are all resources (e.g., servers, databases) explicitly defined in the code?
	Are configurations modular, reusable, and appropriately documented?
Access control	Are IAM roles, policies, and permissions least privileged by design?
	Are secrets (credentials, keys, or sensitive data) securely stored (e.g., HashiCorp

	Vault, AWS Secrets Manager, Key Vault)?
Network security	Are network components like VPCs/VNET, subnets, and firewalls configured correctly? Are security groups and firewall rules restrictive (deny-all by default)? Are public access to resources (e.g., S3 buckets) and open ports minimized?
Compliance	Are configurations aligned with industry standards (e.g., CIS Benchmarks, NIST)? Are audit logging and monitoring enabled for all critical resources?
Resource configuration	Are resource names consistent and follow a naming convention? Have unused or orphaned resources been identified and removed?
IaC security	Are IaC templates (e.g., Terraform, CloudFormation) free from misconfigurations? Are static analysis tools (e.g., Checkov, Trivy) used to scan IaC for vulnerabilities?
Version control	Is the infrastructure code stored in version controlled repositories (e.g., Git)? Are code reviews enforced for all infrastructure changes?
Automation and pipelines	Are CI/CD pipelines secure, avoiding plain-text secrets or insecure dependencies? Are automated tests included to validate infrastructure code changes?
Logging and monitoring	Are logs configured to capture security events without sensitive data exposure? Are monitoring tools (e.g., ELK, Prometheus, AWS CloudWatch, Azure Monitor) used to detect anomalies?

Table 3.6: *Secure IaC checklist*

Secure build

The SolarWinds case study highlighted how attackers can infiltrate a build system, leveraging it to create a supply chain attack by compromising product updates distributed to organizations. A secure build is critical to ensure software is developed and deployed with robust security practices. Organizations can identify and mitigate risks early by integrating automated security checks into the build pipeline, such as static code analysis, vulnerability scans, dependency management, and artefact generation, reducing the likelihood of introducing vulnerabilities into production and ensuring compliance with security standards throughout the development lifecycle.

A secure build pipeline ensures controlled and safe building, testing, and

deployment of applications. Automating security checks (to be covered under the *Security Scan and Test* section) ensures that you identify vulnerabilities early while deploying artefacts through verified processes, ensuring that only trusted code reaches production. A secure build solution comprises the following key controls, shown in *Table 3.7*:

#	Step	Description
1	Version control system	Use a version controlled environment (Git is the most popular) for code and configuration. However, apply the same security controls to your version control: MFA, access controls, audit logs, encryption, and backup and recovery.
2	Code commit signing	Involves digitally signing (<i>Chapter 6, Secure Your Data</i>) software commits to confirm that the code is legitimate and has not been altered/tampered with since signing, preventing potential security issues like unauthorized code changes or supply chain attacks.
3	Branch protection	Protect critical main or master branches from unauthorized, risky, and untested changes. Pull request reviews should be used to enforce successful builds, security scans, tests, and reviews by authorized team members. Depending on the sensitivity of the code repository, you might introduce one or more approvals across one or more teams (such as security or operations).
4	Configuration validation	Verify secure configurations for the environment, servers, and network, use secure communication protocols (e.g., HTTPS, SSH) to transmit data in and out of build systems, and lock down network access to necessary domains and IPs. (<i>Chapter 5, Secure Your Network</i>)
5	Build isolation	Build artefacts in isolated environments (separate from your application environments) to prevent contamination. Ensure the build environment does not share resources (if possible, for each team or dedicated system), runs consistently, and is automatically cleaned up. Try to maintain versioned and immutable build systems; this eliminates the risk of persistent access by attackers.
6	Secure build agents	Use hardened, isolated environments for build agents to prevent malicious actors from compromising the build process. Ensure that all build and deployment tools, including CI/CD software, libraries, and infrastructure, are regularly updated with the latest security patches. Where possible, leverage containerization.
7	Artifact integrity	It is like code signing, but involves signing and verifying build artefacts generated by the build process to ensure integrity. Verifying software authenticity and integrity is crucial for preventing supply chain attacks and assuring end-users that the software is secure and safe.

#	Step	Description
8	Secure package management	Secure package management mitigates the risks associated with third-party dependencies. Use trusted package repositories and configure dependency management tools to enforce version control, avoiding outdated or vulnerable libraries. Implement tools like software composition analysis (SCA) and open-source software (OSS) scans (check Scans and tests, tools sections).
9	Access controls	Enforce strong authentication and MFA; limit code repo access, build and deployment permissions to essential personnel only (least privilege). (<i>Chapter 4, Secure Your Identities</i>)
10	Secrets management	Securely store sensitive credentials (API keys, access keys, logins) using tools like vault or environment variables.
11	Logging and monitoring	Enable detailed audit logging of all pipeline activities and continuously monitor (via alerts) for anomalies or unauthorized actions in the CI/CD process.

Table 3.7: Secure build system controls

All in all, secure build pipelines ensure that applications are built, tested, and deployed in a controlled and safe manner. Ensure that build servers and agents run on isolated and patched systems with minimal permissions. Use secure communication protocols (e.g., HTTPS, SSH) to transmit data and protect credentials, keys, and other sensitive data through dedicated secret management solutions. Regularly audit and monitor build logs for unauthorized changes or anomalies. Enforce MFA and limit access to build tools, ensuring that only trusted personnel can modify configurations or trigger builds.

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Build agent isolation:** A common anti-pattern is to use a single build agent across all application (system) teams. The risk with this approach is that compromising one agent compromises all systems. Automate entire build agent provisioning and hardening to separate agents, but balance cost by keeping this footprint as small as possible.
- **Automation:** Another common misstep is introducing manual steps, such as installing software on servers. We will discuss the importance of automation, but remember that every manual step has the potential to compromise security.

- **Secrets:** Embedding sensitive information (e.g., API keys, passwords) in build scripts or configurations can lead to unauthorized access if exposed; use secret managers.
- **Audit trails:** A commonly overlooked aspect is the lack of centralized logging or monitoring in the build process. As a result, it is difficult to detect, investigate, or respond to security incidents.
- **Access controls:** Sometimes, people keep access to the build system wide open. Poorly implemented CI/CD pipeline permissions or build tools increase the likelihood of unauthorized changes or malicious actions. The other extreme is to make it too granular, such that it limits people from doing their work, hampers productivity, or leads to everyone getting elevated permissions. Create well-defined roles and access, evaluate, and adjust them over time.

Secure scan and test

Scan and test is a subset of a secure build, although it is covered separately here due to the multiple steps and tools involved. Secure scans and testing encompass functional and non-functional tests and are critical to identifying and mitigating vulnerabilities throughout the software development lifecycle. Functional tests ensure core security mechanisms like authentication, authorization, and input validation work as intended, protecting systems from unauthorized access and data breaches and their impact on other functional requirements. Non-functional tests, such as performance and stress tests, assess how well security features perform under heavy loads, simulated attacks, or adverse conditions. Together, these tests ensure systems remain resilient, compliant with security standards, and prepared for real-world threats. Incorporating these tests into build and deployment (CI/CD) pipelines ensures continuous and consistent security validation, reducing risks. [Table 3.8](#) lists key scans and tests to include in the automated pipeline:

No.	Step	Description
1	Secrets scanning	To avoid accidental commits, use tools such as Gitleaks, Gitguard, and Trufflehog to detect secrets in the codebase.

No.	Step	Description
2	Unit testing	Enable automated unit testing and quality gates in your build pipelines. Depending on the language, this is typically available with the build tool (Gradle).
3	SCA and dependency scanning	Scan for vulnerabilities in dependencies, third-party libraries, and open-source components and enforce policies to restrict the use of unverified or deprecated packages using tools like OWASP Dependency-Check, SCA tools like JFrog Xray, Snyk or Black Duck.
4	Static code analysis	Run static code analysis and SAST tools to detect code quality, maintainability and security flaws. Checkmarx, Sonar, Snyk, Veracode.
5	Dynamic testing	Conduct DAST to identify runtime vulnerabilities. These tools perform crawling, simulate attacks, and monitor anomalies and errors. They can help detect attacks such as cross-site scripting (XSS) and authentication bypass. Tools used are Burp Suite, OWASP ZAP, Checkmarx One, HCL AppScan.
6	Container security	If using containers, scan container images for vulnerabilities before deployment for known vulnerabilities, misconfigurations, and insecure dependencies. Also use immutable images. (In Chapter 9, Container Security we will cover this topic in more detail)
7	Artifact integrity scans	Use cryptographic signing for package verification to ensure authenticity and integrity and routinely update dependencies to patch known vulnerabilities. Maintaining a dependency inventory helps organizations monitor, track, and respond to risks effectively.
8	Functional tests	Test system functionality with security controls such as authentication, RBAC, and encryption. Tools used are Postman, Selenium, Cucumber, Cypress.
9	Performance tests	Like functional tests, ensure the system remains resilient and effective under high load, stress, and adverse conditions. Tools used are JMeter, Gatling, and Artillery.
10	Penetration testing	Run periodic automated or manual penetration tests to assess the application's security posture. The focus is on simulating real-world attacks to identify security vulnerabilities in applications, networks, and systems. Tools used are Metasploit, BurpSuite, Cobalt Strike.

Table 3.8: Secure automated scan and test checks

If any of these scans or tests fail or do not meet the agreed thresholds, the pipeline should reject or release the artifacts.

Secure deploy

Deployment and release are the final steps of the software lifecycle. When deploying, it is critical to ensure applications and systems operate safely in production environments. Beyond the various scans and tests during builds, a secure deployment process involves using hardened configurations, validating dependencies, and implementing release management and rollback strategies and procedures. By embedding these practices, organizations reduce the risk of breaches and maintain their systems' integrity, availability, and confidentiality. [Table 3.9](#) shows the steps involved in the secure deployment process:

No.	Step	Description
1	Environment provisioning	Ensure servers, databases, and networks are configured securely and hardened.
2	Configuration management	Ensure application settings are secure and appropriate for production IaC and GitOps.
3	Artefact deployment	Deploy artefacts tracked in version control for traceability and immutable infrastructure to prevent post deployment changes.
4	Security benchmarks	Apply appropriate security benchmarks to the builds of artefacts and images. For instance, CIS Benchmarks for your software or operating system, ISO, or NIST standards applicable to your environment.
4	Deployment strategies	Adopt strategies such as blue green or canary to reduce risk by gradually rolling out changes and providing rollback options.
5	Testing and monitoring	Perform production validation testing to validate the new release's behavior, performance and stability (the <i>canary</i>) and continuously monitor logs and systems for anomalies or unexpected behavior. Tools: Spinnaker (Kayenta), Prometheus + Grafana, ELK, Datadog
6	Rollback procedures	Prepare and automate rollback processes for failed deployments. Options: previous version, database only, feature toggles, and traffic routing (primarily for blue-green). Document rollback procedures and communicate to end users.

Table 3.9: Secure deployment process

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Immutable infrastructure:** Do not perform updates on an existing application in place (say, by updating packages on a virtual machine).

Instead, immutable infrastructure principles for deployment should be adopted, where every change is handled by deploying new versions of infrastructure (either a new virtual machine or a new container based on a new image version) instead of modifying existing ones.

- **Environment segmentation:** Separate production, staging, and development environments to prevent unauthorized access or exposure during the pipeline process.
- **Parity of environments:** Keeping all environments (including networking, firewall setup, and type of infrastructure) similar (albeit with lower specs for lower environments) to ensure you do not face issues first-hand in production. This does need to be balanced against the cost of the infrastructure, so you might use a different tier of service (when on the cloud), but make sure you make the right compromises to avoid surprises in higher environments.
- **Fewer ephemeral environments:** The fewer the environments, the better it is to maintain parity, so if you can think of environments as ephemeral or short-lived and automate the entire setup (including test data population), you will have more control. This will also mean higher security, as you can tear down the whole environment and bring it back up within hours in case of security concerns.
- **GitOps or not:** We covered IaC earlier under secure code, which is necessary. However, GitOps builds upon this further by continuously reconciling infrastructure state against infrastructure code. This is typically more secure as it prevents configuration drift and reduces the need for production access and a full audit trail in code repositories. However, it may not be for all teams; evaluate if your teams are comfortable with performance overheads as environments grow.

Continuous integration and deployment monitoring

Monitoring the **continuous integration/continuous deployment (CI/CD)** process to ensure the stability, performance, and security of the entire pipeline, from code integration to deployment, is an essential way to prevent

the supply chain attacks we saw earlier. Effective monitoring provides real-time insights into pipeline health, build failures, deployment success rates, and potential anomalies. The best way to address these is as follows:

- **Audit logs:** Ensure comprehensive logging of all pipeline activities for auditing and debugging purposes.
- **Centralize logs and metrics:** Capture logs and metrics from your build and deployment systems into a centralized aggregation solution such as ELK, Grafana, or DataDog.
- **Define key metrics:** Identify critical metrics such as build success rates, build and deployment times, test pass rates, and system health indicators. [Table 3.10](#) shows some examples.
- **Enable real-time alerts:** Set alerts for failures, performance degradation, or unusual activities during builds, tests, or deployments.
- **Monitor dependencies:** Track third-party dependencies and integrations for vulnerabilities or compatibility issues.
- **Ongoing review and audit:** Analyze pipeline resource consumption to optimize costs and prevent system overload. Review and audit all key controls in your code, build and deployment processes.
- **Implement continuous feedback:** Link pipeline activity logs to a **security information and event management (SIEM)** such as Splunk, ELK for correlation with system-level threats, and use monitoring insights to improve pipeline processes and reduce failure rates over time

Sample metrics

[Table 3.10](#) shows some sample build metrics that you should measure:

Metric	Description
Build time deviation	Track the duration of builds; significant increases or decreases can indicate tampering or insertion of malicious code.
Artifact and image integrity	Validate artefacts and images (VMs/containers) using checksums or hashes to ensure they match expected values. Mismatches could indicate compromise.
Dependency changes	Monitor newly added or modified dependencies, especially those from unverified sources or with unusual update patterns.
Unusual build logs	Inspect build logs for unexpected commands, scripts, or outputs that deviate from standard build patterns.

Unauthorized commit activity	Track commit histories for suspicious activities such as forced updates, unreviewed merges, or unusual commit times.
Source code changes	Detect unexpected or unauthorized changes to source code (such as no PR approvals) in repositories, including modifications in configuration files.
Deployment frequency	Monitor deployment frequency for abnormal spikes or delays, which could signal unauthorized activity or system issues.
Environment variable changes	Log changes to build and deploy, including environment variables, as an attacker can manipulate them to inject malicious payloads.
Credential and secret access	Track access to secrets or credentials during the build process; unusual patterns may indicate compromise.
Anomalous network activity	Monitor build system network activity for unusual outbound traffic or connections to unknown IP addresses.
Pipeline stage failures	Analyze frequent or unusual failures in specific pipeline stages, which may indicate tampering.
Unsigned or mis-signed artifacts	Verify that all build artefacts are signed and that the signatures match trusted certificates. Unsigned artefacts are a red flag.
Build system resource usage	Look for unusual spikes in CPU, memory, or disk usage during builds, which could indicate malicious activity.
Third-party tool activity	Audit third-party integrations for unexpected behaviors or access patterns during builds and deployments.

Table 3.10: Sample build metrics

Tools and plugins

We have discussed various tools and plugins throughout this chapter. [Table 3.11](#) is a summary of all the steps and some of the popular tools that are in use:

Category	Description	Tool/integration(s)
Build tools	Continuous integration environments for building software and integrating automated tests.	GitHub Actions, GitLab, Jenkins, CircleCI, Bamboo, BitBucket, Azure DevOps, AWS Code Build, Travis CI, GoCD
Deployment tools	Continuous deployment solutions with support for automated testing, rollbacks.	ArgoCD, Harness, Azure DevOps, AWS CodeDeploy, GoCD, Spinnaker, CircleCI
SAST	Static code analysis and detection of vulnerabilities, security flaws in	CodeQL, HCL AppScan, Checkmarx, Sonar, Snyk, Fortify Static Code

	multiple languages; integration with IDEs like IntelliJ and VS Code.	Analyzer, Semgrep CORE, NowSecure, App-ray
SCA	Software composition analysis scans software for open-source and third-party components to identify vulnerabilities.	JSFrog Xray, Snyk, Black Duck, OWASP Dependency-Check, Nexus Sonatype
OSS scanning	Identify open-source software and license.	JSFrog, Snyk, Black Duck, WhiteSource Bolt, FOSSA, Retire.js, Sonatype
DAST	Dynamic application security testing tools identify vulnerabilities in applications, typically by integrating with browser-based environments for real-time testing.	Burp Suite, Acunetix, Netsparker, Cobalt Strike, OWASP ZAP, Checkmarx One, Veracode, Rapid7, HCL AppScan, NowSecure
Secure defaults framework	Pre-configured authentication and authorization for Java-based applications.	Spring Security, Express.js Security Middleware, ASP.NET Identity, Django Security Features
IDE security plugins	Code quality and security analysis; available for JetBrains IDEs and VS Code.	Codacy, DeepSource, Sonatype Lift, Snyk, Sonar
Secrets scanning	Scanning for sensitive information (such as passwords, API keys) committed into code repository.	GitLeaks, GitGuardian, Truffle hog, Vault Radar, GitHub
Functional testing	Automated functional, regression, integration.	Postman, Selenium, Cucumber, Cypress, SauceLabs, AppKnox, QAWolf, BrowserStack, Meticulous, BlazeMeter
Performance	Automated performance testing of the application.	JMeter, Gatling, Artillery, BlazeMeter

Table 3.11: Tools to secure code, build and deploy

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Tools or processes:** Often, teams and organizations focus on adopting tools rather than the end-to-end process. Just procuring tools that are not fully operational and integrated into the day-to-day processes to make them secure and effective is pointless. So, ensure you start small and slowly incorporate all the tools to achieve the outcome of better and safer software and environments.
- **Minimum tools:** Ensure developers use IDE plugins to catch common

errors using SAST and SCA plugins. Identify a minimum set of tools for your CI/CD toolchain that offer the best coverage across all the key controls. Always focus on shift-left; this means focusing on controls earlier in the steps.

- **Not just best of breed:** Organizations or leaders typically select the best tools for each capability, making it expensive and may not achieve the desired outcomes. It is better to identify tools based on what is suited for your organization and teams, the types of languages you use, the software you build, and how the tools interoperate with your technology landscape and other tools you plan to use. Favor tools that rely on open standards as they will integrate better.
- **Prevent proliferation:** Giving teams the autonomy to select what tools work for them is good. However, this might also lead to too many tools for specific use cases; you must monitor consistency and ROI. There is no simple solution (such as mandating a single tool across the entire organization), but it needs a better governance and review process.

Automation and security

The core concepts covered should be sufficient to secure your code in any environment. There is another topic that typically comes up quite often, and that is the importance of automation in security. Often, this is debatable, whether automation helps secure systems or creates potential for attacks. But this phrase from *Gene Kim* from *The Phoenix Project* sums it up:

Get humans out of the deployment business

- The Phoenix Project, by Gene Kim

The main point here is that manual processes are error-prone, slower, and inconsistent. Automation is crucial for improving efficiency, consistency, and scalability across processes. It reduces manual errors, accelerates repetitive tasks, and ensures uniformity in execution, making it especially vital in software development, testing, and deployment. Automation enables teams to respond faster to changes, improve productivity, and maintain high-quality outcomes in dynamic environments. As you have seen above, teams can automate most security checks, monitoring, and compliance through tools, which can help identify and mitigate risks early while freeing up resources for innovation. There is hardly any software configuration that

cannot be written and managed as code in a source control environment these days. So, always choose automation over manual; it is a more secure way.

Guidelines and checklists

In addition to the checklists above, here are some specific guidelines and checklists for the three roles typically responsible for maintaining the security posture of products and applications built. Use these with the checklists for a safer code, make, and deployment process. They may not be exhaustive, and you would still benefit from reading the rest of the topics and creating your versions as an extension of these checklists.

For developers and engineers

This subsection summarizes guidance for anyone developing software or solutions for customers (internal or external). Product managers, designers, and architects can also use this to better understand the key principles that need to be adopted by their team or features to be built:

- **Secure requirements and design:**
 - Is threat modelling required?
 - Are compliance requirements understood
 - Are secure coding guidelines for the language understood and applied?
- **Secure coding (Data validation):**
 - Validate all inputs
 - Sanitize outputs
 - Implement proper encoding
 - Use parameterized queries
- **Secure coding (Authentication and authorization):**
 - Implement MFA for all user-facing services
 - Use secure session management
 - Implement role-based access control

- Use secure password storage
- **Secure coding (Communication security):**
 - Implement proper TLS (viz HTTPS only)
 - Secure API endpoints, including RBAC
 - Implement rate limiting
- **Secure coding (Error handling):**
 - Implement secure error handling
 - Use custom error messages
 - Log security events
 - Implement proper exception handling
- **Secure code and reviews:**
 - Sensitive credentials stored in secret management solutions
 - Conduct peer review on every change using this and other security requirements checklists
- **Secure build:**
 - Third-party libraries reviewed
 - Review dependency vulnerabilities
 - OSS licensing implications understood
- **Secure deploy:**
 - Is the application built to support zero-downtime releases (using blue-green, canary)
 - The application should support scaling

For infrastructure engineers

This subsection summarizes guidance for anyone who deals with infrastructure (either on-premises or cloud). Product managers, designers, and architects can also use this to better understand the key principles that need to be adopted by their team or features to be built:

- **Requirements, design and training:**

- Rightsizing environments
- Participate in requirements and designs, and review alignment of security and operational needs
- Educate developers on environments, tools and best practices
- Ensure environmental isolation
- **Secure code:**
 - Use secure IaC templates for provisioning
 - Scan IaC for misconfigurations
 - Enforce source control, signing of commits, and peer review controls
- **Secure build and scans:**
 - Automate checks and build processes
 - Least privilege access controls
 - Review dependency vulnerabilities
 - Apply code quality gates
 - Automated coverage and build failures post-unit, functional, and non-functional tests
 - Apply policies regarding vulnerabilities and OSS use
- **Secure deploy:**
 - Automate checks and deployment processes
 - Support necessary release strategies based on application requirements
 - Ensure the security of the supply chain before deployment
 - Implement and test necessary rollback procedures
- **Monitoring:**
 - Set up centralized logs and metrics
 - Provide the ability to visualize metrics
 - Set up alerting for key KPIs
- **Security compliance:**
 - Ensure security and compliance oversight from external

members/teams if necessary

- Enforce least-privilege access controls

For leaders

This subsection summarizes guidance for anyone who leads or manages a team; this could be product managers, designers, and architects. It also benefits people by supporting organizational functions, such as risk and compliance, that interact with product teams:

- **Security planning and requirements:**
 - Ensure all planning includes key security requirements and considerations
 - Ensure all security and compliance SMEs are part of the planning
 - Prioritize sufficient technical debt remediation and automation
- **Security culture, training, and awareness:**
 - Plan regular security training for teams
 - Identify security champion(s) and raise awareness
 - Foster a culture of security awareness
- **Tools and process:**
 - Invest in automation for build, scans, testing and deployment
 - Enforce policies around reviews, testing and access controls
- **Investment:**
 - Invest in the automation and tools for developers and infrastructure engineers
 - Provide tools, training, and resources necessary for secure development and deployment
- **Compliance and risk:**
 - Review dependency vulnerabilities
 - Ensure adherence to appropriate compliance standards within the enterprise and externally, such as GDPR, PCI DSS, and HIPAA
 - Conduct periodic audits

- Raise appropriate risk and review risk mitigations, exceptions regularly
- **Metrics and KPIs:**
 - Review vulnerability trends and other CI/CD monitoring KPIs
 - Security debt tracking
 - Report key KPIs to senior leaders

Conclusion

The chapter emphasizes the importance of secure software development, starting with the SolarWinds hack as a significant supply chain attack. It underscores the need for robust security practices throughout the software development lifecycle, including secure requirements, design, coding, build, scanning, testing, deployment, and continuous integration. A secure software development lifecycle is crucial for ensuring security topics, including defining secure requirements and design upfront; secure coding practices using standards such as OWASP top 10, peer reviews; secure build, scan, and test using build agents, robust access controls, SCA and OSS scans; and secure deployment using secure configurations, benchmark tests, and signed artifacts. The next chapter will focus on the second pillar: secure your identities and access.

Points to remember

- Secure requirements and design, such as threat modeling, STRIDE and MITRE ATT&CK, help identify and mitigate potential security weaknesses in applications. CVSS scoring system quantifies the severity of security vulnerabilities, guiding prioritization and resource allocation during threat modeling. OWASP top 10 is a standard awareness resource for developers and web applications.
- Secure coding practices include using checklists to validate code quality, functionality, security, performance, error handling, dependencies, and test coverage. Secure code reviews are vital for identifying vulnerabilities and ensuring software integrity.

- Secure build pipelines, including isolated build environments, secure build agents, robust access controls, commit signing, and secure package management, should be followed to ensure the integrity and security of software delivery.
- Secure scans and tests are also crucial for identifying and mitigating vulnerabilities throughout the software development lifecycle. These tests, including functional and non-functional assessments, ensure system resilience and compliance with security standards. Incorporating these tests into CI/CD pipelines ensures continuous security validation and reduces risks.
- Secure deploy tools like SAST, dependency scanning, and secure default frameworks, at the same time, emphasize on the need for automation, secure coding practices, and a culture of security awareness throughout the development process.
- Continuous integration and deployment monitoring should include tracking core KPIs and logs for suspicious activities, periodic audits, and risk management.
- Favor automation over manual intervention for improved security

References

1. https://www.cs.cmu.edu/~rdriley/487/papers/Thompson_1984_ReflectionsonTrustingTrust.pdf
2. <https://www.cfcs.dk/globalassets/cfcs/dokumenter/rapporter/en/CFCS-solarwinds-report-EN.pdf>
3. <https://rmcglobal.com/wp-content/uploads/2022/08/2020-SolarWinds-Hack-A-Case-Study-of-the-Russian-Cyber-Threat-July-2021.pdf>
4. <https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know>
5. <https://www.npr.org/2021/04/16/985439655/a-worst-nightmare-cyberattack-the-untold-story-of-the-solarwinds-hack>

6. <https://orangematter.solarwinds.com/2021/01/07/our-plan-for-a-safer-solarwinds-and-customer-community/>
7. <https://thehackernews.com/2020/12/new-evidence-suggests-solarwinds.html>
8. <https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/>
9. <https://www.practical-devsecops.com/what-is-stride-threat-model/>
10. <https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/>
11. <https://cheatsheetseries.owasp.org/index.html>
12. <https://vulncure.com/wp-content/uploads/2024/04/CVE-2023-34362.pdf>
13. <https://unit42.paloaltonetworks.com/threat-brief-moveit-cve-2023-34362/>
14. <https://labs.k7computing.com/index.php/cve-2023-34362-moveit-transfer-exploitation-analysis/>
15. <https://cloud.google.com/blog/topics/threat-intelligence/zero-day-moveit-data-theft/>
16. <https://www.progress.com/trust-center/moveit-transfer-and-moveit-cloud-vulnerability>
17. https://owasp.org/www-community/Threat_Modeling_Process
18. <https://docs.oracle.com/javase/8/docs/api/java/sql/PreparedStatement.html>

CHAPTER 4

Secure Your Identities

Introduction

The second pillar is focused on securing **identity and access management (IAM)**; some say this is a fundamental part of security within any organization, and we will shortly see why. However, when securing identities, we need to consider multiple types of threat actors. Firstly, the internal user (employees and other workers) threat actor can access internal systems. This is critical as they typically have much higher access, and if their identity is compromised, it can have severe consequences. The second type is an external user or threat actor (e.g., a consumer, someone linked to the state, cybercriminals, or other threat actors) who can access the system regularly and elevate their permission to gain access. Some users are also third parties (e.g., external firms or contractors) with access to enterprise systems for their work. Lastly, system or API users, such as an application or system accessing your APIs internally or externally, typically overlooked, and do not change their credentials as frequently, allowing a threat actor to gain access to system data.

This chapter will cover the security of all these different identities.

Structure

The following topics are covered in the chapter:

- Case study: Uber
- Strong authentication
- Access controls
- Securing privileged access
- Incident response and recovery
- Secure federated identities
- Identity governance and compliance
- Securing APIs and systems
- Policy design
- Continuous monitoring and response
- Emerging technologies
- Tools, plugins and technologies
- Guidelines and checklists

Objectives

This chapter will discuss multiple approaches to securing identities (credentials, authentication), how and what users can access (RBAC, ABAC), security of APIs and system flows, and ongoing governance and review policies. Lastly, we will discuss how to continuously monitor identity and access, and some of the upcoming trends and technologies.

This chapter also covers many examples of securing identities and access. Most are based on Microsoft Entra (formerly Azure AD), the most widely adopted solution.

We will refer to sensitive data frequently here, and the definition will be discussed in [Chapter 6, Secure Your Data](#). However, for now, you can think of it as any data that needs to be protected. Sensitive data typically includes information about users (names, addresses, dates of birth, or health and financial information).

Case study: Uber

Let us look at the details to better understand the case study:

- **Date:** Oct 2016, **Countries:** Multiple (UK, US, Netherlands)
- **Summary:** Attackers accessed the personal information (names, contact details, driver numbers, payment information) of 57 million Uber users. However, this story involves multiple breaches and lax controls. Uber agreed to settle the case with the **Federal Trade Commission (FTC)** for \$20 million.^[3]

Detailed timeline

In May 2014, an intruder accessed Uber's Amazon S3 datastore using a publicly posted access key and granted full admin privileges to all data and documents. The intruder accessed the PII data of Uber drivers, including 100,000 unencrypted names and license numbers. The breach was not discovered until September 2014.^[1]

Between October 2016 and November 2016, intruders accessed S3 using an access key posted on a GitHub repository. Uber's policy did not prohibit users from reusing credentials and enabling MFA and personal accounts. This time, attackers downloaded 16 files with unencrypted customer personal information, including riders and drivers (25 million).^[1]

In November 2016, attackers contacted Uber, claiming they had compromised its database. Uber paid \$100,00 through a third party that administers bug bounty programs, but Uber failed to disclose this to the FTC.^[1]

In addition, Uber has also been alleged to have committed the following ^[1]:

- The AWS S3 data was not sufficiently secured. Engineers had direct access to it, shared access keys were used with full administrative privileges, and role-based access and multi-factor authentication were lacking.
- Sensitive data was stored in cleartext without encryption.
- There was no security training and guidance for employees or a written information security program.

In addition, there were claims that Uber employees had access to an aerial tracking tool known as *God View* that displayed the personal information of Riders using Uber's services.^[1]

Learnings

This case has many different learnings that will come up in various chapters of the book, as it involves multiple levels of failures:

- MFA was not mandated for the code repositories, potentially leading to compromised data store credentials.
- Access management was not implemented correctly to restrict access based on employees' roles, both on the datastore and among some system employees.
- Encryption of sensitive data was not in place, making it easier for attackers to use data they gained access to, as it was in clear text.
- There was a lack of information security policy and awareness programs across the organization. This is critical and necessary for good security hygiene.
- Lastly, security incident management and reporting were inadequate, as Uber failed to report that they had paid a bug bounty program.

Let us look at some other notable examples.

Case study: Plex

In August 2022, Plex, a media streaming platform, issued password reset notices as an unauthorized party had access to data, including usernames and encrypted passwords. Although passwords were encrypted, brute-force attacks could allow attackers to gain access if passwords were simpler. This highlights the importance of MFA and strong passwords. ^[7]

Case study: Cash App

A disgruntled employee from Cash App downloaded 8.2 million users' information (names, account numbers, stock trading activity) in April 2022. Although this did not include any significant sensitive data (such as address and payment data), it was found that the employee had access to data that was beyond their job responsibilities. This is an essential example of insufficient access controls. ^[7]

Case study: Target

The Target breach covered in [Chapter 2, Architecture and Pillars of Zero](#)

Trust, also underscored the importance of robust authentication mechanisms. The attackers used multiple levels of credentials to breach the perimeter, including credentials for the billing system they gained access to via a phishing attack and local credentials to browse the active directory. They also created new admin credentials using stolen credentials. This emphasizes the importance of protecting credentials, stronger authentication protocols, and access management.

Strong authentication

Strong authentication is the most foundational element of securing identities. Authentication is how a user or a system proves their identity. This typically includes a username and password, but can involve other factors; we will discuss various options to bolster it:

- **Strong passwords:** Strong password policies involve requirements on length, alphabet, numbers, and special characters. 80%+ of data breaches involve weak passwords, susceptible to attacks such as brute force. The following are key requirements as per ISO 27001 guidance:
 - **Password strength:** strong and hard to guess, and should not be based on personal information like names or birthdates, at least eight characters long, and ideally 12 or more. Passwords should include upper and lower-case letters, numbers, and special characters.
 - **Password management:** Users should not share their passwords with anyone else and should change them regularly, ideally every 60 to 90 days.
 - **Masked password:** The password is masked or displayed as asterisks or other abstract characters, and not plain text.
 - **Idle sessions:** Idle login sessions should end after a set time, especially for remote work or user-owned devices.
 - **Authentication session duration:** An authenticated session should be restricted depending on the data accessed.
 - **Password managers:** Organizations should use password managers that (*Ipassword, Norton, NordPass, Google, Microsoft*) allow users to create and share strong passwords securely.

- **Default passwords:** When new software is purchased and deployed, the default passwords should be changed

Real-world relevance

These requirements apply to users and developers building software when setting up credentials. Ideally, when you use an external IAM solution (such as O365, Azure Entra, Google Identity Provider, or Ping Identity), the provider (Microsoft, Google) secures the credentials and allows further customization. The following are the settings for Entra ([Figure 4.1\(a\)](#)) and Google Workspaces ([Figure 4.1\(b\)](#)); since it comes with many default protections, it provides fewer customizations:

The screenshot displays the 'Authentication methods | Password protection' settings page in the Microsoft Entra ID Security console. The left-hand navigation pane includes sections for 'Manage' (Policies, Password protection, Registration campaign, Authentication strengths, Settings) and 'Monitoring' (Activity, User registration details, Registration and reset events, Bulk operation results). The 'Password protection' section is currently selected. The main content area is titled 'Custom smart lockout' and contains the following settings:

- Lockout threshold:** A text input field containing the value '6'.
- Lockout duration in seconds:** A text input field containing the value '1800'.
- Custom banned passwords:**
 - Enforce custom list:** A toggle switch set to 'No'.
 - Custom banned password list:** An empty rectangular box for listing custom banned passwords.
- Password protection for Windows Server Active Directory:**
 - Enable password protection on Windows Server Active Directory:** A toggle switch set to 'Yes'.
 - Mode:** A toggle switch set to 'Enforced'.

Figure 4.1(a): Entra authentication

The screenshot displays the 'Password management' settings page. At the top, it says 'Configure password policies for your organisation'. A yellow warning box states: 'These policies don't apply in some cases, such as when users are authenticated by a third-party identity provider. [Learn more](#)'. The settings are organized into sections: 'Strength' (with a 'Learn more' link and a checked 'Enforce strong password' checkbox), 'Length' (specifying a range from 8 to 100 characters), 'Strength and length enforcement' (with a checked 'Enforce password policy at next sign-in' checkbox), 'Reuse' (with an unchecked 'Allow password reuse' checkbox), and 'Expiry' (showing a 'Password reset frequency' of 60 days with a dropdown arrow).

Figure 4.1(b): Google authentication

Multi-factor authentication

Passwords are considered a knowledge factor in authentication; typically, more secure forms of authentication rely on additional factors. There are three forms:

- **Knowledge factor:** Something you know, e.g., a password.
- **Possession factor:** Something you have, e.g., a mobile phone, FIDO key, or authenticators.
- **Inherence factor:** Something you are, e.g., a fingerprint.

MFA requires multiple factors besides the username and password to authenticate users. The factors are expected to be across different forms. This reduces risks associated with credential theft and has recently emerged as a crucial tool in enhancing security. However, MFA alone does not necessarily eliminate all the risks related to credential stealing.

Real-world relevance

The following settings show how to set up multi-factor authentication within Entra. The first screen (*Figure 4.2(a)*) shows that only Passkey and Authenticator are enabled. The second screen (*Figure 4.2(b)*) shows that Microsoft will manage and allow the most secure authentication method that the user has set up:

Authentication method policies

Use authentication methods policies to configure the authentication methods your users may register and use. If a user is in scope for a method, they may use it to authenticate and for password reset (some methods aren't supported for some scenarios). [Learn more](#)

Method	Target	Enabled
▼ Built-In		
Passkey (FIDO2)	All users	Yes
Microsoft Authenticator	All users	Yes
SMS		No

Figure 4.2(a): MFA methods

System-preferred multifactor authentication

This setting designates whether the most secure multifactor authentication method is presented to users. [Learn more](#)

Note: If the feature status is set to Microsoft-managed, it will be enabled by Microsoft at an appropriate time. [Learn more](#)

State ⓘ *

Microsoft managed ▼

Include

Exclude

Target *

☒ All users

☐ Select group

Figure 4.2(b): MFA preference

By requiring a factor that either the user possesses or is (inherence), MFA reduces the risk of unauthorized access even if passwords are compromised and provides an additional layer of security for sensitive operations. As we saw in the Target and Uber case studies, it also mitigates the impact of phishing attacks and credential theft.

Common pitfalls and solutions

Some of the common pitfalls and potential solutions are listed as follows:

- **Storage of credentials:** Security of the authentication is paramount to ensuring the security of all the protected systems. It is thus important to secure the credentials safely. We will cover hashing and encryption techniques in [Chapter 6, Secure Your Data](#), which are critical to ensuring secure storage.
- **Build versus buy:** There are many secure identity systems with support for configurable policies. It is best not to try and build an IAM system unless the system is the core feature of your product. Use solutions such as Microsoft Entra, ForgeRock, Ping Identity, and Okta.

Access controls

In addition to securing identities, one must focus on how these identities are used and what access the users will have to a system or application. It is essential to ensure users can only perform the actions they need to perform for their day-to-day operations. Access control is the technique that decides who or what a user can view or update in an environment. Effective access control and policy management are crucial for security in federated identity environments. Access control is often used as a broad term for authorization, authentication, access, management, and auditing. This section, however, will focus on authorization and access specifically; the rest of the topics are covered separately in the book.

Least privilege access

Least privilege access (LPA) or **principle of least privilege (PoLP)** is a fundamental concept in cybersecurity that advocates for granting users the minimum levels of access, or permissions, needed to perform their job

functions. It is based on user roles and consents and categorizes users based on job responsibilities. The alternative (open access permissions) is what we saw in the Uber case study, where users had much broader permissions to view confidential data about their drivers, increasing the risk of attacks and threats. Key aspects of LPA are:

- Limiting access rights for users to the bare minimum necessary.
- Regular review and adjustment of access rights.
- Implementing time-based or context-aware access controls.

This principle significantly reduces the risk of insider threats and the potential damage from compromised accounts. Regular reviews and time-based controls will be covered later in this chapter. The best way to apply this when building applications with Zero Trust is as follows:

1. Define granular permissions based on features offered and what accesses would be needed (such as view only, write, download, etc.).
2. Map these permissions to a set of roles and keep this flexible to define new roles in the future.
3. Allow end users to map the roles to users to control who has what access.
4. Implement authorization logic on the front-end and back-end (APIs) to ensure users can only perform approved actions within the feature.

Real-world relevance

The real-world relevance of least privilege access is as follows:

- When assigning permissions to people, always ensure they need the requested access for the duration they require.
- When building software, evaluate permissions to ensure LPA can be set up correctly. LPA applies to all system-to-system integrations, but we will cover that later. [Table 4.1](#) is a sample matrix for a payroll system that shows user management and payroll (view, edit, approve) functionalities:

Role	Feature	Permission	Action
Administrator	User management	Add	Yes
Administrator	User management	Edit	Yes
Administrator	User management	Remove	Yes

Administrator	View payroll	Read	Yes
Administrator	Edit payroll	Edit	Yes
Manager	View payroll	Read	Yes
Manager	Approval payroll	Write	Yes
Employee	View payroll (Self)	Read	Yes

Table 4.1: Role feature permission map

If you need to create a new role to manage approvals in the future, you can add a new role and appropriate features and permissions.

Role-based access control

Role-based access control (RBAC) is a model that is an extension of **least privilege access (LPA)** and helps organizations define and enforce granular access policies across systems. RBAC restricts system access to authorized users based on their organizational roles. It simplifies the management of permissions and improves overall security. Key features of RBAC include:

- Defining roles that align with job functions or responsibilities.
- Assigning permissions to roles rather than individual users.
- Simplifying user management by grouping users into roles.
- Facilitating compliance with security policies and regulations.

Real-world relevance

The real-world relevance of RBAC is as follows:

- It is widely used in enterprise systems due to its scalability and ease of administration. [Figure 4.3](#) is an example of all reader roles within Microsoft Entra. Some of these roles are considered privileged, and we will cover this topic in more detail.
- When building software, ensure you define features and permissions at the right granularity, allowing the creation of roles that combine these features and permissions. Separate the assignment of roles to users from the permission-to-role mapping. See the worked-out example in [Table 4.1](#) for the LPA.

Role	↑↓	Description	Privileged
☐ Directory Readers		Can read basic directory information. Commonly used to grant directory read access to applications and guests.	
☐ Global Reader		Can read everything that a Global Administrator can, but not update anything.	PRIVILEGED
☐ Message Center Privacy Read...		Can read security messages and updates in Office 365 Message Center only.	
☐ Message Center Reader		Can read messages and updates for their organization in Office 365 Message Center only.	
☐ Reports Reader		Can read sign-in and audit reports.	
☐ Security Reader		Can read security information and reports in Microsoft Entra ID and Microsoft 365.	PRIVILEGED

Figure 4.3: Entra reader roles

Attribute-based access control

Although RBAC is a simple and widely adopted approach, it has a few limitations. The following table compares alternatives:

Role-based access control	Attribute-based access control
Helps organizations define and enforce granular access policies across systems. RBAC restricts system access to authorized users based on their organization roles	It is a more flexible and fine-grained access control model that uses attributes to determine access. These attributes can relate to the user, the resource, the action, or the environment.
It uses a static set of roles and permissions that can be assigned to individuals.	It is based on dynamic and context-aware; in the example, it uses department.
Limitations: • RBAC relies on predefined roles, it also does not permit granular resource-level permissions that may be useful in certain use cases • With the increase in roles, it can become more complex over time, with thousands of roles, there can be confusion on when to use which ones. • Not easy to scale with changing resources or features over time. The roles must be updated to support the new resources.	Benefits: • Allows the creation of much more complex policies based on multiple attributes. • Increased flexibility compared to RBAC for handling diverse access scenarios. • Better support for implementing the principle of least privilege due to its granularity.

Table 4.2: Attribute-based and role-based access control comparison

Real-world relevance

Imagine a file or a data store with resources across departments or teams (HR, payroll, payments, finance, etc.), and you want to limit access within specific departments. You can create separate roles per department/team or assign a department/team attribute to the data store (this is easily possible in cloud environments). This attribute can now be an extension to your RBAC to dynamically define access based on the user's department rather than creating separate roles per department. This approach also now scales with new data stores and departments.

The sample in the code block here shows an ABAC policy within AWS. The policy allows principals to create, read, edit, and delete resources, but only when those resources are tagged with the same key-value pairs as the principal. ^[9] In short, this means that if someone is trying to create a secret, the user's project, team, and cost center must match that of the resource they are making. This will limit them from creating resources for some other team or project.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid":
        "AllActionsSecretsManagerSameProjectSameTeam",
      "Effect": "Allow",
      "Action": "secretsmanager:*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/access-project":
            "${aws:PrincipalTag/access-project}",
          "aws:ResourceTag/access-team":
            "${aws:PrincipalTag/access-team}",
          "aws:ResourceTag/cost-center":
            "${aws:PrincipalTag/cost-center}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
```

```

        "access-project",
        "access-team",
        "cost-center",
        "Name",
        "OwnedBy"
    ]
},
"StringEqualsIfExists": {
    "aws:RequestTag/access-project":
"${aws:PrincipalTag/access-project}",
    "aws:RequestTag/access-team":
"${aws:PrincipalTag/access-team}",
    "aws:RequestTag/cost-center":
"${aws:PrincipalTag/cost-center}"
}
}
}
]
}

```

Segregation of duties

SoD is a common term one hears in large organizations within regulated industries. SoD policies ensure that no single individual has excessive control over a process or access to sensitive information that could lead to abuse or unauthorized actions. Let us take the example of an airport; typically, the check-in staff work for the airline. However, the security personnel who handle airport security checkpoints and screening are different. One reason for this is efficiency, but the other reason is also SoD. Imagine if they were all run by the same people or entity. You could issue a ticket or boarding pass and get through the gates with some minor collusion. This helps reduce conflict of interest, unauthorized actions, and thus security breaches, making flying more secure. The same applies to virtual access and systems; many key actions may require separate people or teams to act. In financial institutions, transactions over a certain amount require manager signoffs.

The purpose and benefits of SoD are as follows:

- Prevent fraud and errors by requiring multiple people to complete sensitive tasks.
- Reduce the risk of intentional or accidental misuse of resources.
- Enhance internal controls and support compliance with regulations.
- Improves overall security posture and data integrity.

SoD is typically achieved by:

- **Rule definition:** To create and enforce custom SoD policies.
- **Access request workflows:** Implementing multi-step approval processes aligned with SoD policies, so if an action requires a manager or peer approval.
- **Continuous monitoring:** Detecting and alerting on potential SoD violations in real-time.

The challenges and considerations are as follows:

- Balancing security with operational efficiency.
- Managing exceptions for small teams or specialized roles.
- Adapting SoD policies to evolving business needs and technologies.

Real-world relevance

The real-world relevance of SoD is as follows:

- If a single person can write and deploy code without oversight, there is a risk of introducing malicious changes or unintentional errors that could compromise the entire system. It is thus essential to add key controls we discussed under secure code to ensure that code cannot be deployed without meeting all necessary controls. A good example is pull request reviews we saw in the last chapter; these approvals can be from specific teams and a reasonable SoD control. *Figure 4.4* shows how to control GitHub when someone merges into the master branch. This current policy enforces two approvals and a successful deployment for the merge. This is also why development, testing, and production environments are separated.
- When building software, carefully review each feature to identify any that might involve sensitive data or critical operations requiring SoD

and an additional approval workflow. For example, if your application includes a feature for processing financial transactions, it is essential to separate the roles of users who initiate transactions from those who approve or authorize them. This ensures that no single person can create and approve payments, reducing the risk of fraud or errors. Other examples are dividing system administration and security audit responsibilities.

Branch rules

☐ **Restrict creations**

Only allow users with `bypass` permission to create matching refs.

☐ **Restrict updates**

Only allow users with `bypass` permission to update matching refs.

☒ **Restrict deletions**

Only allow users with `bypass` permissions to delete matching refs.

☐ **Require linear history**

Prevent merge commits from being pushed to matching refs.

☒ **Require deployments to succeed**

Choose which environments must be successfully deployed to before refs can be pushed into a ref that matches this rule.

Show additional settings ▾

☒ **Require signed commits**

Commits pushed to matching refs must have verified signatures.

☒ **Require a pull request before merging**

Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

Hide additional settings ▲

Required approvals

2 ▾

The number of approving reviews that are required before a pull request can be merged.

☐ **Dismiss stale pull request approvals when new commits are pushed**

New, reviewable commits pushed will dismiss previous pull request review approvals.

☐ **Require review from Code Owners**

Require an approving review in pull requests that modify files that have a designated code owner.

☐ **Require approval of the most recent reviewable push**

Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☐ **Require conversation resolution before merging**

All conversations on code must be resolved before a pull request can be merged.

☐ **Request pull request review from Copilot** Preview

Automatically request review from Copilot for new pull requests, if the author has access to Copilot code review.

Allowed merge methods Preview

Merge, Squash, Rebase ▾

When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled.

Figure 4.4: GitHub pull request review ruleset

Common pitfalls and solutions

Some of the common pitfalls and potential solutions are listed as follows:

- **Least privilege extremes:** A common occurrence is for teams to define roles that are so granular that everyone tends to require many role assignments to gain access, and everyone lands up getting all roles after all. It is crucial to define roles based on business needs and adopt simpler, fewer, well-defined roles that allow users to perform their day-to-day operations; start small and then expand versus start big and try simplifying later.
- **SoD that slow down change:** Another anti-pattern is to implement segregation of duties in such a manner that every change either requires multiple stages of approval or involves people who do not even understand the change. When defining SoDs, include the right people who would have sufficient context to approve changes and try to keep the approval process simpler; lastly, leverage automatic approvals versus workflows that require uploads of approval evidence or emails.

Securing privileged access

In [Figure 4.3](#) of the Entra reader roles, some roles were labeled privileged. Privileged roles or accounts have elevated permissions and access. In a system, these are typically accounts with administrative permissions or roles with global access. They need special considerations and protection since they are prime targets for most attacks. Privileged access needs robust controls to limit the duration it is used for, monitoring and alerting, and detailed audit trails.

Privileged access management

Privileged access management (PAM) is the strategy that focuses on controlling and monitoring privileged access. A comprehensive PAM system/solution supports the following features:

- **IAM integration:** PAM systems must integrate identity and access management solutions to provide a centralized platform for managing

user access and privileges. This is critical for ensuring security and standardizing authentication.

- **Access controls:** PAM systems will also help limit the privileged accounts required and enforce LPA for access control.
- **Workflows:** Approval workflows are also supported based on the type of access required.
- **Just-in-time access:** PAM systems will enable time-limited access when required. Tools like Azure PIM enable this but set expiration times or duration on role/group assignments.
- **Session monitoring and recording:** PAM systems also support recording the entire session and activities, keystroke logging, screen recording, and command-level auditing for forensics and incident investigations.
- **Real-time visibility and alerting:** Administrators typically have access to real-time visibility into privileged accounts, and alerts can be set up to detect malicious activity. PAM systems provide visibility into privileged access and can raise alarms when malicious activity is detected.
- **Record and audit:** PAM systems allow administrators to control who accesses data and can produce audit logs in case of a data breach.
- **Using AI:** PAM systems can use **artificial intelligence (AI)** to analyze historical data and predict potential security threats before they occur.

A sound PAM system will provide time-based access and help detect suspicious activities, investigate security incidents, and maintain a comprehensive audit trail for compliance purposes. These systems can integrate with your IAM, infrastructure, and applications to control privileged access. Some examples are Microsoft PIM, CyberArk, and BeyondTrust.

Short-lived credentials

Short-lived credentials are temporary authentication tokens or passwords valid for a limited period, typically ranging from a few minutes to a few hours. Short-lived credentials are often used in conjunction with **just-in-time (JIT)** access and can be automatically generated and rotated by PAM

systems. Organizations can minimize the risk of unauthorized access by implementing short-lived credentials, even if credentials are somehow compromised. This method is beneficial in dynamic cloud environments and managing access to sensitive resources, as it invalidates credentials immediately after use. One such example is the Hashi Corp Vault. It enables various features such as automated password provisioning and rotation of database credentials (using dynamic secrets) and short-lived access keys or tokens (for AWS access tokens).

Secure remote access

We will cover remote access under *Chapter 5, Secure Your Network and* involve various technology solutions such as VPN and **remote desktop protocol (RDP)**. However, when it comes to privileged access, there are two areas of remote access that one should implement:

- **Just-in-time network access to infrastructure:** Usually, firewall changes are required to access infrastructure, say, remotely (using RDP). Typically, this access is only needed when troubleshooting or resolving incidents. With JIT, you can enable access to infrastructure on specific ports in a time-limited fashion.
- **Privileged access workstations (PAWs):** PAWS are dedicated workstations or devices designed to access infrastructure or applications with sensitive data. In most organizations, if your job or role requires you to access production data or infrastructure that has user **personally identifiable information (PII)** (sensitive data), that access has to be from highly restricted devices, along with detailed logging and audit, restricted operations (such as upload, download, copy, paste) and data loss prevention measures. Although this approach may be against the approach for Zero Trust, it is still a good form of security hygiene. PAW is the highest security configuration applied to the user's workstation or a separate workstation for all access to sensitive data. Privileged access to sensitive systems is denied to all other devices.

Incident response and recovery

One issue with privileged access is how it works in an incident(s) scenario.

There are three areas of incident response linked to privilege access:

- **Incident detection:** Detecting an incident caused by privilege escalation requires a sound PAM system and integration with the proper monitoring and SIEM systems, which we will cover later. The aim is to detect anomalies and suspicious activities and have automated or manual response mechanisms in place. It should also include regular audits of access logs, session recordings, and command-level activities to identify any signs of abuse or policy violations.
- **Emergency access procedures (break glass):** In an emergency, there has to be a way to assign elevated permissions. Imagine if the PAM system itself is down or unavailable. How does anyone gain access? Also called *Break Glass*, it needs to allow pre-approved individuals to obtain temporary privileged credentials through a secure, audited, and approved process. Typically, this procedure and the access are short-lived.
- **Privileged access in disaster recovery:** Like emergency access, but specifically for disasters, organizations should also establish secure, out-of-band communication channels for granting and managing privileged access during disasters. Disaster recovery is typically long-term, so access may be required for much longer.

These drills or processes should be regularly tested to be current and people aware.

Real-world relevance

The real-world relevance of incident response is as follows:

- Ensure all live or production system infrastructure access is behind privileged access controls.
- Whenever there is a need to generate access keys or credentials that can be automated and short-lived, use tools such as vault to implement short-lived. E.g., credentials to a database or cloud provider (access keys).
- When building your custom application, identify what actions privileged users (such as super admins or global administrators) might perform. For example, a user might delete a lot of data related to a group of users.

This access must be protected using the PAM system, which might require additional integrations. Every such action needs to be audited and tracked for compliance purposes.

Common pitfalls and solutions

Some of the common pitfalls and potential solutions are listed as follows:

- **Applying strict privileged access for non-privileged roles:** Occasionally, teams can go overboard and use the same strict permissions and privileged access controls to lower environments such as development, testing, or read-only roles. This is unnecessary and can drastically slow development, leading to people requiring privileged access for day-to-day operations. Apply privileged access only to higher environments where the risk of threats is genuine.
- **Lack of visibility and audibility:** Organizations often implement privileged access controls and limit access; however, there is insufficient visibility and audit trails. This makes it more challenging to monitor the effectiveness of the security controls and identify anomalies. Integrating privileged access management solutions with your security monitoring system is critical.
- **Use of static credentials:** Many traditional authentication methods are still in use, and teams tend to rely on static **Secure Shell (SSH)** keys and credentials that are not changed when accessing critical servers over using dynamic secrets. It is imperative to implement a secure rotation of these credentials and, as much as possible, integrate them with privileged access solutions.

Secure federated identities

Until now, we have only seen how to manage identities and access in the context of a single application or organization. However, organizations have multiple applications and sometimes involve numerous other organizations and third-party applications they need to access. One approach is to use separate credentials for each application, but it is neither a good experience nor secure. **Federated identity management (FIM)** is a set of standards and agreements that help solve this problem by enabling the sharing of identities

across applications and enterprises. It involves establishing trust between multiple identity providers across different organizations. However, FIM can introduce security risks by sharing user identity information across organizational boundaries.

There are three main components of identity federation:

- **Identity provider (IdP):** The identity system that manages credentials, e.g., Entra.
- **Service provider (SP):** Applications the user wants to use, e.g. payroll system.
- **Trust relationship:** A preset agreement between IdP and SP on the authentication protocol and process.

Single sign-on

Single sign-on (SSO) is a feature of federation that allows users to access multiple systems within an organization using a single set of credentials. Within your organization, you must have noticed you use standard login credentials to access various services, such as tracking, timesheets, HR, payroll, and collaboration. SSO makes logins more secure and convenient, as you do not have to manage multiple logins.

Standards and protocols

Table 4.3 covers all the standards and protocols used in identity and access management and details of what they are:

Protocol	Description
SAML 2.0	Security assertion markup language (SAML) version 2 is an open standard that allows users to authenticate and gain access using authentication proof. It is a widely used standard for federated identities and enables centralized credentials. Security is managed using XML SAML assertions between these parties (IdP and SP).
OAuth 2.0	Open Authorization version 2.0 (OAuth 2.0) is an open standard primarily for authorization that enables applications to obtain limited access to user accounts on other services without sharing credentials. It provides a secure way for users to grant third-party applications access to their resources. However, OAuth 2.0 does not provide a complete SSO experience on its own.
OIDC	OpenID Connect (OIDC) , built on OAuth 2.0, extends its capabilities by adding an authentication layer. It introduces the concept of an ID token, which contains

	claims about the authenticated user's identity. This allows applications to access resources on behalf of a user and verify the user's identity.
OpenID	On top of OAuth 2.0, OpenID is the identity layer that enables authentication and basic profile information retrieval.
JWT	A JWT is URL-safe and uses JSON to represent claims that can be securely transmitted between two parties as a JSON object.
Linked SSO	The user is sent to an external sign-in page that the identity provider does not manage. Ideal for an app already integrated with a separate SSO provider, during migration.
Password SSO	In this model, application passwords are different but managed within the identity system by an administrator or required when the user logs in for the first time. This can be a good temporary solution, especially for on-premises applications; however, it is not the best option as it still involves separate credentials.

Table 4.3: IAM standards and protocols

Figure 4.5 shows roughly how these standards and protocols overlap. As you can see, FIM is a broad term, and it includes SSO. SAML2 is primarily used for SSO. OAuth 2 has a broader purpose, and OIDC is a subset.

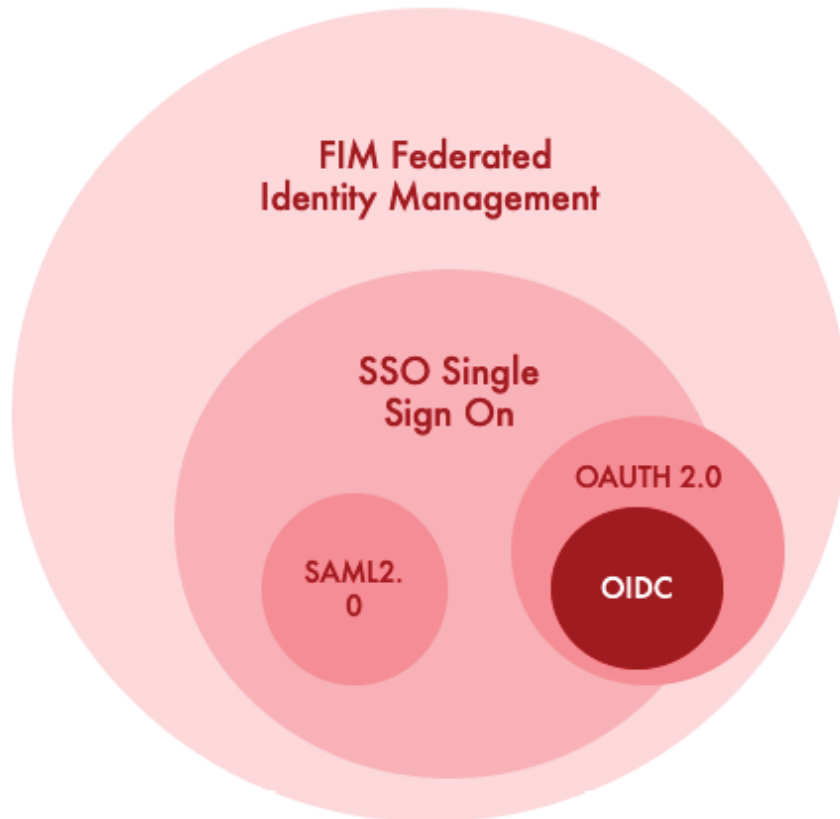


Figure 4.5: IAM standards graphic

Common pitfalls and solutions

Although FIM offers a lot of benefits, it is essential to understand some of the risks associated with:

- Attack surface has increased due to integrating different applications with a single identity and the related trust between them. Now, attackers need to compromise a single identity.
- Since all access is granted via IdP, users cannot access all connected systems if the IdP is out of service, thus leading to a single point of failure.
- Vulnerabilities and misconfigurations, such as policies on the IdP.
- Compliance and governance are challenges due to multiple legal entities and jurisdictions that may be involved.
- Privacy concerns as there is not sufficient visibility into data attributes shared between SP and IdP.

However, even with the risks, it is still more secure to federate identities due to its benefits, such as simplified user experience, cost efficiency, standardization of identity and authentication leading to better protection, and simplified administration. Here are some of the mitigations and best practices to mitigate the risks:

- Enable secure modes of authentication such as multi-factor authentication.
- Only use secure protocols SAML2 OIDC and offer access management guidelines.
- Enforce strict token validation and signature verification.
- Perform regular security audits and penetration testing.
- Comprehensive monitoring and anomaly detection systems.
- Employee training on security awareness.

Real-world relevance

Here are some real-world examples of how you can apply this guidance:

- Support SSO from the get-go for improved user experience and security. Integrating with Google, Facebook, or other social media logins is a good idea. This is a common practice these days. For B2B solutions,

support SAML2, OAUTH 2.0, and OIDC and integrate with Entra (Microsoft 365), Google Workspaces, or similar.

- Build against standards so your applications can easily be integrated with IAM systems (internal or external).
- Do not reinvent the wheel; instead, integrate tried and tested products and solutions, such as Auth0, Okta, AWS Cognito, and PingIdentity, with your products.
- Ensure you meet the audit and compliance requirements discussed previously. The application should generate detailed audit trails of key user actions.
- Implement robust error handling and clear user flows for authentication failures.

Identity governance and compliance

So far, we have covered the aspects of securing identities and access. However, what happens to the identity or access once it is issued? When a person joins an organization or a team, they are typically issued an identity and granted access, but what happens when they leave or change position? Security does not just stop at the issuance of access; it needs to look at the complete lifecycle. Otherwise, people would accumulate persistent access over time, making their credentials prone to attack and source of compromises.

Identity lifecycle management

Identity lifecycle management (ILM) looks at this aspect: managing a user's digital identity from joining an organization until they leave. As people's roles change, their permissions and access must be adjusted to keep users productive and secure. This process is also called the **joiners, movers, and leavers (JML)** process and involves creating, updating, and deleting digital identities for users, partners, contractors, customers, and applications. An identity lifecycle management solution will either be part of the IAM system or will integrate with the IAM system to offer the following features:

Feature	Details
Employee onboarding	Support the creation of new user accounts and grant them access based on

	expected roles and responsibilities.
Approval process	Enable custom approvals based on the types of roles requested and the ability to bypass if required.
Audit, reviews and reporting	Audit trails for all processed requests, regular reviews of access, and reporting to meet compliance requirements.
Self-service	Enable users to self-serve requests with appropriate approvals where required.
Integrations	Support integration with IAM systems such as Microsoft Entra, Active Directory, and Ping Identity and intrusion detection and monitoring solutions.
Leavers/de-provisioning	Support for the removal of access and deactivation or deletion of accounts.

Table 4.4: *Identity and lifecycle management features*

A good ILM will improve security and reduce administrative costs through:

- Automation of onboarding and offboarding through integrations.
- Ensuring access is timely so people are productive immediately.
- Managing external users so the process is standardized.
- Adjusting permissions with changes in roles.

Access reviews

A good ILM will also help manage regular access reviews. This is an essential part of security certifications such as ISO27001. The access people have has to be reviewed regularly to determine whether they still require it and make updates if necessary. Typically, the manager or supervisor a person reports to is the best person to review the access of the individuals who report to them. The idea is to reduce the risk of someone having some access they do not require anymore, especially if it is privileged or grants access to sensitive data.

Figure 4.6 shows a sample of the access review within Entra. The screen shows how to set up the reviewers (who would review), the duration of the evaluation, and the frequency of the review of access. The rest of the tabs allow you to select what access should be reviewed and how to make decisions. These reviews would differ on ILM tools, but generally, the concept is similar: notify the reviewers and take action based on their access review.

New access review ...



 License requirements for this feature have changed. After October 30, 2023, access to capabilities formerly in preview will be read-only, unless you purchase the new Microsoft Entra ID Governance license. Click here to learn more. →

* Review type * Reviews Settings * Review + Create

Determine review stages, reviewers, and timeline below.

Multi-stage review ⓘ

☐

Specify reviewers

Select reviewers *

Group owner(s) ▼

Fallback reviewers ⓘ

+ Select fallback reviewers

Specify recurrence of review

Duration (in days) *

5 ✓

Review recurrence *

Quarterly ▼

Start date * ⓘ

01/02/2025 

End *

- ☒ Never
☐ End on specific date
☐ End after number of occurrences

Figure 4.6: Identity access reviews

Compliance

Compliance is a broad topic; we will cover it in more detail in *Chapters 8, Continuous Monitoring and Chapter 12, Policies, Frameworks and Compliance*. However, compliance in securing identities and access is critical to modern cybersecurity strategies. Most regulatory requirements and

standards set forth by regulators for organizations, such as the **General Data Protection Regulation (GDPR)**, **Health Insurance Portability and Accountability Act (HIPAA)**, **Payment Card Industry Data Security Standard (PCI DSS)**, Open Banking, and ISO27001, mandate strict controls over user identities, access rights, and data protection. In addition, there are a few standards that have emerged on secure identities, such as:

- **ISO 27001:2022 Annex A 5.16 Identity Management:** Establishes a framework for managing identities on any network, including human and non-human identities.
- **ISO/IEC 24760-1:** A framework for identity management.
- **Global Digital Identity Certification (GDIC):** Focused more on KYC/AML, it is a certification framework that combines ISO/IEC 29115 and **Financial Action Task Force (FATF)** recommendations.
- **Secure identity alliance:** Promotes using biometrics and other practices to enable safe, convenient, and privacy-led identification.

Organizations must implement comprehensive compliance management systems incorporating automated checks against regulatory standards to meet these requirements. These systems continuously monitor identity and access management controls, ensuring alignment with industry best practices and legal mandates. Evidence collection for audit trails and compliance reporting must be automated, providing a detailed record of access activities and policy enforcement. Regular audits form a cornerstone of compliance efforts, involving periodic reviews of user accounts and access rights to identify and rectify deviations from established policies. These audits also focus on identifying dormant or unnecessary accounts, which pose potential security risks if left unmanaged. Organizations must maintain a rigorous compliance program to meet regulatory requirements and enhance their security posture, fostering trust with customers and stakeholders while mitigating the risk of data breaches and associated penalties.

The specifics of compliance frameworks and who they apply to are part of [*Chapter 12, Policies, Frameworks and Compliance*](#), but compliance management involves the following:

- Understanding the scope of regulatory compliance requirements.
- Implementation of automated checks for applicable regulatory requirements.

- Evidence collection for audit trails and compliance reporting.
- Ongoing assessment of IAM controls against industry standards.
- Regular audits through periodic reviews of user accounts and access, identification of dormant or unnecessary accounts, and verification of compliance against policies and regulations.

Identity governance and administration

Identity governance and administration (IGA) is a collection of policies, systems, and processes organizations use to manage and control user access to critical data and systems. IGA is also known as identity security. The goal of governance is to ensure that policies such as role-based access control, segregation of duties, and other requirements laid out by the organization are followed and that any violations or exceptions are reviewed regularly to assess risks.

Since identity management can become complex with application adoption and integration and federation across third parties and other organizations, organizations typically have an IAM team that defines the policies, enforces them, assesses the need for certifications and compliance, market trends, and reduces risks. As part of governance, the team must detect and respond to manage IAM policies, policy violations, and privileges, automate workflows for provisioning requests and approvals, improve reporting, and ensure compliance. Risks are typically reported to senior management for review if appropriate. The team will also provide regular reviews, audits, and robust reporting and auditing to meet compliance needs.

Ultimately, IGA helps align users' access with business needs, internal policies, regulatory requirements, and security standards. A good IAM or ILM solution provides many of these features. An effective IGM helps reduce security vulnerabilities, prevent data breaches, and maintain a strong security posture.

Thus, a considerable focus is on ensuring continuous centralized monitoring (aspects we covered earlier) and automating manual steps for more secure, adequate controls and decision-making. Organizations can enhance efficiency and ensure that policies are consistently enforced by automating routine tasks such as user provisioning, access requests, role assignments, and periodic access reviews. For example, an automated workflow can

instantly grant or revoke access based on pre-defined rules, helping to ensure that employees, contractors, and third parties only have the appropriate level of access for their roles.

Real-world relevance

The real-world relevance of identity governance and administration is as follows:

- When building systems, one must ensure that the solution complies with the organization's IAM policies and works well with the IGM. Ensure the implementation lifecycle management functions to support joiners, movers, and leavers (secure onboarding and offboarding); the ability to grant and revoke access; and support integrations with the ILM and IAM solutions in use.
- Understand your compliance requirements and standards such as GDPR, HIPAA, ISO27001 and PCI DSS that may apply to your solution. Build with these requirements in mind.
- Implement exhaustive logging (including audit logs) for activities and integrate with the appropriate enterprise systems.
- These requirements apply when constructing a system (such as a build and deployment system or a monitoring system) for internal users; the system must streamline the lifecycle of user identities.

Common pitfalls and solutions

Some of the common pitfalls and potential solutions are listed as follows:

- **Time sensitivity:** Granting and revoking access on time is very important. In large organizations, it is common for people to continue having access once they move or leave. Implementing a good JML process is vital to ensure data safety.
- **Reliance on manual processes:** Many organizations tend to rely on overly manual processes, creating inefficiencies, the possibility of errors, and potential security gaps. As far as possible, automation should be leveraged throughout the process.
- **Users experience challenges:** A common implication is that users are limited from being fully productive due to slower identity governance

and administration, and users do not have complete visibility around how to get the access they need for their roles. Having a straightforward process and visibility is essential to avoid user frustration.

Securing APIs and systems

So far, we have mostly looked at authentication and access of human users. However, many applications and systems also access other systems, requiring similar controls. To bring this to life, let us look at a reference architecture of a payment system built on modern cloud-native architecture from AWS. Refer to the following figure:

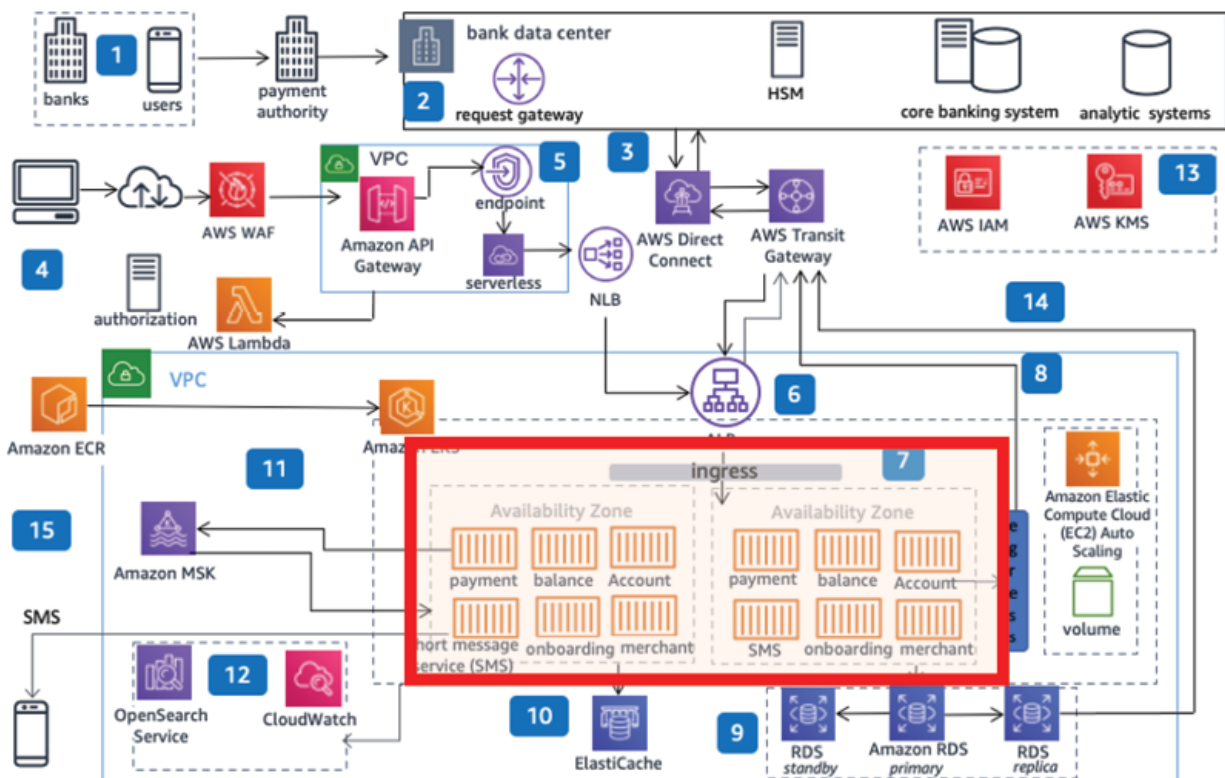


Figure 4.7: AWS payment system architecture [8]

This system uses many AWS services, but if you focus on the center of the diagram, highlighted with a red square, you will notice multiple services (payment, balance, account, and others). These are separate components that interact with each other and services outside. For every request coming from

a consumer, many of these services will need to work in conjunction. If a user were to initiate a payment, it would have to check their balance, confirm account information, check with the merchant bank, make the payment, and send an SMS. So, the question is how these internal connections between these components (microservices) and between the service and its data stores (such as RDS, ElastiCache, core banking system) or external third parties (SMS provider, payment authority) are secured?

This is where machine or system identities and service accounts come into play, and they are crucial to an organization's security posture. Protecting these identities and accounts might often be overlooked in traditional identity management strategies. These non-human accounts, used for automated processes and system-to-system communication, require special attention due to their elevated privileges and potential for abuse. Protecting these identities involves implementing robust authentication mechanisms, regular credential rotation, and strict access controls. Organizations must leverage specialized tools for managing machine identities, implement certificate management solutions, and ensure service account credentials are securely stored and regularly audited. By securing these often-neglected identities, organizations can close critical security gaps and prevent potential lateral movement by attackers.

Zero Trust model for API and microservices

The *never trust, always verify* model is particularly relevant to APIs and microservices. Unlike traditional monolithic applications, modern architectures consist of many small, heavily distributed, integrated components called microservices. The payment system in [Figure 4.8](#) shows 9-10 domains and subdomains, each of which may contain multiple microservices; a typical system involves anywhere from 50 to 100 microservices. As a result, implementing Zero Trust in each service is both labyrinthine and paramount.

There are two layers to API and Microservice security, and neither should be overlooked:

- **Public (Outward facing):** These APIs are exposed externally for public use by external businesses or end users to consume. For a bank, this is an example of the open banking APIs they need to expose to comply

with the regulations. In [Figure 4.8](#), this is marked with the number 5.

- **Partner:** These APIs work similarly to public but are used to share data with trusted partners for collaboration on products and services.
- **Private:** These are internal APIs or microservices, shown within the red block of [Figure 4.8](#). These services may not be exposed externally; however, they integrate and still need the same level of security.

Whether internal or external, the key requirements for securing the identities of APIs and microservices are the same, as listed in [Table 4.5](#). The only difference is where controls are implemented. With public and partner APIs, API gateways are typically used to expose APIs to the outside world, so it is logical to implement controls there.

Feature	Details
Machine and service identities	Any non-human entity such as IoT devices, APIs, microservices, and containers needs to authenticate itself in a network with a unique identity. Even if the microservice may be the same underlying service (say for user management or adaptor service for your payment gateway) deployed as separate instance for a dedicated consumer, each of these instances should have a separate identity. It is important to manage the lifecycle of these identities like human users.
Authentication	A robust authentication mechanism needs to be built using standards such as OAuth 2.0, OpenID Connect, and mTLS ensuring only authorized systems and users can access APIs and microservices. This includes using strong encryption and token-based authentication.
Authorization	Implementing fine-grained access controls to ensure service to service communication is controlled and managed. This means only services that should talk to each other should be allowed to, all other communication should not be allowed and return in authorization errors.
Other areas (these are covered in the chapters to come)	• Security of code, build and deploy such as OWASP top 10 • Encryption of data at rest and transit (or communications) • Network security controls such as segmentation, DDoS protection • API versioning and lifecycle management to ensure APIs are updated when released and older versions are deprecated • Rate limiting to protect APIs from abuse and ensure stability • Comprehensive audit logging and monitoring • Error handling and information exposure • Regulatory compliance (HIPAA, GDPR, Open Banking) may impose specific requirements on say APIs • Continuous testing and monitoring

Table 4.5: Zero Trust in microservices

Note that these requirements are in addition to the secure code principles we saw in the previous chapter, so requirements such as the OWASP top 10, including data validation, are still applicable. Next, we will cover different ways to secure microservices and API access.

Access keys

An access key is typically a long-lived credential to authenticate and authorize access to a service or resource. It is commonly used in systems where automated processes (like server-to-server communication) need access to resources. The access key is like a password for an API or service account. Access keys usually consist of an access key ID (public identifier for the key pair) and a secret access key (a private part of the key that must be kept secure). The secret access key is used to generate secure signatures for requests.

For example, **Amazon Web Services (AWS)** uses access keys (access key ID and secret access key) to authenticate API requests. These keys allow an application to access resources and interact with services without re-entering passwords.

Backend services or applications often use access keys for server-to-server authentication and to authenticate against an API (e.g., AWS, Google Cloud, etc.). Typically, they are long-lived and stored securely in a secret management solution.

Real-world relevance

The real-world relevance of the use of access keys is as follows:

- **Use access keys when:**
 - You need server-to-server authentication (e.g., between microservices, services, third parties, or cloud platforms).
 - Credentials are long-lived, usually rotated regularly, and monitored.
 - Automated access is needed, but fine-grained control over which resources can be accessed is not required.
- Automate the generation and rotation of access keys for better security.
- In AWS, clients or applications use access keys to authenticate API requests and interact with AWS services. See [Figure 4.8](#) for how AWS

access keys work. The user first generates an access key using the AWS IAM service. These access keys are then used to access services and perform various operations.

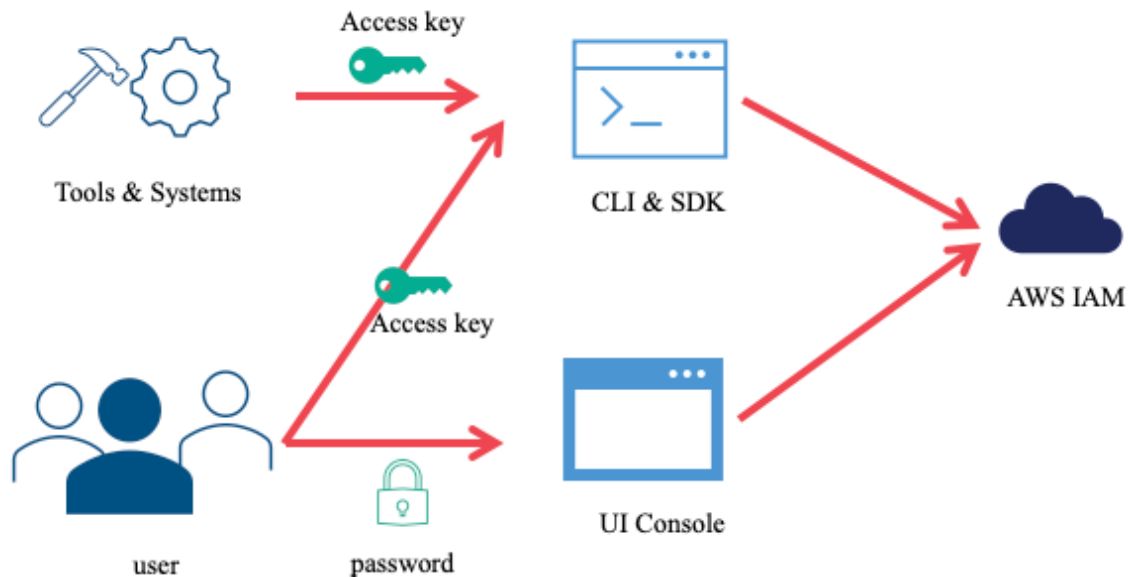


Figure 4.8: AWS access keys ^[10]

Access tokens

The second mechanism is an access token, and unlike access keys, they are short-lived credentials used to grant access to a specific resource, typically on a user's or service's behalf. These tokens are generally issued by an authorization server (e.g., an OAuth 2.0 provider) after a successful authentication process. Access tokens represent a user's or application's permission to access specific resources (e.g., a protected API endpoint) for a defined period. The most common types of access tokens are called JWTs. The token typically contains embedded claims about the user or service making the request, such as the user's role, permissions, and expiration time. OAuth 2.0 is the most common protocol these days for access.

Access tokens are used for user authentication and authorization. They are issued after users successfully log in (e.g., through OAuth 2.0 or OpenID Connect) and are short-lived (e.g., 15 minutes or 1 hour). When the token expires, the client must refresh or re-authenticate to obtain a new one. Access tokens support fine-grained authorization as they come with embedded information (e.g., scopes and roles) that can control what

resources the user or service can access.

Real-world relevance

The real-world relevance of access tokens is as follows:

- **Use access tokens when:**
 - Building a user-facing API and fine-grained access control.
 - Need to authenticate end users or services for specific actions or access to certain resources.
 - Short-lived access is appropriate to limit the risks of token compromise,
- Access tokens are generally a mechanism, but not a protocol, so you must implement OAuth 2.0 and OpenID Connect for user authentication and authorization.
- **Common use cases:**
 - **Front-end app (mobile or web):** It will use access tokens when calling back-end APIs for authorization. Every API then needs to check that the user has access to perform the operations on the data they are trying to access.
 - **Access controls:** Extending the first use case previously mentioned, access tokens enable finer access controls. The API can check the token's claims and validate that the user has the appropriate role to access the requested resource (e.g., an admin can manage users, but a standard user cannot).
 - **Delegated access:** When invoking external services that depend on user consent. This is typically seen in **Open Banking flows**. Your service might have to use OAUTH 2.0 flow to generate an access token and invoke the rest of the APIs.
 - **Single sign-on:** When a user must authenticate centrally and access multiple applications and services.

Mutual authentication

Mutual authentication or **mutual TLS (mTLS)**, or two-way authentication, is a security process where both parties verify each other's identity. In this model, the client not only authenticates the server but also

authenticates the client. This approach significantly enhances security by ensuring both parties are who they claim to be, reducing the risk of MITM attacks and unauthorized access. [Figure 4.9](#) compares TLS with mTLS:

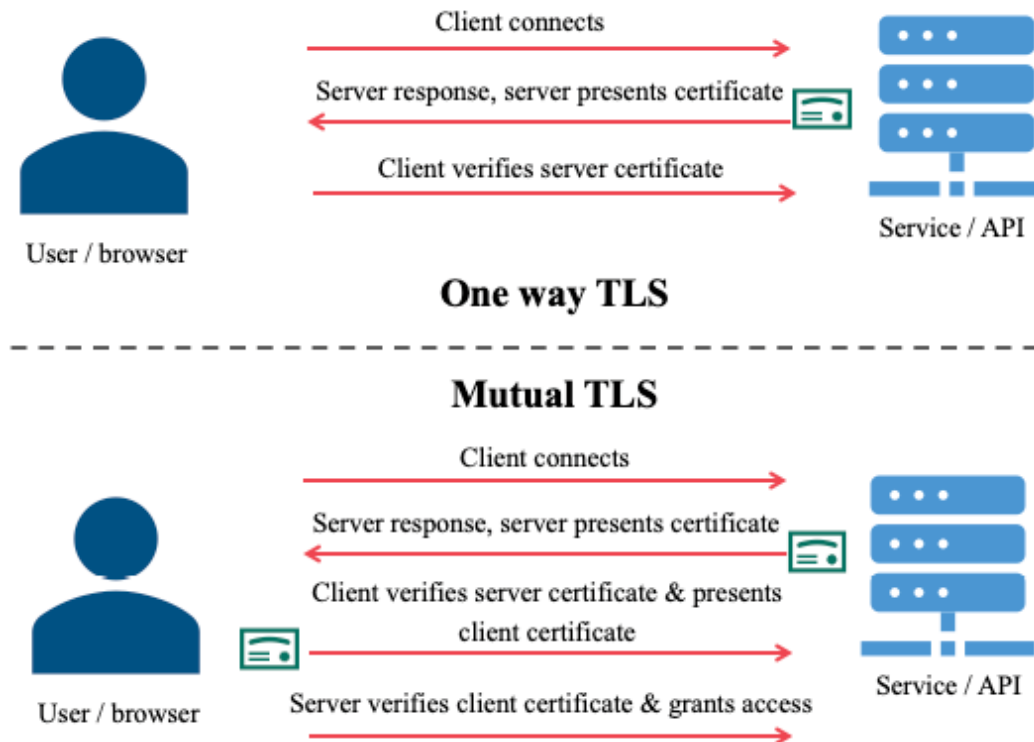


Figure 4.9: TLS versus mutual TLS

We will discuss TLS in detail in [Chapter 6, Secure Your Data](#). However, for a brief overview, refer to [Figure 4.9](#), which shows the difference between the two modes, TLS and mTLS. With TLS, the user (browser) connects to the server, and only the server authenticates by providing its certificate; the client checks the certificate's authenticity. In the case of mTLS, the server also checks the client's certificate is legitimate before allowing access.

Mutual authentication often involves digital certificates, as shown in [Figure 4.10](#). In a web application scenario, the server presents its SSL/TLS certificate to the client, and the client presents its certificate to the server. This process is particularly crucial in high-security environments such as financial institutions, where the bank's server and the customer's device must establish trust before proceeding with sensitive transactions.



Figure 4.10: Mutual TLS components

Three actors are involved in this approach: the client's web browser trying to access a service, the web server providing the service, and a certificate issuing authority that issues a certificate that both the client and server trust. *DigiCert*, *Verisign*, and *Let us Encrypt* are well-known public certificate authorities. In addition, for internal-only purposes, organizations set up a private PKI infrastructure with an internal CA. Mutual TLS operates at the transport layer of the OSI model in networking, much lower than the protocols like OAuth that operate on the application layer 7. As a result, attackers cannot interact with applications until they have passed this layer and, hence, are considered very secure as they operate at the network layer.

With all the benefits, this approach comes with some complexities. To effectively manage these identities, one must implement a robust **public key infrastructure (PKI)** or private PKI to issue and manage digital certificates for machines and services. For instance, in a microservices architecture, each service can be assigned its own identity, allowing for granular access control and secure inter-service communication. Automation plays a critical role in managing machine identities at scale, and tools such as *Hashi Corp Vault* and service meshes such as *Istio* and *Linkerd* can help with this implementation.

mTLS is best suited for internal communications between microservices or

with databases or APIs within a private network, machine-to-machine authentication (with IoT devices, Kubernetes-based container workloads), and high-security environments, where cryptographic certificates provide stronger guarantees than token-based systems.

Real-world relevance

The real-world relevance of mutual TLS is as follows:

- Use mTLS to secure all internal service communications and external communications where possible.
- Using tools like HashiCorp Vault or AWS Certificate Manager, you can automate the certificate issuance, renewal, and revocation process. This will reduce the risk of expired certificates and ensure continuous, secure operations.

OAuth 2.0 and API security best practices

OAuth 2.0 is an authorization framework we discussed earlier that enables applications to obtain limited access to user accounts on a **Hypertext Transfer Protocol (HTTP)** service. It has become the industry standard for API security, providing a secure and efficient method for granting third-party applications access to resources without exposing user credentials.

When implementing OAuth 2.0, best practices must be followed to ensure robust API security. These include using short-lived access tokens, implementing token revocation mechanisms, and utilizing refresh tokens to obtain new access tokens without requiring user re-authentication. Securing the token endpoint with strong encryption and implementing proper scope handling are crucial steps in maintaining API integrity.

Combining OAuth 2.0 with OpenID Connect adds an identity layer for enhanced security. This allows for secure user authentication and authorization, which benefits single sign-on scenarios across multiple applications or services.

Real-world relevance

In a microservices architecture, the client (a web or mobile application) will typically authenticate against an identity provider to get an identity and an OAuth 2.0 access token that includes claims and permissions specific to the

services it wants to use. Access tokens are then sent to every service the client interacts with directly or indirectly. JWTs, frequently used as OAuth 2.0 access tokens, enable services to verify their identity and authorize actions based on encoded claims.

API and system security options

Here is a quick summary of the options discussed and what they are suited for:

- **Access keys** are typically long-lived credentials that automated systems use to authenticate with an API or service, such as AWS or Google Cloud. They do not provide fine-grained access control and must be handled securely.
- **Access tokens** are short-lived credentials that authorize access to specific resources. They are typically embedded with claims like user roles or permissions. Access tokens are commonly combined with OAuth 2.0 to implement fine-grained access control.
- **mTLS** ensures secure communication using certificates for mutual authentication, typically in service-to-service communication. It provides strong security and encryption at the transport layer (Layer 3) but does not manage fine-grained authorization by itself and requires additional access control mechanisms.
- **OAuth 2.0** is an authorization framework that issues access tokens and can handle both delegated access (where one service is authorized to access resources on behalf of a user) and third-party integrations (such as granting an app access to your Google account). OAuth 2.0 integrates access tokens, refresh tokens, and authorization servers to provide secure, granular resource access.

Policy design

Securing identities and access in a Zero Trust system is typically based on predefined rules and guidelines called policies. There are various types of policies: authentication, authorization, data access, device and network, user lifecycle, application access, cloud and third party. Policy design in secure

identity and access management is a critical component that defines how access to resources is controlled and managed within an organization. [Table 4.6](#) shows the key aspects an identity and access policy should cover:

Type	Key aspects
Authentication	• Password complexity requirements • Multi-factor authentication rules • Conditional access to applications and services • Session timeout limits • Device trust assessment
Authorization	• RBAC rules • ABAC conditions • LPA enforcement
Application access policies	• Types of protocols and standards allowed for access • Application-specific access controls • API access and integration rules
Lifecycle	• Account provisioning and de-provisioning rules • Regular access review schedules • Separation of duties enforcement

Table 4.6: Key aspects of identity and access policy design

When designing these Zero Trust access policies, make sure they cover the following elements:

- **Context-aware access (based on user, device and behavior):** It should consider user attributes such as location, device status, time of access, IP Address, user behavior, and recent activity. For example, it might restrict access to a sensitive application if the user is on a personal device.
- **Policy enforcement points (PEPs):** PEPs should be clearly understood. With the Zero Trust model, policies should be enforced at every request, regardless of the type of system, and access control needs to be dynamic.
- **Dynamic access policies and real-time decision-making:** Access decisions can no longer be static. They must be based on real-time events and conditions, evaluate various risk factors, and adapt to threat levels within and outside the organization. For example, if a user logs in from two locations quickly, the policy will block or limit their access.

We will discuss policies in the future, but policies are typically implemented

in the authentication and authorization tools. In the context of this chapter, this means your IAM and PAM solutions will each have to support policies. For example, conditional access policies within Microsoft Entra, Ping Identity Authentication policies, and CyberArk Privileged Access Manager Policies. *Figure 4.11* shows a sample conditional access policy within Entra. This policy enables MFA for all users. As you can see, the policy allows you to configure multiple settings, specifically users to apply this to (all or specific groups), target resources, and network conditions (user behavior).

[Home](#) > [Conditional Access | Policies](#) >

All Enable MFA

...

Conditional Access policy

 Delete

 View policy information

Control access based on Conditional Access policy to bring signals together, to make decisions, and enforce organizational policies.
[Learn more](#)

Name *

All Enable MFA

Assignments

Users

All users included and specific users excluded

Target resources

All resources (formerly 'All cloud apps')

Network

NEW

Any network or location

Conditions

2 conditions selected

Access controls

Control user access based on their network or physical location. [Learn more](#)

Configure

Yes

No

Include

Exclude

☒ Any network or location

☐ All trusted networks and locations

☐ All Compliant Network locations

☐ Selected networks and locations

 To create a Conditional Access policy ensuring your tenant's members are coming from their compliant network, make sure Global Secure Access (GSA) is deployed and Adaptive Access Signaling in GSA is enabled in your tenant. [Learn more on how to enable GSA Adaptive Access Signaling.](#)

Enable policy

Report-only

On

Off

Continuous monitoring and response

Like your house or dwelling entry door, an IAM system is an entry door to the organization's digital space. All large organizations issue passes or entry cards for every employee and guest; you must sign in at the reception and log your times in and out on a register (or via guest pass if available). This practice plays a crucial role in security auditing and monitoring. Organizations can quickly identify unauthorized access or potential security breaches by tracking who enters and exits the building. The logs serve as a reference in case of incidents, allowing security teams to trace movement within the premises and verify the legitimacy of visitors or employees. Furthermore, these records provide valuable data for internal and external regulations to meet specific compliance requirements and standards, ensuring the organization maintains a secure and controlled environment. The higher the building or organization stakes, the more rigor you will see.

It is evident that, like a physical entry point, your IAM system requires constant vigilance, rigorous auditing, and monitoring to ensure its effectiveness and security. After all, it is the entry point to almost everything your employees and guests use, and attackers could abuse it. Identity monitoring and threat detection are critical components of a robust IAM strategy, providing organizations with the visibility and control needed to protect their digital assets. Following is the list of the main components of a continuous monitoring and response system:

- Comprehensive logging is essential for monitoring the solution. To be valuable and practical, the logging system needs to cover the following aspects:
 - A centralized logging system will streamline the process and capture detailed logs of authentication and access attempts (both successful and failed).
 - Access requests and authorizations are recorded across all systems (including changes to user permissions and roles) through various integrations.

- Continuous monitoring of identity access and lifecycle management systems are:
 - Real-time tracking of user activities and access patterns (where they log in from and what they access, etc.).
 - Automated alerts for suspicious behaviors or policy violations (repeated incorrect passwords, login from disallowed locations).
 - Monitoring of privileged account usage and administrative actions.
- Robust reporting and analytics to provide valuable insights into the overall performance and security posture. The system should support:
 - Generation of customizable reports for different stakeholders.
 - Dashboards providing real-time visibility into IAM system health.
 - Assistance in quick issue identification, troubleshooting, and resolution.
 - Trend analysis to inform policy adjustments and security improvements.
- Threat and anomaly detection using artificial intelligence and machine learning algorithms to:
 - Identify unusual access patterns that may indicate security breaches.
 - Behavioral analysis to detect potential insider threats.
 - Correlation of events across multiple systems to uncover sophisticated attacks.
 - Assistance with setting up and review of alerts for specific security events and anomalies.
 - Identify and prevent identity fraud in real-time.

Incident response is how organizations respond to threats and anomalies. When suspicious activity is detected, a streamlined process must be implemented to help respond to incidents. Specific responses can include temporarily locking the affected account, requiring additional authentication factors, or alerting security teams. Automated incident response workflows can initiate password resets, revoke access tokens, and isolate potentially compromised systems to minimize the impact of a security breach. However, large-scale attacks such as DDoS will also require manual checks and

processes to be initiated. This rapid response capability is essential in today's threat landscape, where the speed of containment can significantly influence the extent of damage caused by a security incident.

Emerging technologies

As digital devices and virtual environments become more and more prevalent, security is only getting more challenging to execute. There are various new technologies around securing identities and access to help protect organizations and users against these evolving threats. We will discuss some of the notable ones here.

Passwordless authentication

A recent trend is to move towards passwordless authentication; Microsoft recently announced it would remove passwords for 1 billion user accounts. These authentication measures are a further extension of MFA where the user does not use a password. Instead, it is replaced by secure passkeys on their phones (or keys connected to their device) with a pin or passcode. There is usually a fallback alternative when the passkey is not available. However, the IAM solution tracks user behavior, devices, and additional attributes in this scenario.

Cloud-based access control

Due to the growing number of remote workers, cloud-based systems are becoming more prevalent. Compared to typically hosted solutions, cloud-based systems offer better protection against threats as they are managed by companies specialized in this space. These companies can constantly innovate and provide updates to mitigate emerging threats. Cloud-based systems are easier to integrate into your ecosystem with more remote devices, can scale faster, and offer better compliance with changing regulations.

AI and machine learning integration

Lately, there has been a lot of innovation in **artificial intelligence (AI)** and **machine learning (ML)**, specifically with the advent of generative AI.

Chapter 10, Security in the Age of Artificial Intelligence will dive deeper into the impact of AI. However, briefly, it has had two effects: firstly, it has reduced the barrier to create sophisticated effects, and second, it is also helping identify attacks faster and better. All in all, it is revolutionizing identity and access management. Its benefits are enhancing anomaly detection and threat intelligence capabilities, enabling automated security operations with minimal human intervention, and improving real-time decision-making and adaptive access controls.

Next-generation IAM systems will provide more dynamic security posture evaluations and updates through real-time risk assessment based on user behavior and context, dynamic access rights and security policy adjustment, and continuous authentication and verification for all users and devices.

Behavioral biometric

Modern technologies and the commodification of computing due to the cloud led to emerging threats with identities, such as credential stuffing (automated use of credentials), MITM attack, and synthetic identity fraud. The use of a username and password alone is no longer sufficient. One of the modern ways to counter this is to look at unique behavioral traits when authenticating. These include how the user interacts with a device, including typing speed, mouse movements, touchscreen patterns, keystroke dynamics, mouse or cursor dynamics, signature recognition, voice dynamics, swipe gestures (on touchscreens), and device usage patterns. Behavioral biometrics can be super beneficial in identifying anomalous behaviors during transactions (say, large payments) and improving user experience.

Decentralized identity

Decentralized identifiers (DID) can identify yourself without using a central authority, like a government or a company, using blockchain-based **distributed ledger technology (DLT)**. Blockchain introduced the concept of a distributed ledger, a shared, immutable ledger that facilitates the process of recording transactions and tracking assets in a decentralized network. Although it has been associated with crypto and the downfall of crypto assets in recent years, the technology has tremendous potential due to its immutable nature, making it suitable for many security use cases. Unlike a

centralized identity system, where individuals managing the system have complete control of the system, data on decentralized ledgers can be more trustworthy. Also, the data is inherently shared and thus available to all network participants immediately, offering a complete audit trail for every change.

This has already led to new types of self-sovereign identity, and blockchain-based identity solutions are emerging. These solutions allow users to control how their data is stored and managed, giving them better control of their identity and improving privacy, simplifying onboarding by facilitating the sharing of identity across systems, and enhancing trust if governed properly.

Edge computing and IoT

It is no longer just human identities that are important; there are also devices at the edge and the **Internet of Things (IoT)**. As more devices connect to the network and act on users' behalf, traditional IAM solutions and approaches are no longer sufficient. They need to evolve to address the unique challenges this distributed ecosystem poses and the identities of countless devices operating at the network edge.

As we saw earlier, IAM solutions adapt with digital certificates for IoT devices and use PKI at scale to enable more secure authentication. There is also the adoption of a **Trusted Platform Module (TPM)** and security enclaves, which provide an area of high security to establish trust and enable device authentication. In addition, the standard practices of LPA, real-time monitoring, and emerging trends such as DID and use of AI/ML are also helping improve solutions in this space.

Regulatory change

Regulators across the globe are also taking notice of these advancements and their impact on the landscape of IAM. The digital identity system in India Aadhar has been in place for a while and has already been integrated into most systems and services. The European Union's GDPR laid the foundation for data privacy in Europe and globally. In the United States, various states are introducing digital identity initiatives, such as mobile driver's licenses, with the Department of Homeland Security working on standardizing digital ID security. In 2021, the European Union proposed a unified digital identity

framework to enable users to identify and share electronic documents across any EU member state, including a digital wallet. The UK has also created the **Office for Digital Identities and Attributes (OfDIA)** to enable trusted digital identity services; the **digital verification services (DVS)** was also announced to allow digital identities. These regulatory changes drive organizations to adopt more sophisticated IAM solutions, including multi-factor authentication, risk-based access controls, and automated identity governance to ensure compliance and enhance security.

Tools, plugins and technologies

We have discussed various tools and plugins throughout this chapter. [Table 4.7](#) is a summary of all the areas and some of the popular tools that are in use:

Area	Tools
Identity and access management	Auth0, Okta, AWS Cognito, Ping Identity.
Privileged access management	Microsoft PIM, CyberArk, and Beyond Trust, HashiCorp Vault
Federated identity	Ping Identity, Microsoft Entra, Google Identity
Identity lifecycle management	SailPoint and CyberArk
Just in time access	Hashi Corp Vault
API gateways	Google Apigee, AWS API Gateway, Azure APIM
Service Mesh	Linkerd, Istio

Table 4.7: Key aspects of identity and access policy design

Guidelines and checklists

Let us summarize some specific guidelines and checklists for the three roles typically responsible for maintaining the security posture of products and applications built. Use these with the checklists to secure identities and access to systems you create or manage.

For developers and engineers

This sub-section summarizes guidance for anyone developing software or solutions for customers (internal or external). Product managers, designers, and architects can also use this to understand better the key principles that need to be adopted by their team or features to be built:

- **Authentication implementation:**

- Implement strong authentication
- Enforce complex passwords
- Include MFA support
- Secure session management (HTTPS only, same site cookies, timeouts)
- Support SSO with enterprise IAM solutions

- **Authorization:**

- Grant minimal permissions LPA
- Support regular reviews of permissions
- Implement RBAC
- SoD for sensitive operations
- Support ABAC where appropriate

- **Privileged access:**

- Integrate PAM solution for privileged roles
- Enable JIT for privileged access
- Short-lived credentials
- Production access should be on secured devices

- **Secure API endpoints:**

- Authenticate all flows and use access keys and tokens as necessary
- Use OAuth 2.0 and OpenID Connect for authentication and authorization
- Use mTLS for all service-to-service communication
- Implement rate limiting and request throttling token security

- **Identity lifecycle management:**

- Implement support for JML in your solutions

- Support ongoing access reviews
- Understand and implement compliance requirements in scope
- **Monitoring:**
 - Generate detailed audit trails of all operations
 - Integrate a centralized logging solution

For infrastructure engineers

This subsection summarizes guidance for anyone who deals with infrastructure (either on-premises or cloud):

- **Authentication implementation:**
 - Identity provider setup
 - Directory services integration
 - SSO integrations and setup
 - Cross-domain trust
 - Implement strong authentication
 - Enforce complex passwords
 - Include MFA support
 - Enforce the use of HTTPS
 - Implement proper session timeout and invalidation
- **Authorization:**
 - Grant minimal permissions LPA
 - Support regular reviews of permissions
 - Implement RBAC for all users
 - Segregation of duties for sensitive operations
 - Setup attribute-based access control were supported by tools
- **Privileged access:**
 - Integrate PAM solution for privileged roles
 - Enable JIT for privileged access
 - Short-lived credentials
 - Enforce policies for production access via secured devices

- **Secure API endpoints:**
 - Setup policies on rotation of access keys and usage of tokens
 - Implement a robust PKI and private PKI
 - Implement proper key management and rotation
 - Use OAuth 2.0 and OpenID Connect for authentication and authorization
 - Use mTLS for all service-to-service communication
 - Configure API gateway and service meshes policies
- **Identity lifecycle management:**
 - Setup JML process
 - Setup policies and processes for ongoing access reviews
 - Understand compliance requirements and standards to meet
- **Monitoring:**
 - Setup centralized logging and monitoring solution
 - Support necessary integrations across the estate
 - Enable teams to set necessary alerts and monitoring
 - Create and maintain infrastructure alerts and monitoring

For leaders

This subsection summarizes guidance for anyone who leads or manages a team, including product managers, designers, and architects. It also benefits people from supporting organizational functions like risk and compliance that interact with product teams:

- **Security planning and requirements:**
 - IAM architecture and tools selection
 - Ensure all planning includes key security requirements and considerations
 - Ensure all security and compliance SMEs are part of the planning
 - Allocate resources for continuous improvement of security posture
- **IAM policy:**

- Define clear roles and responsibilities
- Establish request and approval processes
- **Identity governance and lifecycle management:**
 - Ensure JML processes are in place and followed
 - Establish ongoing reviews of access
- **Compliance and risk:**
 - Ensure adherence to appropriate compliance standards and policies
 - Conduct periodic third-party audits
 - Raise appropriate risk and review risk mitigations, exceptions regularly
- **Monitoring and incident response:**
 - Monitor key metrics around authentication and access patterns and failures
 - Define procedures for handling security breaches
 - Conduct regular drills and simulations
- **Security culture, training and awareness:**
 - Plan regular security training for teams
 - Identify security champion(s) and raise awareness
 - Foster a culture of security awareness
 - Stay informed about emerging threats and best practices

Conclusion

This chapter focuses on securing identities and access management and highlights the importance of strong authentication and access controls. We also examined how privileged access management is crucial for securing systems and using short-lived access to enable it. Implementing secure practices like multi-factor authentication, role-based access control, and Zero Trust models mitigates security risks. Emerging trends like passwordless authentication and AI integration will further shape the future of identity and access management. In the next chapter, we will focus on the third pillar:

secure your network.

Points to remember

- Strong authentication requires securing credentials and strong passwords and implementing multi-factor authentication.
- Enforcing access controls by leveraging least privilege access, role-based access control, and attribute-based access control to limit user permissions and mitigate security risks. SoD is also crucial for preventing fraud and errors by requiring multiple approvals for sensitive tasks.
- PAM systems control and monitor privileged access, integrating with IAM solutions for access control and monitoring. In addition, short-lived credentials, just-in-time access, and secure remote access can further enhance security around privileged access.
- Incident response for privileged access involves detecting anomalies, establishing emergency access procedures, and managing access during disasters.
- FIM enables identity sharing across applications and enterprises but introduces security risks. Implementing SSO using SAML2.0, OAuth 2.0 and OIDC are some of the secure ways of implementing federated identity.
- ILM manages user identities from onboarding to offboarding, ensuring timely access and reducing security risks. Access reviews are crucial for security certifications like ISO 27001, reducing the risk of unnecessary access. Compliance in securing identities and access is vital for modern cybersecurity strategies.
- IGA helps manage and control user access.
- Securing APIs and microservices requires a Zero Trust model with robust authentication, credential rotation, and access controls. This starts with unique identities for non-human entities, robust authentication using standards like OAuth 2.0, and fine-grained authorization for service-to-service communication. Access keys, access tokens, and mTLS are key mechanisms for securing microservices and API access.

- Automating certificate management with tools like HashiCorp Vault or AWS Certificate Manager reduces the risk of expired certificates and ensures secure operations.
- OAuth 2.0, when implemented with best practices like short-lived access tokens and token revocation, provides robust API security.
- Zero Trust access policies should consider context-aware access, dynamic enforcement, and real-time decision-making based on user, device, and behaviour.
- Some upcoming trends that will further change this space include passwordless authentication, cloud-based access control, and AI integration, which are emerging trends in IAM.
- Regulatory changes like the EU's GDPR and digital identity initiatives across various countries, such as India's Aadhar, will drive organizations to adopt sophisticated IAM solutions.

References

1. https://www.ftc.gov/system/files/documents/cases/1523054_uber_technologies_revised_analysis.pdf
2. <https://media.journoportfolio.com/users/75047/uploads/1703d740-d792-4a08-8ebb-227e10d3c1de.pdf>
3. <https://www.ftc.gov/news-events/news/press-releases/2017/01/uber-agrees-pay-20-million-settle-ftc-charges-it-recruited-prospective-drivers-exaggerated-earnings>
4. <https://core.ac.uk/download/pdf/291262002.pdf>
5. <https://apisecurity.io/issue-72-vulnerabilities-in-wordpress-themerex-addons-and-voatz-facebook-postmortem-jwt-talks-openapi-specification-3-0-3/>
6. <https://frichetten.com/blog/aws-api-enum-vuln/>
7. <https://www.upguard.com/blog/biggest-data-breaches-us>
8. <https://docs.aws.amazon.com/architecture-diagrams/latest/payment-system-interface-modernization/payment-system-interface-modernization.html>

9.

https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_attribute-based-access-control.html

10. **<https://aws.amazon.com/blogs/startups/how-setting-up-iam-users-and-iam-roles-can-help-keep-your-startup-secure/>**

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



CHAPTER 5

Secure Your Network

Introduction

Most attacks involve two common compromises: unauthorized access and network perimeter breaches. We covered unauthorized access in the previous chapter. In this chapter, we will discuss the network. In today's fast-paced digital era, robust network security has become paramount. As we explored in the introduction, the evolution from traditional parameterized security approaches to modern Zero Trust architectures reflects a significant shift in how organizations defend their networks.

According to the 2019 analysis of **Internet of Things (IoT)** device security^[1], an alarming 93% of routers possess HTTP administration interfaces on port 80. Some devices also support protocols such as **File Transfer Protocol (FTP)**, **Secure Shell (SSH)**, and Telnet that are not required for day-to-day use. Of the devices that support FTP and Telnet, 12% have weak FTP, and 1.6% have weak Telnet credentials. Certain providers have around 50% of routers with default credentials for FTP. Of routers that are externally exposed, 51.2% are vulnerable. This exposes people and organizations to significant vulnerabilities. Adopting a Zero Trust model is essential, ensuring that no user or device is trusted by default and that rigorous authentication measures are in place to protect sensitive data and maintain operational continuity.

Structure

The following topics are covered in the chapter:

- Case study: NotPetya
- Core concepts
- Secure network design and segmentation
- Secure communication
- Network operations
- Emerging technologies
- Guidelines and checklists

Objectives

This chapter will explore the *never trust, verify everything* architecture principle of Zero Trust networks. It will provide a comprehensive understanding of the criticality of network security and its myriad challenges. We will cover various concepts behind building a secure network architecture, securing communications, and adopting key principles around access controls, including API gateways and service meshes. Lastly, we will discuss protective monitoring, operational readiness, and upcoming trends and technologies impacting the space.

Case study: NotPetya

To better understand the impact, let us look at the following case study:

- **Date:** June 2017, **Countries:** Multiple
- **Summary:** On June 27, 2017, *Maersk*, the global shipping and logistics giant, fell victim to the NotPetya cyberattack. This malware, disguised as ransomware, rapidly spread through Maersk's international network, encrypting files and disabling thousands of computers worldwide. The attack crippled Maersk's operations, affecting its shipping, logistics, and port operations globally.^[2]
- **Impact:**

- 17 out of 76 Maersk-operated ports were disrupted.^[2]
- Estimated financial impact: \$250-300 million.^[2]
- Rebuilt the entire IT Infrastructure in 10 days, restoring 4,000 servers, 45,000 PCs, and 2,500 applications. Full recovery took two months.^[2]
- 49,000 laptops and computer devices were lost.^[3]
- 1,200 critical business applications became inaccessible.^[3]
- **Root cause(s):**
 - **Infected software:** The malware entered Maersk's system through a compromised update of *M.E. Doc*, a Ukrainian tax preparation software installed on a company computer in Odesa.
 - **Unpatched systems:** Maersk had vulnerabilities in its network, including outdated operating systems.
 - **Lack of network segmentation:** The malware spread beyond the Ukrainian operation to Maersk's global network due to insufficient network segmentation.
 - **Global interconnectivity:** Maersk's tightly coupled computer network allowed the malware to spread rapidly once it entered the system.
 - **Delayed security improvements:** Despite IT executives flagging issues with outdated systems and insufficient network segmentation, planned security redesigns were never executed.
- **Details:** Maersk is a Danish multinational shipping and logistics company with operations across 130 countries, 100,000 people worldwide, and annual revenues of \$ 39 billion. On June 27th, 2017, multiple Maersk staffers saw an unusual message on their machines: **Oops, your important files are encrypted** and demanded a payment of \$ 300 worth of bitcoin to decrypt. Within half an hour, the mysterious malware had spread across Maersk's network, and people ran down hallways. Even the keycard gates were locked, and the malware rendered desk phones useless. After two hours of panic, the entire Maersk global network was disconnected, and employees were told to turn off their computers and leave them at their desks.^[5]

It turns out this story of this massive attack started in the Ukrainian

capital of Kyiv, at the headquarters of *Linkos Group*. This family-run software business sells accounting software M.E.Doc. In the spring of 2017, Russian military hackers hijacked the company's update servers, pushing software updates and security patches to their customers to allow them a hidden back door.^[5]

In June, they used the piece of the back door to release a piece of malware called NotPetya (named after its resemblance to ransomware Petya). NotPetya comprised of two exploits working together: *EternalBlue* and *Mimikatz*. *EternalBlue* was a penetration tool created by the *National Security Agency* but leaked in another breach of the agency. It took advantage of a vulnerability in Windows to allow hackers free rein to run their code remotely. *Mimikatz* was a proof of concept by a French security researcher. It could pull passwords out of **random access memory (RAM)** and use them to hack into other machines on the network accessible with these credentials, allowing it to spread from machine to machine. Microsoft had released a patch for *EternalBlue* vulnerability before the launch of NotPetya, but systems that were not patched—as was the case with Maersk—could be infected by it.^[5]

M.E. Doc had been installed on a company computer in Odesa, a Ukrainian port city on the Black Sea. This was all NotPetya needed to infect the entire system. Across the globe, port facilities shut down, and tens of thousands of truckloads of goods were turned away. Maersk's entire booking system went down, and the complex loading systems used to load container ships to avoid capsizing them systematically. Maersk was dead in the water. At shipping terminals, angry cargo owners were calling; they had just heard from truck drivers that their vehicles were stuck outside Maersk's. The same scene was playing out at 17 of Maersk's 76 terminals.^[5]

A few days later, people were stationed in the Maidenhead recovery center to rebuild Maersk's network with piles of new laptops and prepaid Wi-Fi hotspots. However, when rebuilding, the staff realized that all of Maersk's servers' backups were from three to seven days before NotPetya's onset. No backup of the domain controls (that store data around user access) could be found. This was because the 150 or so

domain controllers were programmed to sync data with one another; every domain controller was wiped out simultaneously, and there were no offline backups.^[5]

Analysis of the servers showed that they had not been updated for at least four years, and security patches were non-existent. There was evidence of Russian presence in the servers, and several employees' accounts had been compromised. Intellect Services subsequently closed the back doors into the software, and state prosecutors promised the company would be held to account for the vast damage caused by their lax security procedures.^[5]

Miraculously, though, after an exhaustive search, one pristine backup was finally found in their Ghana office. By a stroke of luck, a blackout had knocked the server offline before the NotPetya attack, disconnecting it from the network. It contained a clean copy of the company's domain controller data, and its discovery was a great relief to the recovery team. To bring the clean copy, someone needed to grab the hard disk and travel to the UK from Ghana to help start the recovery.^[5]

The port operations resumed after a few days, but it took more than a week before terminals could function normally. Two weeks after the attack on the Maersk network, it was time to reach a point where new computers could be issued. It took about five months for a full recovery.^[5]

The malware cost Maersk \$250 million to \$300 million, but the broader cost of the disruption to the global supply chain is far more complex to measure. On top of that, Maersk was just one of the many victims. A \$400 million blow hit TNT Express (FedEx's EU subsidiary); others included chocolate-maker *Cadbury* and *Reckitt Benckiser*.^[5]

- **Lessons learned** ^[6]: The biggest lesson is that this could happen again and on a larger scale: global corporations are too interconnected, information security complex, and attack surfaces too broad to protect against state-trained hackers^[5]. There are a few basics that could have slowed the malware:

Lesson	Details
Supply chain	The origin of supply chain and automated updates was the third-party installation of accounting software. Before propagating such updates, setting

	up an automated process with the proper integrity and malware checks is critical.
Network segmentation	The rate at which the malware spread across the global infrastructure, including offices and ports, was mind-boggling. This begs the question of whether the global networks were sufficiently isolated.
Regular offline backups	Maersk's recovery would have been much slower and more complex without the lone backup in Ghana. Businesses should ensure that they have geographically dispersed, frequently updated offline backups and that these backups are regularly tested for integrity.
Updates and patching	Microsoft had released a fix for one of the vulnerabilities, but it appears this was not rolled out throughout the estate. Ensuring the application of critical security patches promptly is vital.
Incident response plans	A strong incident response plan saves time and resources. Maersk's ability to rapidly respond to the attack was key to limiting its impact. Having a clear, documented plan that is regularly rehearsed enables teams to act quickly and effectively in the event of a cyber incident.
Redundancy and multi-region	Redundancy in IT systems and multi-region resilience: The global nature of the business and decentralized IT infrastructure provided sufficient redundancy, and Maersk could pull resources from unaffected regions to assist in recovery. Infrastructure resilience can distinguish between a full-scale collapse and a manageable recovery.
Crisis communication	Crisis communication plans are essential. The ability to maintain communication during an attack or significant disruption can help avoid further chaos. Businesses must ensure that employees communicate effectively even when primary systems are down.

Table 5.1: Basics that slowed the malware

Let us look at some other notable examples:

The incidents in this section show weaknesses in the traditional perimeter approach to security, which we saw in [Chapter 1, Understanding Zero Trust](#). When combined with poor network segmentation, this approach can lead to catastrophic breaches that affect millions of users and cause billions in damages.

Case study: Target

The 2013 Target data breach ([Chapter 2, Architecture and Pillars of Zero Trust](#)) is a notable attack that exploited perimeter and network vulnerabilities. The attackers initially compromised Target's network through its HVAC vendor, which had access to Target's network. The attackers used

stolen credentials from the HVAC company to access Target's vendor portal. Using a vulnerability in the portal, they were able to gain remote access to Target's servers. The attackers exploited weak network segmentation and moved laterally into Target's POS systems, which should have been the most protected segment of the network. Proper network segmentation could have contained the breach and prevented the attackers from moving freely within Target's network.

Case study: Sony

In 2014, infiltrators broke into Sony Pictures' network security perimeter using *spearfishing* (emails containing false Apple ID verification links). This allowed them to gain Apple account passwords, which were used to test against Sony network accounts until the match was found. Once they had access, they could compromise confidential data due to a lack of data segmentation within Sony; a lot of data was stored on shared servers regardless of sensitivity.^[7]

Case study: TJX companies' breach

The TJX companies' data breach in 2007 was one of the most significant cybersecurity incidents in the history of the retail industry. 45 million customer records containing personal information (customer name, address, telephone numbers, credit card number and expiration dates, social security number, driver's license number) were stolen and sold on the black market. Millions of credit and debit cards were exposed, leading to fraudulent transactions. The attackers accessed TJX's network by exploiting weak Wi-Fi encryption through **Wired Equivalent Privacy (WEP)**, an encryption standard with known vulnerabilities. Once in the network, packet sniffing tools were used to steal sensitive unencrypted data such as credit card numbers, **Social Security Numbers (SSNs)**, etc. Several critical systems on the network were exploited due to weak network security controls. The exfiltration of sensitive customer data continued for over 18 months.^[8]

Core concepts

As you can see, networks are the backbone of modern communication and

data exchange across users, devices, and systems; hence, they are critical to security. However, how did it come to be? The origin of computer networks dates back to the 1960s, when the primary network was the telephone network, with lines buzzing to help with communications during the Cold War period. U.S. **Advanced Research Projects Agency Network (ARPANET)** was the first packet-switched network, first used in 1969 and decommissioned in 1989; the main limitation was that it could not support simultaneous communications between multiple connected systems. In the 1970s, Ethernet, which was a wired computer network technology, was introduced and standardized as **Institute of Electrical and Electronics Engineers (IEEE) 802.3**. Initially using 10Base5 thick cables, this was the first actual adoption of a connected network that further evolved as the technology improved further and with higher bandwidths. In 1983, the **transmission control protocol/internet protocol (TCP/IP)**, the real beginning of the modern Internet, was introduced. ^[9] A logical extension to this was the invention of wireless networking in 1997 when the IEEE developed the 802.11 standard for Wi-Fi technology. Before reviewing the details of securing networks, let us dive into some networking basics.

A network is a group of interconnected devices allowing them to share resources and data. This now includes laptops, computers, mobile devices, printers, file-sharing systems, or any connected and reachable device on the network. There are a few different types of networks that are talked about and listed as follows:

- **Local area network (LAN)** is for small geographic areas like homes or offices.
- **Wide area network (WAN)** covers more extensive areas, connecting cities or countries. The largest WAN is the Internet, which connects many international networks, allowing people to browse information and share data.
- **Virtual private network (VPN)** is a secure connection over public networks that helps protect your privacy online. VPNs provide an encrypted tunnel for sending your data to protect your online identity, especially on public Wi-Fi.

Networks rely on a few standard sets of network devices listed as follows:

- **Switches** connect devices within a network and manage data flow,

typically for wired connections.

- **Routers** typically connect multiple networks and direct traffic efficiently. Your internet provider usually offers a router that connects your home network to the WAN.
- **Firewalls** protect networks by filtering traffic based on pre-defined rules and policies.

The best way to understand networking is the **Open Systems Interconnection (OSI)** ^[10] model developed in the late 1970s and designed to allow standalone systems to communicate. It comprises of seven layers defining how data is transmitted across networks, starting with the physical layer for hardware connections (e.g., Ethernet); the data link layer for connecting two systems on top of the physical layer; the network layer concerned with routing, forwarding and addressing; transport layer ensures packets of data arrive in the correct order without losses or errors; session layer is responsible for managing sessions across applications; presentation layer is focused on syntax of the data; and application layer is the standard communication between applications.

Data transmission across these layers typically uses small, formatted units of data transmitted over networks called **packets**. Packets are the fundamental building blocks of digital communication, enabling efficient and reliable data transfer across the Internet and other computer networks. They help in efficient data transmission, error handling, flexible routing, scalability, and interoperability between diverse systems.

In addition to OSI, an alternative model commonly used is TCP/IP. It comprises five layers and is more widely used in practice. [Table 5.2](#) shows the TCP/IP protocol layers, their purpose, and the list of associated protocols:^[11]

Layer	Description	Protocols
Application	Interacts directly with software applications, prepares data for transmission, and provides user services.	DNS, HTTP, FTP
Transport	Responsible for end-to-end communication, breaks data into packets, and ensures reliable delivery.	TCP, user datagram protocol (UDP)
Network	Handles routing of packets between networks using IP addresses.	IP (IPv4, IPv6), ICMP, ARP

Data link	Organizes bit streams into frames and delivers them to adjacent systems.	Ethernet, PPP, IEEE 802.2
Physical	Deals with the physical transmission of data over network media.	Ethernet (IEEE 802.3), Wi-Fi (IEEE 802.11), Bluetooth

Table 5.2: TCP/IP layers

Most network security controls and strategies are typically applied to the network and transport layers. This chapter will also cover certain application layer security concepts.

Most networking today is **software-defined networking (SDN)**; the network of devices is all connected in an innovative network architecture, followed by the separation of the control plane (setup, configuration, and management of the network) from the data plane (the use of the network and flow of data packets). SDNs enable centralized network management and programmability, allowing network administrators to dynamically adjust network-wide traffic flow and respond quickly to changing needs through a software-based controller. These policies can be based on multiple factors, such as identity, application, or device posture, instead of traditional methods, such as **access control lists (ACLs)** and firewall rules, allowing better micro-segmented networks. This approach offers numerous benefits, including improved network performance, enhanced security, and cost savings. A further extension of SDNs is the **software-defined wide area networking (SD-WAN)**, which is focused on optimizing traffic across the WAN and providing a dynamic policy-based application path selection across multiple WAN connections. SD-WAN enhances Zero Trust by enforcing traffic segmentation and application-aware routing across distributed networks

Now that you have a quick background in networking, let us understand how this applies to Zero Trust architectures today. A secure information technology estate combines a robust network infrastructure design and business, applications, and operational processes developed with a Zero Trust approach; the lack of any of these leads to vulnerabilities that hackers can exploit. *Figure 5.1* explains the core concepts for achieving this:

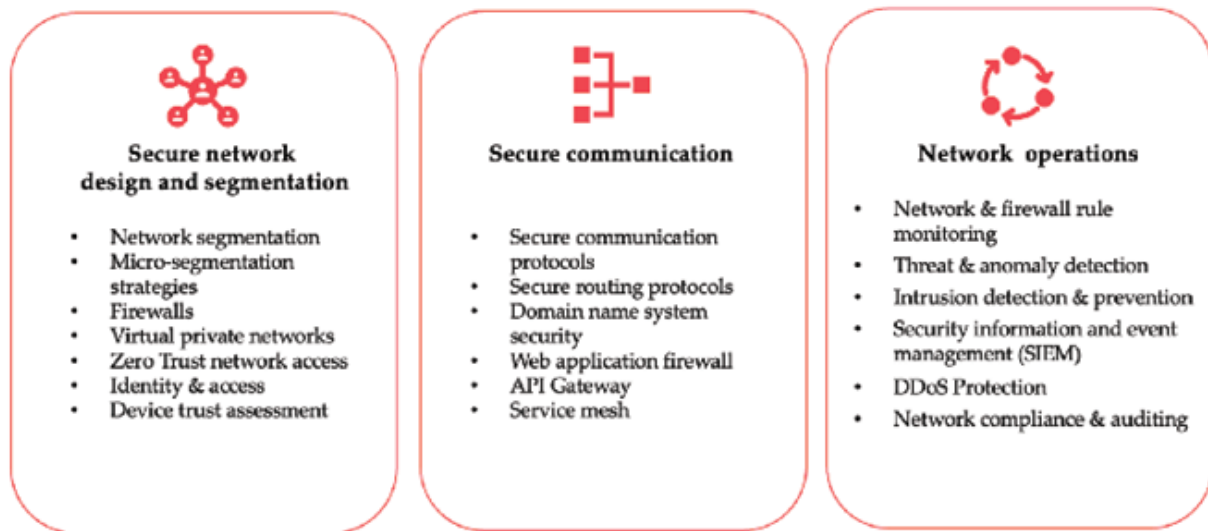


Figure 5.1: Core concepts

- **Secure network design and segmentation** focuses on the design and architecture of an enterprise's network, using a layer security approach to minimize attack surfaces, prevent unauthorized access, and ensure data confidentiality, integrity, and availability.
- **Secure communication** covers techniques for securing communication across multiple endpoints, including servers, devices, services, and APIs, both within and outside the enterprise's network.
- **Network operations** cover operational aspects such as network monitoring, continuous threat, anomaly detection, key processes such as change management, intrusion detection and prevention, and **security information and event management (SIEM)**.

We will look at each of them in detail in the next sections.

Secure network design and segmentation

In [Chapter 1, Understanding Zero Trust](#), we saw that traditional network architectures focus on securing the perimeter network alone. This alone is insufficient in today's world, with a complex web of networks and connected devices, users, and systems. Instead, a radical approach to segmenting networks is needed when designing them.

Network segmentation and isolation

Most of the case studies discuss the idea of network segmentation, but what is segmentation? Network segmentation splits an organization's network into smaller sub-networks or segments. Each network segment is a self-contained security zone where systems within it share the same level of security, access policies, and trust level. Segmentation allows network teams to compartmentalize networks into different zones for two main reasons: firstly, so they can apply varying controls based on the criticality of the assets in each zone, and secondly, so an attack on one does not compromise the entire network. Suppose we were to use the analogy of moats, drawbridges, and multiple layers of defense around a castle (in [Chapter 2, Architecture and Pillars of Zero Trust](#)). In that case, it is around creating multiple such protections to segregate different areas of the castle being protected, either making it impossible to penetrate the entire castle or slowing down the attack.

Splitting the network physically and logically can stop communications between segments by default. Communication between segments is only allowed explicitly for specific and defined cases. A well-segmented network could have curtailed most attacks, if not all, including the NotPetya one with Maersk.

Real-world relevance

[Figure 5.2](#) shows a sample implementation of network segmentation in an enterprise. Although this network shows a **demilitarized zone (DMZ)** and perimeter firewalls are part of traditional network security, it is crucial to understand that the security of systems, applications, and users does not rely on their location. We would apply a similar set of controls, such as authentication, access control, data security, etc. (as covered by the various pillars of Zero Trust in this book). However, Zero Trust network policies may be configured to allow or deny access to critical resources from specific networks (in that sense, the network location is an extra attribute within the policy).

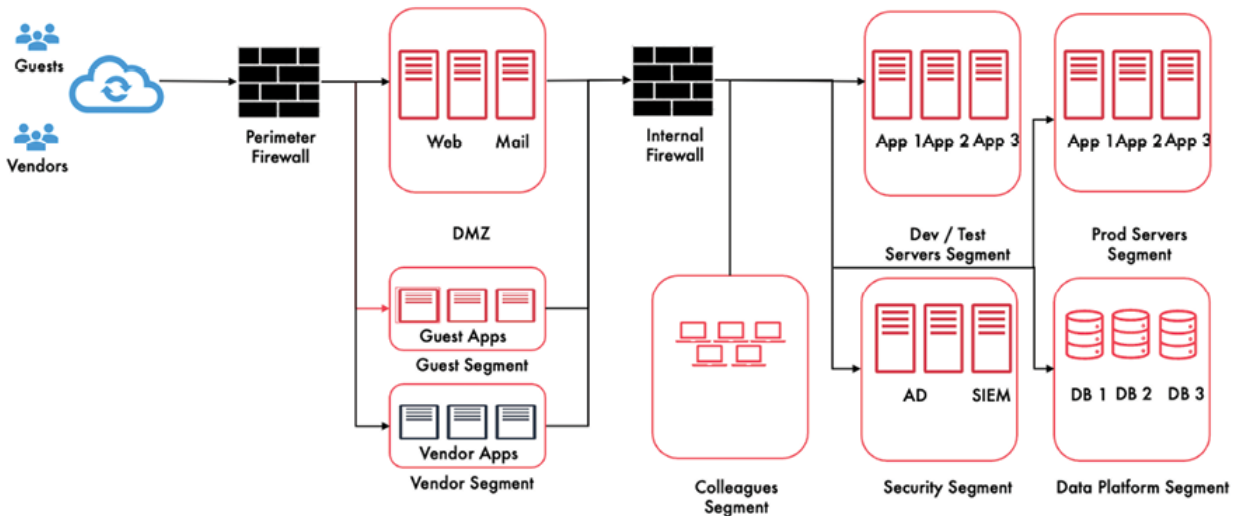


Figure 5.2: Network segmentation and isolation

The following are the main segments demonstrated in the figure and their purpose:

- The DMZ is a group of network segments between the perimeter and internal firewall. The perimeter firewall provides the first layer of defense to the enterprise network from the Internet. Internet-facing servers, such as web and mail servers, are part of the DMZ.
- Separate network segments are created for guest users who require temporary access to a guest segment in a dedicated virtual LAN. Access is limited to specific time periods, and the credentials automatically expire.
- Vendors typically require access to specific systems/applications and should not have access to the rest of the enterprise network; they should be in a separate vendor segment. Restrict their access using identity-based authentication, and access should be continuously monitored using SIEM.
- The internal firewall controls traffic between these network segments and ensures that only specific network segments can communicate with each other. All critical servers hosting applications, data, and crown jewel services are hosted behind the internal firewall.
- Corporate users or employees are placed in a dedicated network segment (colleagues segment), which separates them from guests and vendors.

- Dev or test environments (dev or test servers segment) are separated from production environments (prod servers segment) to ensure no unintended network connectivity between the two segments.
- The data platform is an enterprise's most critical asset, and it is secured by placing data services in a dedicated network segment (data platform segment).
- The security segment hosts security and identity services such as **Active Directory (AD)**, vulnerability management, SIEM, and threat detection systems.

Most segmentation uses a combination of **virtual local area networks (VLAN)** using products such as Juniper Networks and Cisco Catalyst, SDN, and cloud-native micro-segmentation using network security groups (SGs or NSGs) and firewalls.

Micro-segmentation strategies

While network segmentation and establishing trust boundaries across network segments prevent unauthorized access to a network segment, an attacker can still move between workloads deployed within the same network segment. Micro-segmentation overcomes this limitation by defining fine-grained application-level segments and access control based on identity, policies, and real-time context, restricting lateral movement between workloads. A typical example of micro-segmentation would be to adopt security policies set up on a group of applications and database servers, only to allow explicit communication between them. *Figure 5.3* shows the differences between typical network segmentation and micro-segmentation requirements:

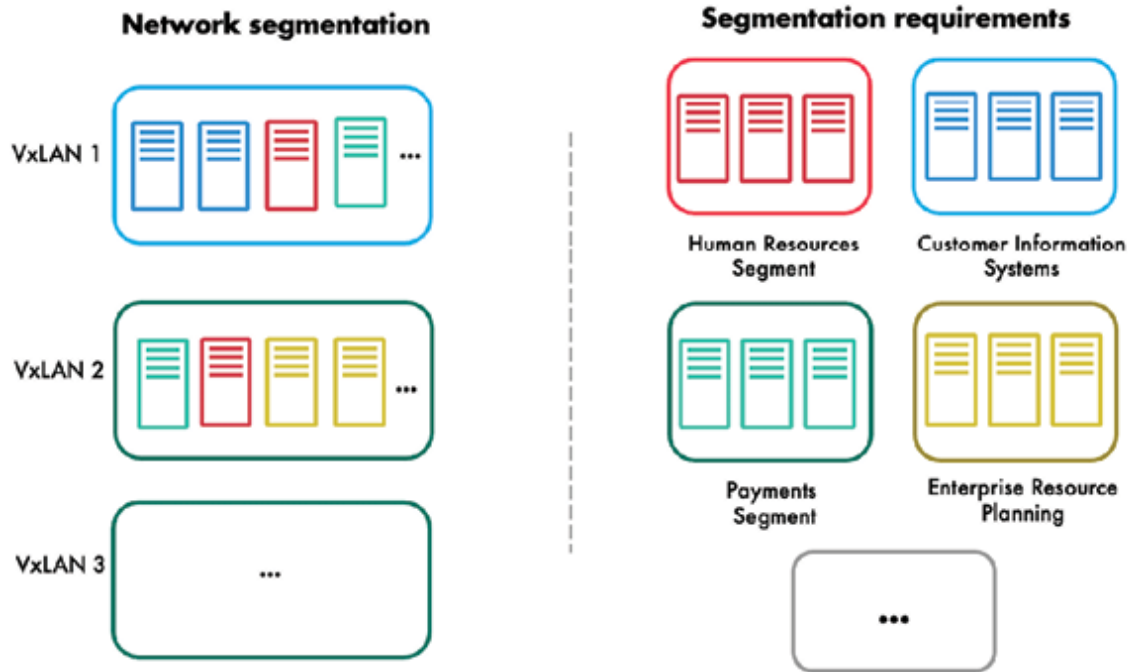


Figure 5.3: Micro-segmentation requirements^[12]

Real-world relevance

In the TJX data breach, access to multiple critical workloads could have been avoided with an appropriate micro-segmentation strategy. In the case of Maersk, a combination of network segmentation, i.e., to have Active Directory domain controllers and application servers in separate VLANs, as well as a micro-segmentation strategy only to allow explicit communication between these groups of workloads, could have prevented the spread of NotPetya significantly. Micro-segmentation is specifically helpful in containing ransomware attacks. Some examples of micro-segmentation include creating dedicated **virtual networks (VNet)** and **virtual private cloud (VPC)** for application workloads in cloud-native environments. This can be further extended to Kubernetes (container-based) applications through the adoption of network policies (we will cover in [Chapter 9, Container Security](#)) and service mesh (discussed later section in this chapter).

Firewalls

[Figure 5.1](#) also shows two firewalls (perimeter and internal). A firewall is a security device or software that monitors, filters, and controls incoming and

outgoing network traffic based on predefined security rules. Firewalls enforce policies that allow or deny traffic based on IP addresses, ports, protocols, and applications, ensuring that only legitimate communication flows within an enterprise network. In modern IT environments, firewalls play a vital role in network segmentation, access control, and intrusion prevention, forming the foundation of enterprise security architectures.

Firewalls have evolved significantly from their traditional role as perimeter-based gatekeepers to becoming dynamic, identity-aware security enforcers in modern **Zero Trust architectures (ZTA)**. In the past, firewalls primarily focused on blocking or allowing traffic based on IP addresses and ports, assuming that internal networks were inherently trustworthy. However, this perimeter-based model became obsolete as enterprises expanded into cloud environments, remote work, and third-party integrations. The rise of advanced persistent threats, insider threats, and lateral movement attacks has forced organizations to rethink security, shifting from a castle-and-moat approach to granular, policy-driven protection across all network layers. The following are the types of firewalls available today:

- Packet-filtering firewalls filter traffic based on IP addresses, ports, and protocols. Examples are Cisco ACLs and AWS Network ACLs.
- Stateful inspection firewalls track the state of the active connections and are more advanced than packet filtering as they match the responses to requests.
- Proxy or application-layer firewalls act as intermediaries between users and the Internet and perform deep packet inspection and content filtering.
- **Next-generation firewalls (NGFWs)** operate at multiple layers, including stateful inspection, deep packet inspection, intrusion prevention, and application awareness. They can detect and block advanced threats such as malware and zero-day attacks.
- Web application firewalls protect web applications from common attacks, and we will cover them later in the chapter.

Real-world relevance

Firewalls are no longer passive gatekeepers but are integral to identity-aware security frameworks. Unlike traditional firewalls that rely on network zones,

Zero Trust firewalls assume that every internal or external request must be verified. [Table 5.3](#) shows Zero Trust sample firewall policies that show how modern firewalls can be used to manage flows. The first policy states that all guest access is denied to the security segment services or apps, the second policy allows vendor traffic to specific prod applications, and the third and fourth control access from the segment running production services.

Source zone	Source identity or device context	Destination zone	Allowed traffic	Port/Protocol	Action
Guest segment	Any	Security segment (AD, SIEM)	None	Any	Deny
Vendor segment	Authenticated vendor (MFA, approved device)	Prod servers segment (App1, App3)	API access only	443 / HTTPS	Allow
Prod servers segment	Server-to-server Authentication	Security segment (AD, SIEM)	Authentication only	389 / LDAP 636 / LDAPS	Allow
Prod servers segment	Server-to-server Communication	Data segment	Database access	1433 / TCP 1521 / TCP	Allow

Table 5.3: Zero Trust firewall policy examples^[13]

Modern NGFWs also integrate with **identity and access management (IAM)** systems, forcing security policies based on user identity, device posture, and contextual risk assessment rather than static IP rules. For instance, Palo Alto NGFW supports authentication policies that enable you to authenticate users before they can access specific services and applications.

Virtual private networks

VPN is a technology that establishes a secure and encrypted network connection over unsecured networks such as the Internet. In a typical enterprise network setup, a VPN enables secure communication across the following:

- **Remote user connectivity via a point-to-site VPN setup:** This setup

establishes an encrypted network connection between the remote user's device and the enterprise network, ensuring that only users authorized to use the VPN service can connect to the enterprise network.

- **Branch connectivity via a site-to-site VPN setup:** This setup establishes a secure connectivity tunnel between two sites of an enterprise, allowing workloads within specified VLANs or subnets in one site to talk to workloads on the specified VLANs on the other site of an enterprise.
- **On-premises to cloud connectivity:** For a hybrid environment setup, enterprises often need to establish connectivity between an on-premises network and the cloud. This is typically achieved by establishing a site-to-site VPN tunnel between on-premises and cloud networks.

It is important to note that VPNs primarily operate at a network layer (layer 3 of the OSI model), establishing secure tunnels but do not inherently enforce granular application-level access controls. A user uses a VPN client software to establish a point-to-site VPN connection. The client software stores the VPN key needed for the connection to complete. VPN keys are typically stored in the VPN client's configuration on the end user's system. If a hacker obtains access to an end-user system, they can initiate a VPN connection using the VPN client. It, therefore, becomes imperative to add additional layers of security over and above VPN, typically by enforcing the VPN client to authenticate the user's identity against the enterprise's **identity and access management (IAM)** system.

Real-world relevance

Figure 5.4 shows point-to-site and site-to-site VPN tunnels on the Azure cloud. The Azure and enterprise on-premise networks are connected via a site-to-site VPN tunnel. A remote user of the enterprise connects to the Azure network via a point-to-site VPN tunnel. The on-premise radius server is integrated with the enterprise Active Directory server and adds a layer of authentication when the remote user initiates the point-to-site connection.

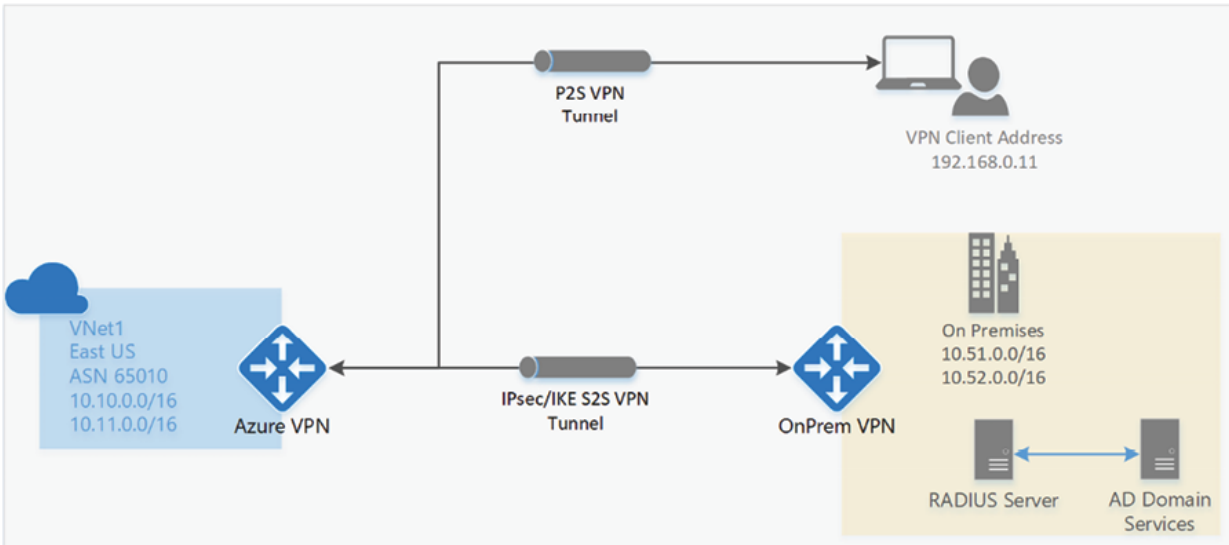


Figure 5.4: VPN tunnels ^[14]

Zero Trust Network Access

Traditional VPNs are limited because they allow the remote user to establish the VPN connection and grant broad access to the network once users are authenticated. This can increase the likelihood of unauthorized access to workloads and potential breaches.

Unlike traditional perimeter defenses, **Zero Trust Network Access (ZTNA)** builds on the *never trust, always verify* principle of the Zero Trust architecture. The ZTNA security model provides access to approved applications/workloads instead of everything in the virtual LAN or subnet, with strict identity verification and least privilege principles. In addition, ZTNA authenticates the user against the enterprise IDAM system and continuously monitors device security posture and behavior. The key differences from traditional technologies are listed as follows:

- **Trust model:** Traditional models rely on implicit trust for internal users or devices, whereas ZTNA continuously verifies every access request.
- **Access scope:** VPNs provide broad network access once authenticated, while ZTNA grants granular, application-specific access.
- **Network architecture:** Legacy approaches focus on securing a fixed perimeter; ZTNA operates in decentralized environments without relying on a defined perimeter.
- **Attack surface:** ZTNA minimizes exposure by hiding applications from

unauthorized users (e.g., through outbound-only connections), unlike VPNs, which expose IP addresses to the Internet.

In the case of Target's data breach, access to point of sale systems could have been secured further by implementing ZTNA, ensuring only authorized devices and users have access to these systems.

Real-world relevance

Some of the current day examples of solutions that implement ZTNA are as follows:

- **Zscaler Private Access (ZPA):** It provides direct user-to-application connections without exposing the network and enforces least privilege access, reducing the risk of lateral movement. It is ideal for replacing legacy VPNs in hybrid work environments.
- **Palo Alto Networks Prisma Access:** It integrates ZTNA with **secure web gateways (SWG)** and cloud-delivered firewalls for comprehensive protection. It offers continuous monitoring of user behavior and device health.
- **Cloudflare Zero Trust:** Combines identity-aware policies with application-level segmentation, preventing shadow IT by controlling unsanctioned app usage.
- **Cisco Duo:** When implemented with ZTNA, it uses MFA and device trust assessments to enforce granular access controls.

Identity and access

Chapter 4, Secure Your Identities, covered the importance and techniques behind identity and access management. These techniques complement strong network access controls, mitigate data breaches and insider threat risks, and limit lateral movement during potential breaches. The following are examples of how identity and access can be integrated with network controls:

- Use identity as a perimeter and identity-aware firewalls instead of traditional firewalls that relied on IP addresses to control or filter traffic. Modern identity-aware firewalls integrate with identity systems such as Active Directory, map users' identities to network traffic and help you

define policies and granular access controls. Solutions like Juniper SRX series firewalls and **Secure Access Service Edge (SASE)** allow identity-aware controls and policies.

- **Just-in-time (JIT)** access minimizes exposure to sensitive resources by granting temporary permissions only when needed. Instead of providing users with standing privileges that could be exploited if compromised, JIT ensures that access is provisioned dynamically for a limited time based on specific tasks or requests. This reduces the attack surface and helps prevent misuse of credentials. JIT solutions such as Prisma Cloud and CloudEagle allow controlling access to applications and services for a limited time. In addition, cloud providers like Azure also allow JIT network access to virtual servers; this eliminates the need to enable rules in firewalls that leave your virtual machines vulnerable to attack. JIT lets you allow access to your VMs only when the access is needed, on the ports needed, and for the period required.
- Privileged access management focuses on securing administrative accounts and privileges, which attackers often target. These tools enforce strict controls over privileged accounts by including approval workflows, additional monitoring, audit, recording, and stricter password rotation policies. JumpCloud, CyberArk, BeyondTrust, and Azure PIM are popular solutions for privileged access.
- MFA adds a layer of security by requiring users to verify their identity through multiple factors, something they know (password), something they have (security token), or something they are (biometric). MFA significantly reduces the risk of unauthorized access, even if credentials are stolen or guessed. Through better integration with the network access controls, modern implementations support adaptive MFA, which adjusts authentication requirements based on contextual factors like user location or device type. Some examples of MFA are *Okta*, *Google*, and *Microsoft Authenticator*.

Device trust assessment

Device trust verifies the security and compliance of devices connected to an organization's network. The process involves verifying the identity and health of devices, whether managed (corporate) or unmanaged (personal),

before granting access. As part of the assessment, key factors, including secure boot, device attestation, operating system versions, patch levels, installed security software, posture validation, and compliance with organizational policies, are evaluated before a device is granted access to systems and services.

Mobile device management (MDM) systems or endpoint security solutions often enforce device trust by maintaining an inventory of trusted devices and monitoring their security posture in real-time. For example, solutions like *Cisco Duo* and *Microsoft Intune* integrate device trust into broader Zero Trust architectures to enable granular access control based on device compliance. Device trust ensures that only secure and compliant devices gain access to corporate resources, forming a critical layer of defense in Zero Trust architectures. It protects against threats like malware infections or data breaches caused by compromised endpoints and reduces risk of compromised or non-compliant devices, ensuring only trusted endpoints can interact with sensitive data and systems.

Real-world relevance

Figure 5.5 shows an example of a conditional access policy within Microsoft Entra. This policy works in conjunction with Microsoft Intune device registration and compliance, allowing access to specific target resources, applications, and services based on the user's risk level, device platform, and compliance status.

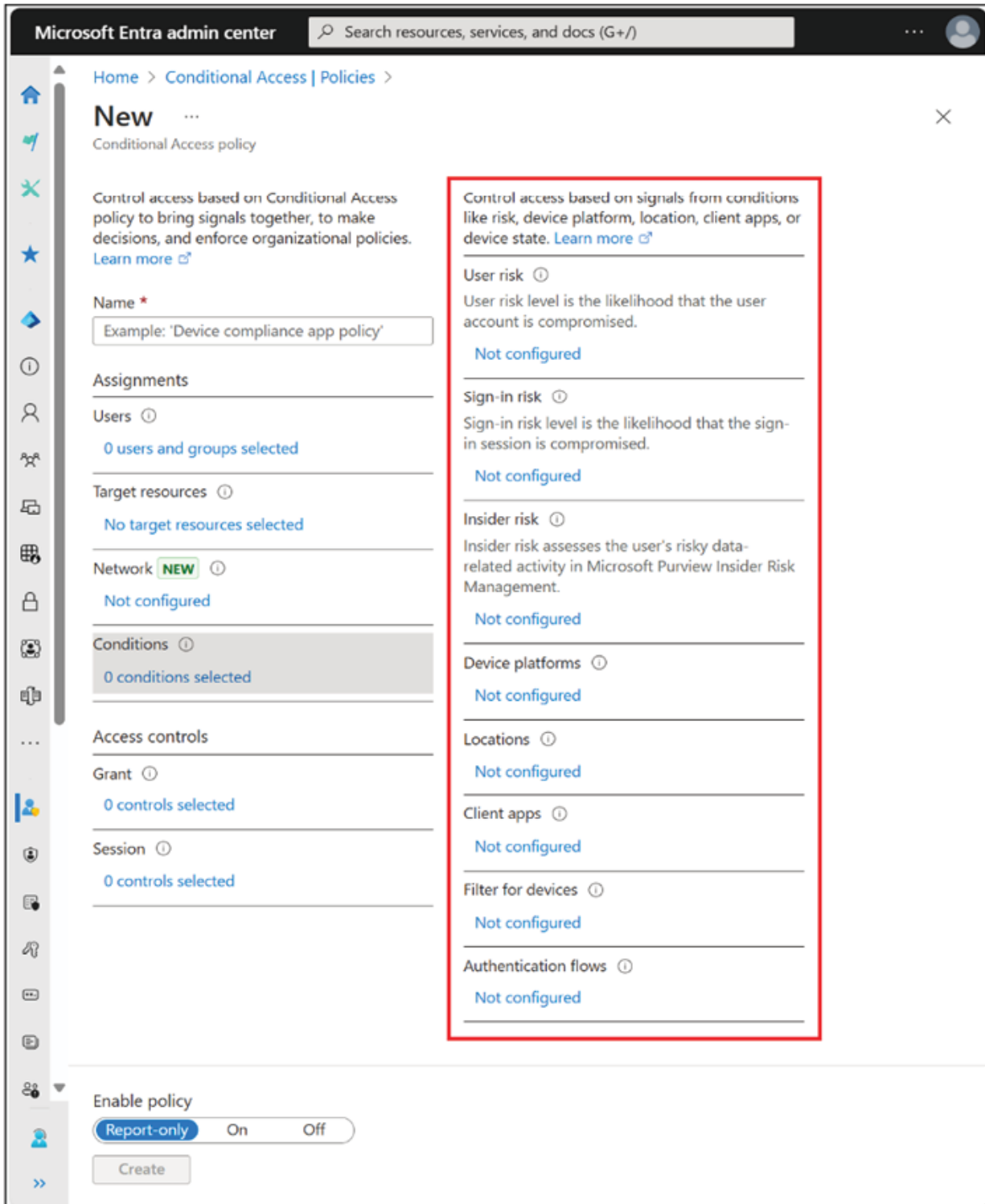


Figure 5.5: Microsoft Entra conditional access policies ^[15]

Secure communication

Beyond securing an organization's network infrastructure, another aspect of network security is securing communications across entities. Most organizations host internal services and consume many external services; the boundaries are also blurred when adopting cloud-based infrastructure. However, the core concept is the security of how users and systems talk to each other. *Figure 5.6* summarizes an organization's network and communication across internal and external applications:

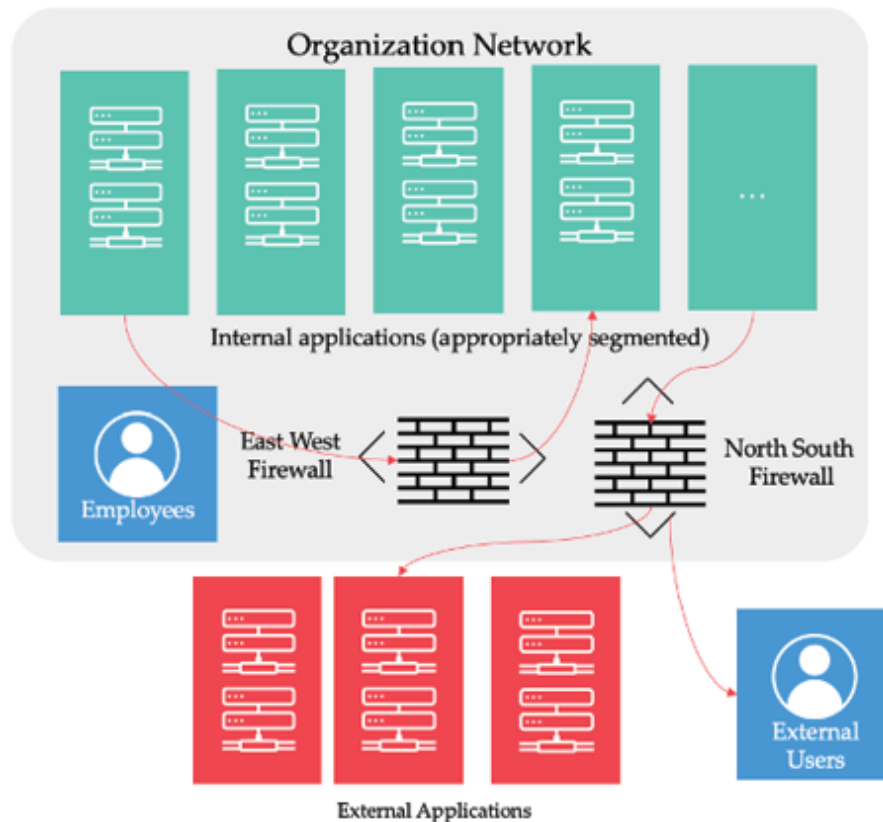


Figure 5.6: Network communications in an organization

North-south network communication references all traffic inside an organization's network boundary with the outside world or the internet. This traffic typically involves the following key use cases and managed using north-south firewall:

- **Internet access and email:** General internet traffic that internal users consume external websites and productivity tools or send and receive emails.
- **Customer-facing public services:** These are services hosted by

organizations and offered as public services available over the internet.

- **Cloud services and third-party applications:** An organization consumes third-party or cloud provider services.
- **Remote access and VPN:** Any remote users who connect to the organization's network from home or use virtual private network connectivity across offices and site locations. This has become a norm and the most common form of traffic flow.
- **Integrations and file transfers:** These are the data exchanges across systems or through **application programming interfaces (APIs)**.

East-west network communication refers to the internal traffic within an organization's network boundary. This involves traffic that originates, traverses, and terminates inside the organization between its applications, devices, and users. Traditional security did not emphasize this type of communication enough and was typically open and considered trusted, but as the case studies have demonstrated, this was a flawed assumption. East-west communication is of various types between users, across services or applications, and, lastly, users accessing deployed services. They all require the same level of rigor and security controls. Let us now dive into various types of security controls to secure communications.

Modern-day systems are mostly microservices and API based; this means a system can be comprised of hundreds of smaller components deployed together to offer a set of end-user functions. Many of these microservices also interact with external third-party services. As such, systems today depend on many back-and-forth communications across these components or micro-services, all of which happen over the network, internal or external. Securing such applications and workloads is complex but critical to modern network security in cloud-native and hybrid environments. Unlike traditional network security, which focuses on protecting the perimeter, workload security emphasizes safeguarding the components that run applications: virtual machines, containers, and serverless functions. Workload security ensures these components are protected from malware, misconfigurations, unauthorized access, and zero-day vulnerabilities.

Figure 5.7 shows the key network-level mechanisms available today to safeguard workloads that we will dive into in detail:

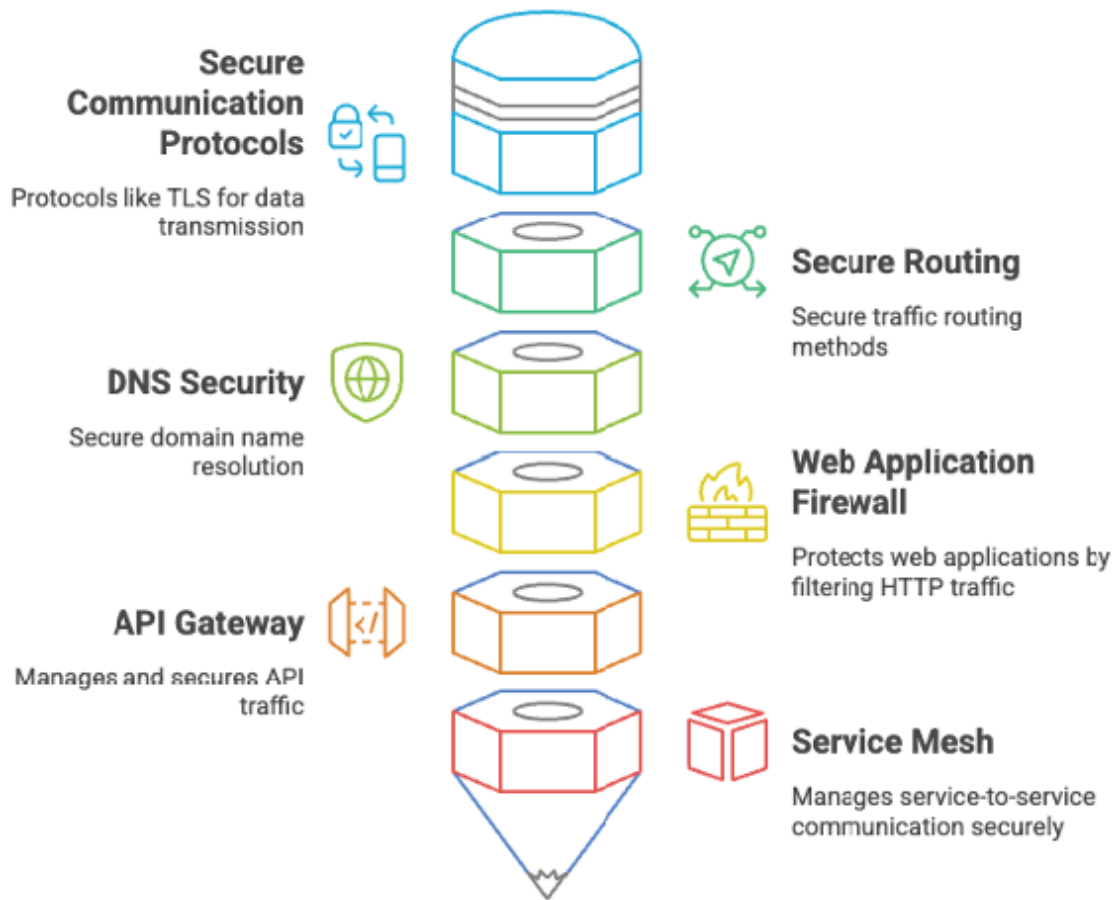


Figure 5.7: Components of secure communication

Table 5.4 summarizes the network controls we will cover here and the types of communications (as depicted in *Figure 5.7*) they are used to secure:

Network control	Description	Used to control
Secure communication protocols	Communication protocols used to transmit data across networks such as transport level security (TLS)	All communications related to domain resolution internal and external
Secure routing	Security way of routing traffic across internal and internal flows	All communications related to domain resolution internal and external
DNS security	Secure domain name resolution protocols	All communications related to domain resolution internal and external
Web application firewall	Protects web applications filtering and monitoring HTTP traffic	• North south user to application • North south external application to internal application
API gateway	Manages and secures API traffic between clients and services	All application to application north south and east-west

Network control	Description	Used to control
Service mesh	Dedicated security to manage service-to-service communication	East-west application to application and internal application communication (across microservices)

Table 5.4: Secure communication controls mapping

Secure communication protocols

Communication protocols are protocols used to transmit data across networks, and ensuring the confidentiality, integrity, and authenticity of these communication protocols is essential to protect data movement. Some protocols covered here will be discussed further in depth in *Chapter 6, Secure Your Data*, as it requires some fundamental understanding of encryption and cryptography addressed in that chapter. However, it is helpful to have a brief overview of the secure protocols related to networking. These protocols operate at different layers of the OSI model and TCP/IP model we discussed earlier and are tailored for specific use cases, such as web browsing, file transfers, or network-level communication. In this subsection, we discuss the overview of key secure communication protocols at the most common layers of the model network and application levels:

- **Network layer:** It is the layer 3 of the OSI model which sends data packets and is made of Internet Protocol. **Internet Protocol Security (IPSec)** ^[16] is this layer's most common set of security protocols. IPsec secures communications at the network layer by encrypting and authenticating IP packets. IPsec supports various types of encryptions, including the **Advanced Encryption Standard (AES)**, and is commonly used for VPNs to establish secure tunnels over public networks. It includes multiple protocols:
 - **Authentication header (AH)** for integrity checks using headers.
 - **Encapsulating security payload (ESP)** for encrypting entire IP packets or just the payload.
 - **Internet key exchange (IKE)** negotiates cryptographic keys for secure communication between two devices; the device negotiates encryption keys and algorithms to transmit and receive data.
- **Application layer:** It is layer 7 of the OSI model and is the top layer

and provides an interface for applications to access network services and data. Some of the examples of protocols in this layer include HTTP, FTP, and DNS. There are multiple security protocols in use in this layer; we will cover details of these further in [Chapter 6, Secure Your Data](#) after covering some basics on cryptography and encryption. The following is a brief summary of these protocols:

- **Secure sockets layer (SSL)** was the predecessor of TLS and provided encryption for secure network communication. While replaced by TLS due to vulnerabilities in older SSL versions, the term *SSL* is often used interchangeably with TLS in practice. Some legacy systems still rely on SSL for encrypted communication.
- **Transport layer security (TLS)** is widely used to secure web communications by encrypting data between a client and server. It ensures confidentiality, integrity, and authentication through cryptographic algorithms and digital certificates. An example is securing HTTPS connections for websites.
- **Mutual TLS (mTLS)** is an extension of TLS that provides mutual authentication by requiring both the client and server to present digital certificates. It is ideal for secure APIs, inter-service communication in microservices architectures, and environments handling sensitive data like healthcare or financial systems.
- **Hypertext Transfer Protocol Secure (HTTPS)** combines HTTP with TLS to secure web traffic. It protects against eavesdropping, man-in-the-middle attacks, and data tampering during web browsing.
- **Secure Shell (SSH)** is a cryptographic protocol that provides secure remote access to systems over an unsecured network. It encrypts communication between a client and server using public key cryptography, ensuring confidentiality and integrity, commonly used for remote administration, file transfers, and tunneling other protocols securely.
- **Secure File Transfer Protocol (SFTP)** operates over SSH to securely transfer files between a client and server. It encrypts authentication credentials and file data during transmission, making it ideal for sensitive file transfers.

Table 5.5 summarizes these secure communication protocols with example use cases:

Protocol	Layer	Details
IPSec	Network	Encrypt IP packets, VPNs, site-to-site tunnels
AH	Network	Authentication Header for integrity checks using headers.
ESP	Network	Encapsulating security payload for encrypting entire IP packets or just payload
IKE	Network	Key exchange for IPsec, Dynamic VPN setup
SSL	Application	Legacy encryption protocol, Legacy systems requiring encrypted communication
TLS	Application	Transport layer security encrypts web communications. HTTPS and email security.
mTLS	Application	Mutual transport layer security for mutual authentication. APIs and microservice
HTTPS	Application	Secure hypertext transmission protocol, secure web traffic
SFTP	Application	Secure file transfers
SSH	Application	Secure remote access and tunneling, server management, encrypted file transfers

Table 5.5: Secure communication protocols

Secure routing protocols

As the name suggests, routing protocols specify how network routers route the traffic and are essential for determining the best paths for data to travel across networks. Traditional routing protocols like **Border Gateway Protocol (BGP)** ^[17] and **Open Shortest Path First (OSPF)** lack inherent security measures, making them vulnerable to attacks such as route hijacking, spoofing, and DoS attacks we covered in *Chapter 1, Understanding Zero Trust*. Securing these protocols is critical to ensuring the integrity and reliability of network communication.

BGP is the backbone of internet routing, facilitating communication between **autonomous systems (AS)**. However, the lack of built-in security features exposes it to risks like denial of service, route manipulation, and route hijacking, where malicious actors advertise incorrect routes. To counter

these threats, there are a few options available:

- **Access control lists** can be enabled so that only legitimate BGP traffic from authorized BGP neighbors is permitted.
- **Neighbor authentication**, where the neighbors prove they are legitimate and trusted, verifies communications between and ensures routes are only learned from known neighbors using **message-digest algorithm (MD5)** authentication, BGPsec, or **resource public key infrastructure (RPKI)**.
- **Route analytics, logging, and monitoring tools** monitor BGP routes in real time, detect anomalies such as route leaks or hijacks, and alert operators immediately.

OSPF is a widely used interior gateway protocol for routing within an organization. While OSPF is efficient, it can be exploited if not appropriately secured. Some of the enhancements it supports for better protection are MD5 authentication to secure routing updates between peers by adding a cryptographic hash to each message and encrypted sessions using TLS and SSL.

Real-world relevance

By implementing these secure routing mechanisms, organizations can prevent malicious actors from disrupting their networks or redirecting traffic for nefarious purposes. However, the specifics depend on the technologies in use; most cloud providers support elements of BGPsec, such as MD5 authentication and tracking monitoring. Cloud providers are working on helping secure routing by adopting similar technologies. ^[18]

Domain name system security

DNS is a crucial element of every organization's network infrastructure, converting human-readable domain names into IP addresses. However, DNS was developed in the early days of the Internet and is inherently vulnerable to various attacks, including DNS spoofing, cache poisoning (locally cached IP addresses), tunneling, hijacking, and DDoS. Therefore, securing DNS is essential for maintaining trust in online communications. The following are some standard techniques for enhancing DNS security:

- **Domain Name System Security Extensions (DNSSEC):** DNSSEC

adds cryptographic data authentication, ensuring their authenticity and integrity. It prevents attackers from spoofing DNS responses or redirecting users to malicious websites.

- **DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT):** These protocols encrypt DNS queries and responses, protecting them from interception by attackers. DoH sends DNS traffic over HTTPS connections, while DoT uses TLS encryption. TLS and HTTPS are covered further in detail in *Chapter 6, Secure Your Data*.
- **Response rate limiting (RRL):** It is a security feature that mitigates DDoS attacks by limiting the rate at which a server responds to repeated queries from the same source, making it difficult to overwhelm the system.
- **Threat intelligence integration:** Modern DNS solutions proactively integrate threat intelligence feeds to block malicious domains.

Cloudflare and Akamai are examples of some of the popular, secure solutions for domain name resolution for public domains.

Web application firewalls

Web application firewalls (WAFs) protect web applications by detecting and blocking malicious traffic (typically HTTP/HTTPS), specifically many of the OWASP top 10 vulnerabilities covered under *Chapter 3, Secure Your Code, Build and Deploy*, such as SQL injection or **cross-site scripting (XSS)**.

In addition, some of the available automated WAFs, such as Check Point Cloud Guard WAF capabilities, enable AI-driven threat detection for proactive protection and bot and DDoS prevention to block attacks targeting application programming interfaces or web applications. WAF capabilities from cloud providers also offer automated runbook responses to attacks, such as blocking specific sources generating the attack.

Real-world relevance

Cloud providers include default support for WAF within their services. *Figure 5.8* shows the default rules in Azure's Web Application Firewall policy. The default rulesets include the most seen threats, from malicious bots (shown in *Figure 5.8*) to SQL injection and **Open Web Application**

Security Project (OWASP) top 10 attacks. Azure also allows for the WAF to be set in prevention mode (which stops access) or detection mode (monitors and alerts based on threats).

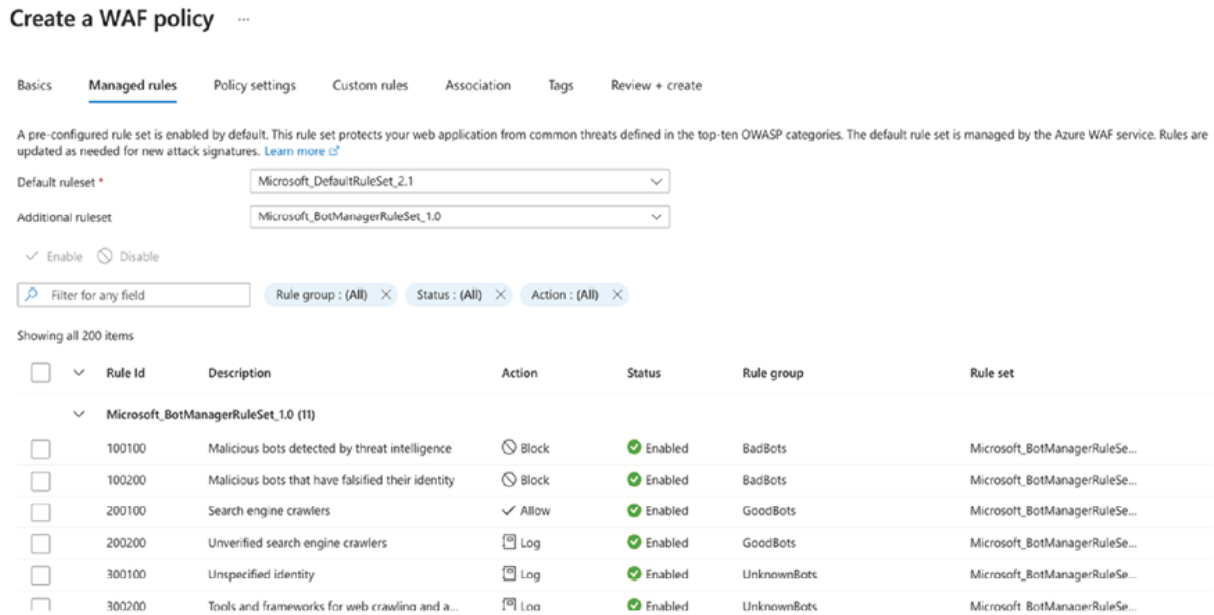


Figure 5.8: Azure Web Application Firewall policy

API gateways

An API gateway is a central point for managing access to APIs, ensuring that only authorized users or applications can interact with them. It is an integral part of securing application layer communication and microservices environments. It can authenticate, authorize, and route traffic based on predefined access control policies, enabling centralized management of access control policies across multiple microservices. API gateways provide the following capabilities:

- Authentication and authorization using OAuth 2.0, **OpenID Connect (OIDC)**, or API keys. Each microservice authenticates through the API gateway, either using mTLS, OAuth 2.0, or another form of authentication; access control policies are also enforced before forwarding the request to the destination.
- Rate limiting to prevent abuse or DDoS attacks.
- Encryption in transit (e.g., HTTPS) to secure data exchanges.
- Similar to web application firewalls, most API gateways also include

protection against OWASP top 10 vulnerabilities.

- Routing of requests to the appropriate back-end services and application programming interfaces.
- API management features include API discovery, documentation, versioning, and developer portals.
- Observability insights such as performance and usage statistics.

Real-world relevance

Let us examine a sample API gateway policy to ensure that only Service A can call Service B's protected API. The way this works is as follows:

1. Any request to a service first goes to an API gateway; typically, the domain resolves to the gateway or a load balancer fronting the gateway.
2. The API gateway then authenticates the service's request.
3. The API gateway checks the authorization policies, such as whether the JWT contains the correct service role or whether the server certificate is valid (mTLS).
4. If the request passes the access control checks, it is forwarded to Service B. However, the API gateway can reject the request and return an HTTP 403 error if the service does not meet the authorization criteria.

For example, with Kong API gateway, you can define a plugin to perform OAuth2.0 or JWT validation to ensure that only services with the correct JWT token can access protected resources in Service B. This is depicted in the following policy fragment:

services:

- name: service-b
url: https://service-b.internal
routes:
 - name: service-b-route
paths:
 - /service-b

plugins:

- name: jwt
config:
 - uri_param_names:


```

      - jwt
      key_claim_name: iss
      secret_reference: "vault:jwt-secret"
      claims_to_verify:
        - exp
- name: mtls-auth
  config:
    ca_certificates:
      - /etc/kong/ca.crt
    allow_without_client_cert: false
- name: acl
  config:
    whitelist:
      - service-role
consumers:
- username: service-a
  custom_id: service-a-id
  jwt_secrets:
    - key: my-key-id
      algorithm: RS256
      rsa_public_key:<key>

```

Network policies

Network policies are a key feature of Kubernetes container orchestration for managing and securing communication between pods (workloads) within a cluster. By default, Kubernetes operates on a flat network model where all pods can communicate freely. Network policies allow administrators to enforce rules that control traffic flow, providing network segmentation and enhanced security. We will discuss this topic further in [Chapter 9, Container Security](#).

Service mesh

If API gateways are the central point for all internal and external API communications, service mesh (e.g., Istio, Linkerd) controls service-to-service communication in distributed applications with multiple

microservices or components. Service meshes often support fine-grained network policies, allowing you to restrict which services can communicate with each other. The capabilities of a service mesh are as follows:

- Automatic encryption and authentication of all service-to-service traffic using mTLS.
- Fine-grained access control at the service level based on identity and roles. The service mesh enforces policies determining which services can talk to each other. For example, only specific services might be allowed to call a service based on their service identity or roles.
- Traffic management policies can also include rules for traffic routing, ensuring that only authorized services can access specific endpoints or services.
- Observability features for monitoring traffic patterns and detecting anomalies.

Real-world relevance

The following code block is a sample policy within the Istio service mesh to enforce mutual TLS on all applications within a namespace. Istio automatically handles mTLS for encrypted service communication.

```
apiVersion: security.istio.io/v1
kind: PeerAuthentication
metadata:
  name: "default"
  namespace: <your namespace>
spec:
  mtls:
    mode: STRICT
```

Istio also supports other features such as fine-grained access control using identity-based access control (via authorization policy), observability (telemetry), traffic management for release, fault injection, rate limiting, traffic mirroring and many other features that are configurable using policies.

Network operations

Once you have designed and implemented a secure network, operating it by monitoring, managing, and optimizing it is crucial for seamless connectivity, security, and performance. This ensures the confidentiality, integrity, and availability of all the critical systems to your employees and end users. Network operations is a specific area within operations focused on an organization's network. A modern **network operations center (NOC)** is the hub for these activities, ensuring round-the-clock oversight and collaboration with **security operations centers (SOCs)** to enhance cybersecurity. The importance of network operations lies in their ability to prevent downtime, detect anomalies early, and respond swiftly to incidents. By integrating Zero Trust principles, such as continuous authentication and micro-segmentation, network operations can further strengthen defenses against both internal and external threats. Network operations typically comprise multiple capabilities, as summarized in [Table 5.6](#), and ensuring these are in place is a critical part of network operations.

Type	Details
Network ACL and firewall rule monitoring	Network access control lists (NACLs) and firewalls enforce traffic filtering based on rules. Monitoring these ensures that configurations align with security policies and block unauthorized access. Regular review of overly permissive access and detection of misconfigured rules is crucial to maintain security.
3 rd party access	As was seen in the Target case study in Chapter 2, Architecture and Pillars of Zero Trust monitoring third party vendor access is critical and tools such as Zero Trust Network Access help restrict access to specific applications instead of the entire network.
NetFlow analysis	NetFlow provides metadata about network flows that enables analysis of network traffic flows to detect unusual patterns and bandwidth misuse, such as large movements of data outside an organization's network.
Threat and anomaly detection	Leveraging machine learning and behavioral analytics to identify deviations from normal network activity. For example, solutions such as Fidelis NDR, Cisco Secure Analytics detect lateral movement within the network or unusual login attempts.
IDS and IPS	Intrusion detection systems (IDS) monitor suspicious activity, whereas intrusion prevention systems (IPS) actively block threats.
NDR	Network detection and response (NDR) solutions analyze network traffic for signs of compromise, focused on advanced threats and bypass traditional defense, for instance, by detecting malware signatures.

Type	Details
SIEM	SIEM aggregates logs from multiple sources for real-time threat detection and compliance reporting. It is an integral part of network operations and security operations in general. We will discuss this further in Chapter 8, Continuous Monitoring .
Patching and updates	Regular software updates reduce vulnerabilities by addressing known exploits.
Disaster recovery and business continuity	Maintaining operational resilience during outages or cyberattacks is crucial to maintaining the availability of systems to end users. This requires organizations to build sufficient redundancy as well as test failover systems regularly to ensure minimal downtime during disasters.
DDoS protection	Protection, monitoring, and response to DDoS attacks is a key capability to ensure the availability of systems. Most CDNs, such as Akamai, Cloudflare, as well as all leading public cloud providers, include this as a standard capability.
Network packet capture and analysis	Intercepting the network traffic to analyze insights around performance and security weaknesses, and help detect anomalies such as unauthorized data exfiltration. Most next generation firewalls support this capability; however, it can be processor intensive and should only be used when needed.
Change control and incident response	Structured processes for managing network changes reduce risks of misconfigurations, while incident response plans enable swift action during breaches. Documenting all firewall rule changes and conducting post-mortem analyses after incidents.
Network compliance and auditing	Ensure networks adhere to regulatory standards through regular audits. Conducting vulnerability assessments and penetration testing as part of a compliance check.

Table 5.6: Network operations

Emerging technologies

As digital devices and virtual environments become more and more prevalent, security is only getting more challenging to execute. There are various new technologies around securing networks to help protect organizations and users against these evolving threats. We will discuss some of the notable ones here.

Artificial intelligence and machine learning

Artificial intelligence (AI) and **machine learning (ML)** are and will continue to impact network security significantly. As we have seen in previous chapters, attacks are becoming increasingly sophisticated due to

advancements in AI and ML. At the same time, the technology is helping build better proactive protections against these sophisticated attacks. Specifically, AI or ML-based systems in network security help analyze user behavior and network patterns to detect threats more effectively; predictive analytics also helps identify vulnerabilities and emerging threats. Unlike traditional systems that rely heavily on rule-based alerts, AI-based cybersecurity systems can be trained continuously and learn from evolving attack patterns and incident response processes. This implies that organizations need to embrace AI and ML in modern-day security. ^[19]

Remote browser isolation

Web browsers are one of the primary threat vectors for most attacks across various chapters; most attacks leverage browsers and use them to compromise data using, say, phishing attacks, etc. This has led to the emergence of businesses. **Remote browser isolation (RBI)** isolates the browsing activity from the end user's device and the company's network; instead, it runs it in a secure, remote cloud-based environment. This means the web pages the user visits are rendered on a sandboxed server and deliver a sanitized representation of the user's browser. This minimizes the risk of malicious content, malware, phishing attacks, and zero-day vulnerabilities. There can be drawbacks to end users, such as increased latency as their session data travels from remote locations and the need for higher bandwidth, which is why it is essential to implement it using the right integrated solutions and select products such as Palo Alto SASE and Zscaler, amongst many others, that support this feature. Adopting RBI would signify a proactive step towards minimizing the surface area of attacks, particularly for businesses with a large remote workforce. ^[20]

Quantum-safe networking

We discussed quantum computing and its effects in *Chapter 2, Architecture and Pillars of Zero Trust*. Quantum computers, which leverage principles of quantum mechanics, can perform calculations at a blazing-fast speed and have the potential to break traditional encryption methods, which rely on computationally complex problems for security. We will discuss encryption and its impacts in further detail in *Chapter 6, Secure Your Data*; however,

this technology has also led to the emergence of quantum-safe networking, which implements advanced cryptographic solutions designed to protect data and communications against the emerging threats posed by quantum computing. Quantum-safe networks integrate **post-quantum cryptographic (PQC)** algorithms that are resistant to quantum attacks to protect network communications. These algorithms use mathematical problems that remain secure even against the immense processing power of quantum computers.

Additionally, technologies like **quantum key distribution (QKD)** enhance network security by leveraging the laws of quantum physics to ensure encryption keys cannot be intercepted or decrypted during transmission. These changes allow networks to adapt to evolving standards and offer a robust defense against threats such as *store now, decrypt later* attacks, where malicious actors collect encrypted data today to decrypt it when quantum technology matures. Embracing quantum-safe networks and approaches will ensure future-proof security for critical infrastructure, sensitive communications, and long-term data protection in an era of advancing quantum capabilities.

Edge computing and network security

Edge computing relies heavily on processing data closer to the source, reducing the latency, bandwidth, and usage from sending data back and forth to servers hosted remotely. The security of edge devices, however, is a common issue due to the difficulty of monitoring and keeping them updated; at the same time, the distributed nature increases the attack surface for an organization and increases the risk of breaches. Multiple strategies are being applied to secure the data in transit and at rest (such as using encrypted tunnels and firewalls), securing the firmware and operating system on edge devices, and implementing better identity and access management solutions to authenticate distributed devices. Hence, technologies such as **Secure Access Service Edge (SASE)**, which are built with edge computing and security in mind, are seeing increased adoption.

Guidelines and checklists

Let us summarize some specific guidelines and checklists for the three roles

typically responsible for maintaining the security posture of products and applications built. Use these with the checklists to secure your network.

For developers and engineers

This sub-section summarizes guidance for anyone developing software or solutions for customers (internal or external). Product managers, designers, and architects can also use this to better understand the key principles that need to be adopted by their team or features to be built:

- **Prepare applications for segmentation:**
 - Review and document application communication patterns and dependencies for granular policies.
 - Ensure applications can function within isolated zones by minimizing unnecessary inter-service communication.
- **Zero Trust Network Access compatibility:** Ensure applications are compatible with identity-aware firewalls and ZTNA solutions that enforce user-to-application access.
- **Secure application design:**
 - Integrate secure identity and access mechanisms across applications and devices.
 - Encrypt all data at rest and transit.
- **Use secure network communication protocols:**
 - Ensure applications are only exposed to current communication protocols (viz., TLS 1.2 or higher for data encryption in transit).
 - Implement WAF within applications to protect from threats like SQL injection and cross-site scripting.
- **Use service mesh for service-to-service communication for:**
 - Design microservices with security in mind, enabling service mesh tools like Istio or Linkerd to enforce mTLS and access control policies.
 - Support identity-aware communication between microservices, which defines which microservices can talk to each other.

- **Monitor logs and anomalies:**

- Ensure appropriate logging of access and communications across applications.
- Setup alerts around communication breakdowns or critical anomalies.

For infrastructure engineers

This subsection summarizes guidance for anyone who deals with infrastructure (either on-premises or cloud):

- **Network design and segmentation:**

- Use firewalls and identity-based policies to isolate workloads and prevent unauthorized lateral movement.
- Deny communication between zones by default; allow only specific traffic based on need.
- Implement micro-segmentation to separate discrete applications using firewalls, network policies, and service meshes.
- Use VLANs or SDN tools (e.g., Cisco ACI, VMware NSX-T) to create isolated zones based on sensitivity (e.g., finance, HR).
- Replace legacy VPNs with Zero Trust Network Access to minimize attack exposure.
- Enforce device authentication and restrict untrusted devices from connecting.

- **Secure firewalls:**

- Use next-gen firewalls for granular control over network traffic based on application awareness.
- Apply identity-aware policies integrated with identity and access systems.

- **Secure communication protocols:**

- Configure secure routing and domain name system capabilities offered by the network appliances and cloud provider.
- Configure web application firewall policies to protect from emerging web threats.

- Configure API gateway and service mesh policies mandating authentication on all flows, rate limiting, and transport layer security.
- **Network traffic logging and operations:**
 - Integrate network device and flow logs into centralized logging, SIEM.
 - Configure dashboards to visualize network flows and metrics.
 - Set up alerts for key KPIs and detect anomalies across routing, domain resolution, and network flows.

For leaders

This subsection summarizes guidance for anyone who leads or manages a team, including product managers, designers, and architects. It also benefits people from supporting organizational functions like risk and compliance:

- **Network security strategy and planning:**
 - Adopt a Zero Trust networking strategy; enforce the *never trust, always verify* principle.
 - Enforce strict access controls across all networks.
- **Policy enforcement:**
 - Establish a process for documentation and review of traffic flows and dependencies during application design for micro-segmentation readiness.
 - Establish guidelines for integrating IAM systems into network security tools like firewalls and ZTNA platforms.
- **Monitoring and incident response:**
 - Regularly evaluate network vulnerabilities using penetration testing and threat intelligence.
 - Leverage SIEM and network monitoring tools to detect compliance violations.
 - Ensure incident response plans align with regulatory breach notification requirements (e.g., GDPR mandates breach reporting within 72 hours).

- **Security culture, training and awareness:**
 - Promote security first culture.
 - Create cross-functional teams involving developers, security engineers, and infrastructure engineers.
 - Train employees on secure coding practices (e.g., OWASP), micro-segmentation concepts, and Zero Trust principles.

Conclusion

Cyber adversaries continuously find ways to exploit vulnerabilities, move laterally within networks, and compromise critical systems. The Zero Trust network approach offers a proactive and identity-aware approach to cybersecurity, ensuring that no entity, inside or outside the network, is inherently trusted. Organizations can limit attack surfaces, mitigate insider threats, and protect sensitive assets by enforcing micro-segmentation, secure communication protocols, least privilege access, and continuous verification. The next chapter will focus on the fourth pillar: secure your data.

Points to remember

Securing your network spans multiple aspects of design, implementation, and monitoring, listed as follows:

- Build your network design considering a layered security approach. Securing the network perimeter is not enough, so use a combination of techniques such as micro-segmentation and identity awareness for a robust network design.
- Implement secure network communications using current versions of supported protocols, such as IPSec, TLS, Secure HTTPS, and mutual TLS. Implement secure routing (e.g. BGP Sec and RPKI) and DNSSEC should be used to protect against route hijacking and DNS spoofing, as well as techniques such as response limiting on domain resolution, to protect from attacks.
- Use continuous monitoring within in-network operations, including monitoring of network access control lists and firewalls, NetFlow analysis,

and Intrusion detection and prevention systems, and integrate logs into security information and event management.

- Emerging technologies will continue to disrupt network security. AI and ML will continue to reduce the barrier to sophisticated attacks and improve the performance and scalability of detection and the cost of identifying and analyzing these attacks. Quantum computing will push the adoption of quantum-safe network principles and algorithms. RBI and SASE are now crucial for better segmenting and protecting the extended network due to remote work and edge computing use cases.

References

1. <https://par.nsf.gov/servlets/purl/10118968>
2. <https://www.sipa.columbia.edu/sites/default/files/2022-11/NotPetya%20Final.pdf>
3. https://www.researchgate.net/profile/Charlie-Pownall/publication/346080185_The_Context_and_Impact_of_Maersk's_NotPetya_cyber_attack/links/5fba37eb299bf104cf6af615/The-Context-and-Impact-of-Maerks-NotPetya-cyber-attack.pdf
4. <https://www.wemeanbusinesscoalition.org/blog/company-profile-maersk/>
5. <https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/>
6. <https://sosintel.co.uk/case-study-maersks-response-to-notpetya-how-cybersecurity-best-practices-mitigated-a-major-cyberattack/>
7. <https://www.sipa.columbia.edu/sites/default/files/2022-11/Sony%20-%20Written%20Case.pdf>
8. <https://www.zdnet.com/article/wi-fi-hack-caused-tk-maxx-security-breach/>
9. <https://www.vcelink.com/blogs/focus/history-of-computer-network>
10. <https://www.techtarget.com/searchnetworking/definition/OSI>
11. <https://www.techtarget.com/searchnetworking/definition/TCP-IP>
12. <https://www.paloaltonetworks.co.uk/cyberpedia/what-is->

microsegmentation

13. <https://www.cisco.com/c/en/us/solutions/collateral/enterprise/design-zone-security/zt-ag.html>
14. <https://learn.microsoft.com/en-us/azure/vpn-gateway/point-to-site-about>
15. <https://learn.microsoft.com/en-us/entra/identity/conditional-access/concept-conditional-access-conditions>
16. <https://aws.amazon.com/what-is/ipsec/>
17. <https://www.nsa.gov/portals/75/documents/what-we-do/cybersecurity/professional-resources/ctr-guide-to-border-gateway-protocol-best-practices.pdf>
18. <https://www.fcc.gov/sites/default/files/AWS%20and%20Secure%20Internet%20Routing%20-%20FCC%20BGP%20Wrkshp073123.pdf>
19. <https://wire.com/en/blog/top-cybersecurity-trends-2025>
20. <https://www.parallels.com/blogs/ras/zero-trust-trends/>

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



CHAPTER 6

Secure Your Data

Introduction

In the previous chapters, we explored the foundational elements of Zero Trust security, covering the essential areas of code integrity, deployment safeguards, identity verification, access management, and network protection. Before progressing toward the operational dimensions, such as ongoing vulnerability management and monitoring, it is crucial to pivot our focus onto any organization's core asset or crown jewel—its data. A lot of security controls and principles are used to protect the data.

Data represents the most valuable resource, often referred to as the *crown jewel* of enterprise operations. In 2013, merely 40% of the global population had internet access, with a significant portion of data traveling unprotected in plaintext; now, that percentage has tilted drastically, with encrypted communication the de facto standard for all flows. Despite such technological leaps over the past decade—widespread adoption of SSL protocols and the expansive growth of internet connectivity—data breaches persist as a formidable threat. This highlights the critical need for multi-layered security frameworks and robust data protection strategies.

Structure

The following topics are covered in this chapter:

- Case study: Marriott
- Data security requirements and design
- Cryptography and encryption
- Access control
- Secure archival and deletion
- Datastore security
- Emerging technologies
- Guidelines and checklists

Objectives

Even with sufficient security across all layers, compromises happen, and you want to protect your crown jewels (the data). This chapter will cover adopting simple, easy, and comprehensive data security management practices. We will cover the importance of data across industries, the various data classifications, and how to protect the different data types. The chapter will cover examples of organizations that fail to protect sensitive customer data. It will explore various techniques for implementing comprehensive data security, which can be used to protect data at rest and in transit through the use of encryption, transport protocols, access controls, and retention policies. By the end of this chapter, you should have a solid foundation in understanding data security classification and requirements, as well as the implementation of key data security measures to protect data at rest and in transit, including advanced options and emerging technologies.

Case study: Marriott

Let us look at a case study for a better understanding:

- **Date:** October 2018 (detected), **Countries:** US, UK
- **Summary:** Marriott Hotel discovered hundreds of millions of customer records, including confidential credit card and passport information, that attackers had exfiltrated. ^[1] Although Marriott detected the attack in

2018, it began in 2014.^[2]

- **Root cause:** An attacker accessed Starwood Hotels' systems in 2014, before its merger with Marriott (2016).^[2] The root cause was a compromised server, allowing attackers to access Starwood Systems' customer data.
- **Impact:** In addition to legal ramifications, recovery costs, and reputational damage, the following were the impacts of this attack:
 - 344 million customers were impacted worldwide.^[3]
 - \$52 million penalty to 49 states and the District of Columbia.^[3]
 - The UK **Information Commissioner's Office (ICO)** fined Marriott £18.4 million.^[2]
- **Detailed timeline:**

Marriott International is a large conglomerate with 30 leading brands and operations in 142 countries worldwide. ^[4] In 2016, Marriott International acquired Starwood Hotels and Resorts. Starwood was another big chain with multiple brands and thousands of properties globally.

On September 8th, 2018, Marriott discovered cybercriminals had breached its guest reservation system, exposing the personal information of hundreds of millions of customers. However, the origins date back to Starwood's, before the Marriott acquisition. Two years after the acquisition, Marriott discovered the vulnerability in 2018 and reported it to law enforcement officials. ^[3]

In July 2014, a physical machine on the Starwood network was compromised, and a **Remote Access Trojan (RAT)** was placed on the server. Exactly how that was compromised was not clear, but typically, this happens via phishing email attacks, which we have seen in some case studies before. Investigators discovered a RAT and MimiKatz, a tool for sniffing out username and password combos in system memory, as they were trying to find clues. These tools could have given the attackers control of the administrator account.^[1]

After the purchase in 2016, Marriott was not yet ready to book guests at its newly acquired hotels using its own reservation system, so Starwood's old system with the infected server remained active for

about two years. ^[1]

When a security tool flagged an unusual database query, Marriott learned it had been hacked. A user made the database query with administrator privileges, but analysis quickly revealed that the person to whom that account was assigned was not the one who made the query; someone else had managed to take control of the account. ^[1]

As per the FTC, there were three different breaches. The first breach occurred in June 2014, involving payment card details of over 40,000 Starwood customers. Undetected for 14 months, Starwood notified customers in November 2015, just four days after Marriott announced its acquisition of Starwood. The second breach was around July 2014, undetected until September 2018, when 339 million Starwood guest account records globally, including malicious actors, accessed 5.25 million unencrypted passport numbers. The third breach, undetected from September 2018 to February 2020, affected Marriott's network, and globally, 5.2 million guest records were compromised, including names, addresses, emails, phone numbers, birthdates, and loyalty account information. ^[3]

Although the breach was discovered in September 2018, it was not disclosed publicly until November. In addition, the company's response to customer inquiries and concerns was considered poor, as it took several weeks for Marriott to set up a call center to handle customer inquiries. ^[2]

- **Learnings:**

- **Strengthen data security:** While the credit card numbers were encrypted, the encryption keys were stored on the same server and scooped up in the breach. As for the passport numbers, while some were encrypted, the majority were saved in clear text. We will cover this aspect in more detail in this chapter. ^[1]
- **Data minimization and deletion:** Customer data retention should be limited to the required duration. Customers must be informed of the purpose of data collection and the specific business need to retain it. They should also be allowed to request the deletion of their data. ^[3] This is one of the key requirements of GDPR compliance, which we will also cover.

In addition, the standard practices around securing identities through secure coding, strong passwords and MFA, access, firewalls and network segmentation controls, patching outdated software and systems, and adequately logging and monitoring network environments are all relevant. These could have prevented attackers from gaining access to the Starwood server or reduced their ability to compromise the rest of the systems.

Let us look at other notable examples.

Case study: Yahoo breach

Yahoo is not a big name in the technology industry today, but it was a big player during the early adoption of the Internet and the World Wide Web. Yahoo faced two attacks that exploited poor encryption practices, resulting in data theft from all 3 billion user accounts, one of the most significant known compromises. The first went undetected for three years, during which hackers obtained names, email addresses, phone numbers, birthdates, and encrypted passwords. The second one affected over 500 million of Yahoo's user accounts; hackers accessed personal data, including security questions and answers (including some unencrypted questions and answers).^[5] Yahoo was not using strong encryption standards to encrypt passwords; in some cases, data was not encrypted. Implementing robust encryption and regular vulnerability testing is crucial to preventing such incidents.^[6]

Case study: LastPass

In 2022, LastPass was part of multiple security incidents. The first attack involved an outsider gaining access to the cloud environment source code and secrets through a compromised device. The threat actor then leveraged a vulnerability in a third-party software to deliver malware, bypass existing controls, and ultimately gain unauthorized access to cloud backups. These backups included system configuration data, API secrets, third-party integration secrets, and encrypted and unencrypted LastPass customer data. The unencrypted personal data is a notable example of a key data security control that will be discussed in this chapter.^[7]

Case study: H&M

H&M was fined € 35.3 million for illegally surveilling several employees, violating data privacy laws. The company kept excessive records on its workforce's families, religions, and illnesses. Superiors had *Welcome Back Talk's* with individuals coming back from vacations and illness absences, which were stored digitally and were sometimes readable by up to 50 other managers. Due to a configuration error in 2019, these notes were accessible company-wide for a few hours. Although the misconfiguration itself is a separate incident, the real issue was the unlawful data retention and lack of privacy. H&M employees' data, including private details about family issues and religious beliefs, was stored without proper justification. [8]

Data security requirements and design

Before implementing specific data security measures and controls, it is helpful to understand the key drivers and use cases behind data security. The main drivers are data classification and governance, regulatory compliance requirements, and data privacy expectations. There are two types of use cases related to data protection: data-at-rest and data-in-transit.

Data classification and governance

As the name suggests, data is classified into various categories based on sensitivity, content, metadata, and expected usage. Typically, this activity is performed at the source, where the data is generated, and is the most essential part of protecting data. [Table 6.1](#) shows the most common types of data classification found in any company, what they are, and the controls (controls will be discussed further in this chapter):

Type	Description	Common controls
Public	Information that is freely available in the public domain (website, news, press releases).	No special protection measures required, appropriate internal reviews and approval process before being published.
Internal	Business only information that is used for regular business activities. Typically meant for internal purposes only and shared with third parties on a need-to-know basis.	Available to employees and third parties working with the company. Basic encryption is recommended, including when sent via email or file shares.

Type	Description	Common controls
Confidential	Secret or sensitive information, unauthorized access can cause legal, commercial, or reputational damage, only accessible on a need basis, customer personal information, sensitive data.	Access requires documented approval. Strong encryption end-to-end, access controls, logging, data loss prevention measures and secure destruction.
Restricted	Top secret or highly sensitive information and if exposed, can cause significant legal and financial impact. Trade secrets, proprietary information, and research.	Access requires documented approval. Strong encryption, access controls, logging, data loss prevention measures, secure destruction and explicit management approval.

Table 6.1: Data classification

Data ownership is another crucial aspect of ensuring data security; it defines who possesses and is responsible for the information (typically, the application business owner). Identifying the roles or people responsible for managing the data lifecycle is crucial, as those individuals are responsible for ensuring compliance against regulations, data protection controls, intended use and purpose, and data integrity to prevent loss.

The term sensitive data is used throughout this topic and most of the book, referring to any data that requires protection. Usually, sensitive data includes all confidential and restricted classifications and, in some cases, internal data. Many core concepts and practices covered here apply to most classifications, although it is best to understand data classification first when designing or building a system.

Real-world relevance

The real-world relevance of data security is as follows:

- Understanding your organization's data classification guidelines is crucial to protecting your data or the data your application generates. These are typically found in the company's information security policy guidelines, and security **subject matter experts (SMEs)** can help ensure they are correctly understood. Regulatory compliance can require additional sensitivity to specific processes to protect data from some of your systems.
- Use the classification and compliance requirements to define the required security controls to protect the data.

- Next, you need a good data labeling strategy that ensures all data (structured or unstructured) is clearly labeled with the classification. Depending on where the data is stored, this can be done in multiple ways, using **data loss prevention (DLP)** solutions or tagging supported by the data stores.

Regulatory compliance

Regulation drives many data protection requirements, and depending on your domain, multiple ones may apply to you. We will cover the details in [Chapter 12, Policy, Frameworks and Compliance](#). A few notable regulations around data security include the **General Data Protection Regulation (GDPR)** and the **Health Insurance Portability and Accountability Act (HIPAA)**.

Real-world relevance

When building systems, ensure you understand the regulations you must comply with to drive your security requirements.

For instance, GDPR requires businesses to inform customers about everyday data processing activities, such as CCTV surveillance, customer data used for marketing, and employee data handling. Any such data, customer or employee data, gathered by the business needs to be appropriately protected from unauthorized access and use, including the use of encryption and responding to subject access requests.

Data privacy

Data privacy is the protection of information about a person and the ability of every individual to control who, what, how, and when their information is shared, communicated, or used. It is a critical part of data security and equivalent to the confidentiality pillar of the CIA triad covered in [Chapter 2, Architecture and Pillars of Zero Trust](#). When designing solutions, it is essential to ensure the following:

- **Consent management:** Obtain informed consent from individuals before processing their data and ensure that consent is revoked.
- **Rights management:** To protect individuals' rights and ensure they control their data and how it is used.

- **De-identification:** Removing personally identifiable information from data to make it anonymous and less sensitive. However, you might still leave behind indirect identifiers that can be used to re-identify the person with additional information later (if required).
- **Anonymization:** Similar to de-identification, it also refers to the removal of the possibility of being able to re-identify; this is done by masking and randomizing.
- **Cross-border data flow:** It is the movement of data across nations or regions. Although this is quite prevalent due to international trade and innovation, it poses specific design challenges due to data protection and residency rules.

Real-world relevance

When building systems, your data security design needs to consider all of these aspects to ensure the confidentiality and privacy of the data. Privacy by design ensures that privacy is integrated into the system's architecture and processes from the very beginning. Some of the data breaches and penalties discussed have been due to a lack of awareness and implementation in the design of the systems, specifically at Marriott and H&M.

Data protection at rest and in transit

There are two distinct use cases for protecting data: data at rest and data in transit (or motion). [Figure 6.1](#) helps visualize the two and their differences. Data at rest is stored and usually static on the server or the user's system. Data in motion, on the other hand, is exchanged between users or systems.

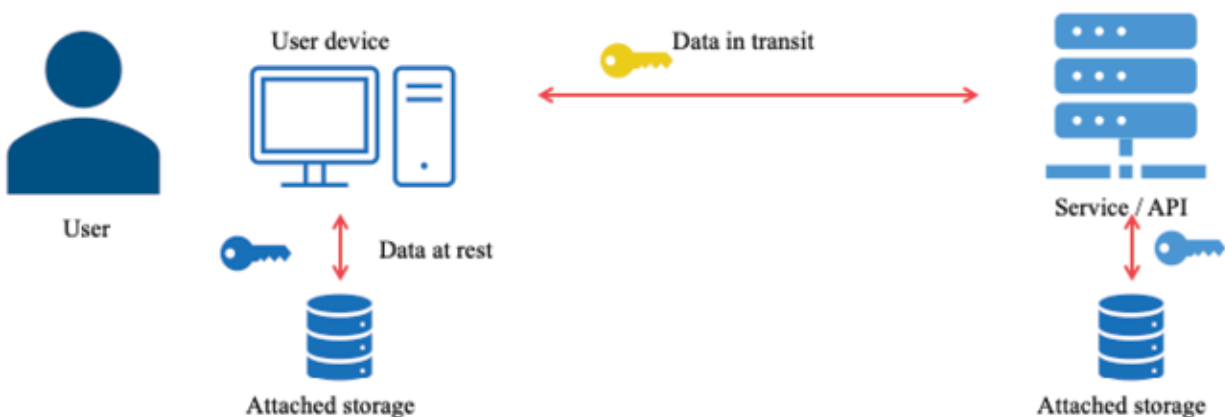


Figure 6.1: Data at rest and in transit

Table 6.2 compares the two and the measures to protect them:

Data at rest	Data in transit
Information stored on a device or system and not actively moving between devices or networks	Also known as data in motion, refers to information that is actively moving from one location/user/system to another.
Data stored on hard drives, solid-state drives, or other storage media: • Information saved in databases • Files on local or cloud storage systems • Backups and archives	Data being transferred over a network: • Information sent via email • File upload or download from cloud • Data transmitted between applications or systems, or users
Symmetric encryption, Access controls, Physical security measures for storage devices	Asymmetric encryption protocols, virtual private networks (VPNs) for secure remote access, and Secure File Transfer Protocols

Table 6.2: Data at rest and in transit

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Applying traditional controls to modern architectures:** Traditional controls and processes must be uplifted to meet today's threats and harness the power of modern technologies. For instance, basic data security, such as access controls and encryption, should be implemented regardless of data classification.
- **Not understanding data residency and classification requirements upfront:** When consuming software as a service and cloud-based offerings, a key starting point should be data residency and usage and storage based on classification.

Cryptography and encryption

The most common way of protecting information is to encrypt it. But what is encryption? Encryption converts plain text (readable form) or data into secret code (unreadable form) that hides its true meaning. If you need to store users' names, dates of birth, addresses, and licenses in a flat file or **comma-separated values (CSV)**, storing in plain text means that if the

storage solution (disk or machine) is compromised, the attacker can access it directly. An alternate approach is to convert the sensitive data fields into a format that cannot be easily interpreted, typically using mathematical algorithms.

The most straightforward algorithm is the Caesar cipher^[9], which typically involves replacing each letter with another. You start by choosing a key, usually a number between 1 and 25. Each letter is then shifted by the number of places the key indicates. [Table 6.3](#) is an example of applying this to a plain text message: HELLO WORLD:

Plain text: HELLO WORLD Key: 4	Add the key (+4) H → L, E → I, L → P, L → P, O → S W → A, O → S, R → V, L → P, D → H
Cipher text: LIPPS ASVPH	

Table 6.3: Caesar cipher

It is hard to decipher the message from the ciphertext unless the person also knows the key (in this case, 4). Caesar cipher is simple and easy to crack if you have enough ciphertext samples, but it requires some computations.

There are two fundamental aspects of safeguarding such encrypted text:

- **Algorithm's complexity:** Caesar is so simplistic and deterministic (as the pattern is similar across all characters) that it is easy to crack. A more sophisticated algorithm that uses randomization would be more secure and more challenging to crack.
- **Key management (security of the key):** It is easier to decrypt data if data and keys are stored together; one reason attackers could decrypt data in case of a Marriott data breach. Secure storage and management of keys are thus essential.

The Caesar cipher is an example of **symmetric encryption**, in which the same key is used to encrypt and decrypt. Symmetric encryption is often used to encrypt large amounts of data, typically at rest in data stores.

Another approach, called **asymmetric encryption**, involves two keys: a public key available to anyone and used to encrypt data, and a private key kept secret and used for decryption; it is also known as public key cryptography. Asymmetric encryption works by choosing a random number to generate a public key, followed by a private key that can help decrypt the

data. The public key is shared with anyone who needs to encrypt data, and the private key is protected. The algorithm can then use a private key to decrypt the data. This technique is most used when multiple parties are involved.

Both these techniques focus on protecting the data that needs to be converted back to plain text. However, what if you did not need that? For instance, a user's password does not ever need to be decrypted if you can compare the ciphertext by itself. This is where **hashing** is used. It is a cryptographic process that converts your plain text to an arbitrary but fixed-size string of characters called a hash value. One of the essential characteristics of Hashing is that it needs to be deterministic, that is, generate the same hash for the same input every time; it has a fixed output length; and lastly, it is a one-way function such that there should be no way to get back the plain text. Hashing is used in password storage, data integrity checks, and digital signatures.

Federal Information Processing Standards

Now that you understand cryptography and encryption, let us look at some standard algorithms. For that, it is essential to comprehend **Federal Information Processing Standards (FIPS)**^[10]. FIPS is a publicly available standard developed by the **National Institute of Standards and Technology (NIST)** in the United States. The standard was created to ensure that government agencies' cryptographic modules and data security protocols met specific security requirements. Although most countries have their own security standards, FIPS is still recognized globally.

The most important FIPS today in cryptography ^[11] are:

- **FIPS 140:** It defines security requirements for cryptographic hardware (such as the Hardware Security Module or HSM) and software (such as the Caesar cipher). The latest active version is 140-3.
- **FIPS 197:** AES is an industry standard for encrypting data.
- **FIPS 186-5:** Digital Signature Standard that specifies algorithms to be used, such as **Rivest-Shamir-Adleman (RSA)**, **Digital Signature Algorithm (DSA)**, and **Elliptic Curve Digital Signature Algorithm (ECDSA)**.
- **FIPS 202:** It defines the SHA-3 family for secure hashing, replacing

FIPS 180-4, which laid out SHA-1 and SHA-2.

- **FIPS 199 and 200:** Federal information security guidelines.

Table 6.4 shows the most used algorithms and protocols:

Algorithm	Type	Description
AES	Symmetric	The AES adopted by the US Government supports key sizes of 128, 192 and 256. 256 is considered the most secure. It replaced legacy data encryption standards (DES) .
RSA	Asymmetric	Rivest-Shamir-Adleman relies on factoring large prime numbers. Many protocols we will talk about later, Secure Shell (SSH) , OpenPGP, and Secure Socket Layer (SSL) rely on RSA. The strength of the encryption is determined by the key and currently, 2048-bit and 4096-bit are considered effective.
ECC	Asymmetric	Elliptic Curve Cryptography (ECC) uses elliptic curves in mathematics. It provides the same security level as RSA but is faster and efficient. Unlike RSA, ECC can rely on smaller key sizes; 256-bit ECC key is equivalent to 3072-bit RSA Key.
PGP	Hybrid	Pretty Good Privacy is a hybrid encryption system used for email and data security. Every user has a public key that is shared and a private key only they know, combining the speed of symmetric encryption and secure key exchange using asymmetric encryption. Typically used in email, files and communications.
SHA	Hashing	Secure hashing algorithm (current version 3) is a secure way of generating hashes with key properties collision resistance and prevents identification of the input message.

Table 6.4: Cryptography algorithms

Real-world relevance

Encryption and hashing are commonly used and adopted for encrypting data at rest or in transit. The best approach to implementing the proper encryption is to use these steps:

1. Define encryption requirements based on information classification levels and regulatory compliance.
2. Understand the parties involved and the purpose of the data (at rest or in transit). Asymmetric encryption is best suited when multiple parties are involved, while symmetric encryption protects data at rest.
3. Choose a strong algorithm for the use cases using the FIPS guidance.

4. Secure the key generation and management process (we will cover this next).
5. Implement encryption using secure processes and libraries, and principles covered in [Chapter 3, Secure Your Code, Build and Deploy](#).

Guidance on the selection of encryption techniques

- Use symmetric (AES) for bulk encryption as it is faster and best for large datasets.
- Use asymmetric (RSA or ECDH) for key exchange and authentication; they are slower but secure for identity validation.
- Field-level encryption (covered later) helps protect **personally identifiable information (PII)** and sensitive data while allowing controlled access.
- **Transport layer security (TLS)**, which we will discuss in this chapter, is the standard for securing communication between parties.
- Use hybrid approaches for efficiency (asymmetric for key exchange, symmetric for data encryption).

[Figure 6.2](#) shows an example of an App Service within Azure that allows you to host applications in the cloud environment without managing the infrastructure. Two of the settings, **HTTPS Only** and **Minimum inbound TLS Version**, ensure the service is only accessible over secure HTTP and the version of TLS is allowed.

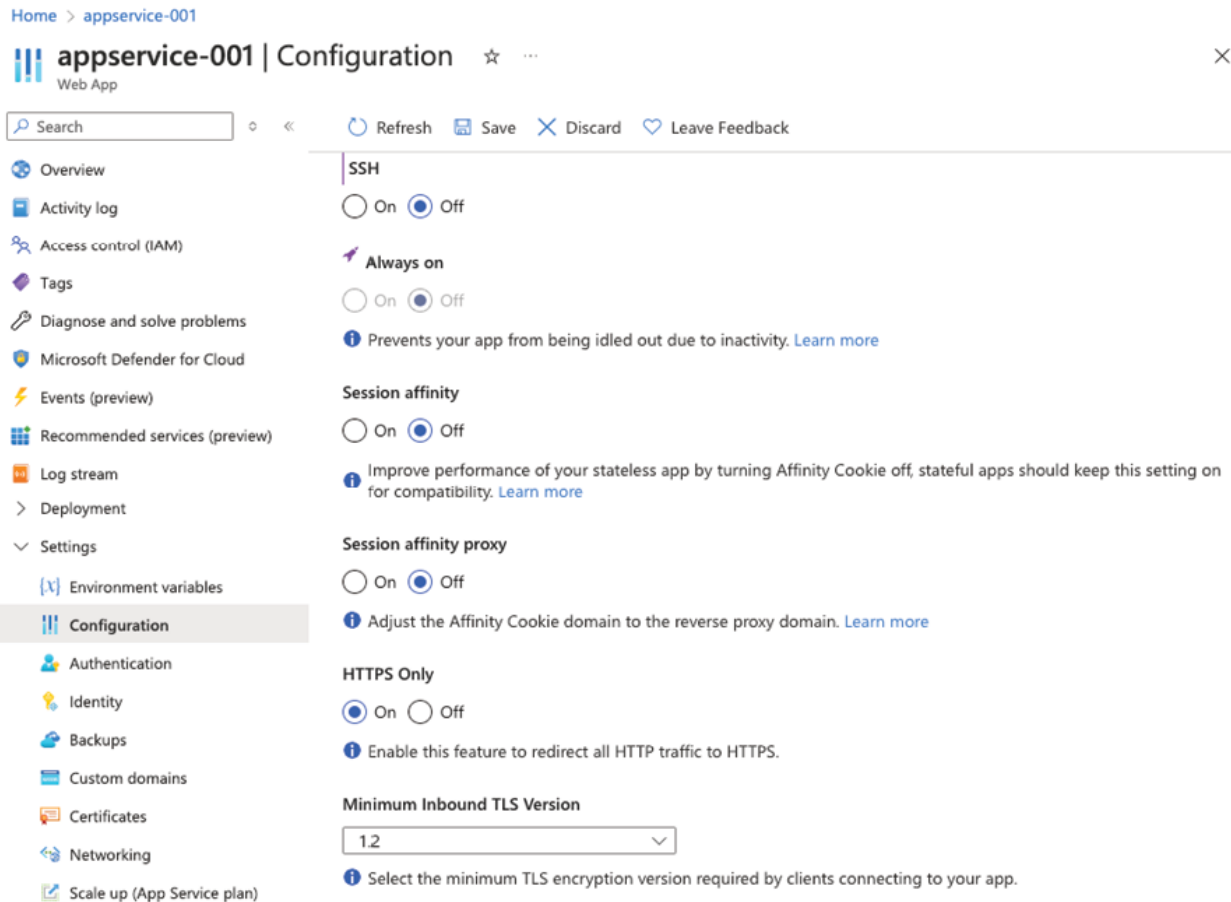


Figure 6.2: TLS setting in Azure

Key management

If you look back at the Marriott case study, keys were hosted alongside the data it protected; this is akin to leaving the keys to your front door next to the door. If you live in a safe neighborhood, that might be okay, but securing your keys is as important as using a strong locking mechanism for your door today. This is why protecting and managing keys is a big topic that organizations spend a lot of effort on. Key management involves the entire lifecycle of the keys, from creating, storing, distributing, and rotating cryptographic keys used in various security processes to destroying them when they are no longer needed. [Figure 6.3](#) shows the key functions of a key management solution:



Figure 6.3: Key management^[12]

Each of these stages is described in the following table:

No.	Stage	Details
1	Generation	This is the process of generating a key. It is recommended to use hardware security modules (HSMs) or trusted key management systems (KMS) to generate keys.
2	Distribution	Once a secure key exchange is created, protocols (e.g., Diffie Hellman) should be used to exchange or distribute keys over secure channels.
3	Use	The user or system uses the key to encrypt or decrypt the message or data. It is important to use secure and up-to-date algorithms in this step.
4	Storage	It is super important to store keys separately from encrypted data, but they should also be maintained in secure storage. Keys should be stored in HSMs or secure key vaults. FIPS 140-2 provides storage guidelines for cryptographic keys.
5	Rotation	Over time, it is also important to rotate keys as the longer a key is used, the higher the likelihood of it being compromised. This requires regular key rotation schedules, ideally in an automatic fashion, which is supported by most HSM and KMS systems.
6	Backup and	Loss of encryption keys can imply a lot of irrecoverable data, and so it is

	recovery	important to maintain secure backups of encryption keys. Implement and test key recovery procedures across the key threats based on your threat model.
7	Revocation and access	Manage access and revoke compromised or expired keys immediately. Use the principle of least privilege for key access.
8	Destruction	Securely destroy keys when no longer needed, and use secure deletion methods for key removal that should make it impossible to recreate them.

Table 6.5: Key management lifecycle

Two additional areas that are equally important but independent of the lifecycle are:

- **Auditing and logging:** Maintain detailed logs of all key management activities (all items listed previously) and regularly audit key usage and management processes.
- **Compliance:** Ensure key management practices align with relevant standards (e.g., NIST and FIPS). Review and update practices regularly to ensure compliance with evolving requirements.

Real-world relevance

For those administering and managing keys in an organization:

- Use HSMs or trusted KMS for storage and operations.
- Implement strong access controls for key management systems.
- Implement key rotation policies to change encryption keys regularly.
- Separate duties for key management tasks to prevent single points of failure.
- Maintain detailed logs of all key management activities.
- Establish secure key backup and recovery procedures.

When building solutions that encrypt data:

- Never store keys with the data; use a secure key management solution.
- Use automation to generate, distribute, and manage keys and encryption.
- Keep encryption protocols and libraries up to date with changes to standards over time.

Figure 6.4 shows the settings for creating an encryption key on AWS. The high-level options include the key type (symmetric or asymmetric), usage

type, and whether the keys are managed by **Amazon Web Services (AWS)** or externally. With symmetric encryption, AWS automatically selects the AES-GCM encryption algorithm on your behalf.

Step 1
● **Configure key**

Step 2
● Add labels

Step 3 - optional
● Define key administrative permissions

Step 4 - optional
● Define key usage permissions

Step 5 - optional
● Edit key policy

Step 6
○ Review

Configure key

Key type [Help me choose](#)

☒ **Symmetric**
A single key used for encrypting and decrypting data or generating and verifying HMAC codes.

☐ **Asymmetric**
A public and private key pair used for encrypting and decrypting data, signing and verifying messages, or deriving shared secrets

Key usage [Help me choose](#)

☒ **Encrypt and decrypt**
Use the key only to encrypt and decrypt data.

☐ **Generate and verify MAC**
Use the key only to generate and verify hash-based message authentication codes (HMAC).

▼ **Advanced options**

Key material origin
Key material origin is a KMS key property that represents the source of the key material when creating the KMS key. [Help me choose](#)

☒ **KMS - recommended**
AWS KMS creates and manages the key material for the KMS key.

☐ **External (Import Key material)**
You create and import the key material for the KMS key.

Regionality
Create your KMS key in a single AWS Region (default) or create a KMS key that you can replicate into multiple AWS Regions. [Help me choose](#)

☐ **Single-region key**
Never allow this key to be replicated into other regions

☒ **Multi-region key**
Allow this key to be replicated into other regions

[Cancel](#) [Next](#)

Figure 6.4: Key management system AWS

The Azure Key Vault service offers a similar service. You can select the key type (Rivest–Shamir–Adleman and Elliptical Curve), key size, activation, and expiration dates. It also includes a rotation policy that you can set to manage the key management lifecycle. [Figure 6.5](#) shows the settings available on Azure:

[Home](#) > [kv-zero-trust-uks](#) | [Keys](#) >



Create a key ...

Options

Generate

Name * ⓘ

Key type ⓘ

☒ RSA

☐ EC

RSA key size

☒ 2048

☐ 3072

☐ 4096

Set activation date ⓘ

☐

Set expiration date ⓘ

☐

Enabled

Yes

No

Tags

0 tags

Set key rotation policy

Not configured

Confidential Key Options

Exportable ⓘ

☐

Immutable ⓘ

☐

Confidential operation policy ⓘ



Figure 6.5: Azure Key Vault settings

Tokenization and data masking

Cryptography and encryption replace plain text data with ciphertext. As a result, if either the datastore or a subset of the data in the datastore itself is compromised, the user can decrypt the data. There are a few drawbacks to encryption:

- It requires expensive infrastructure, such as key management.
- Encrypted data has no direct business value because it cannot be used as-is; it is challenging to use encrypted data for testing and analytics.
- Encryption is reversible, and although most secure algorithms (such as Advanced Encryption Standard AES 256) are virtually uncrackable or can take millions of years, it is possible, and this could change with quantum computing.
- Since encrypted data is stored along with the rest of the system data, it also does not eliminate the need for compliance against regulations that apply to that data.

Tokenization

This leads us to two additional techniques that play a critical role in securing data. Tokenization is a data security technique that replaces sensitive information (such as your permanent account number, PAN) with unique identifiers called **tokens**. These tokens have no intrinsic value and cannot be mathematically reversed to reveal the original data. [Table 6.6](#) shows some examples of tokenization, and the security of the tokenization system depends on the type of system used:

PAN	Token	Comment
3125 005917 23387	7aF1Zx118523mw4cw15x2	Alphanumeric token
4959 0059 0172 3389	729129118523184663129	Numeric token
5994 0059 0172 3383	599400x18523mw4cw3383	Token includes truncated PAN

Table 6.6: Tokenization examples^[13]

Tokenization systems typically require a central repository of tokens mapped to the sensitive data (such as PAN). This system is called a **data vault**. The benefit, though, is that the other systems that use tokens do not require the same level of security. [Figure 6.6](#) shows how tokenization works:

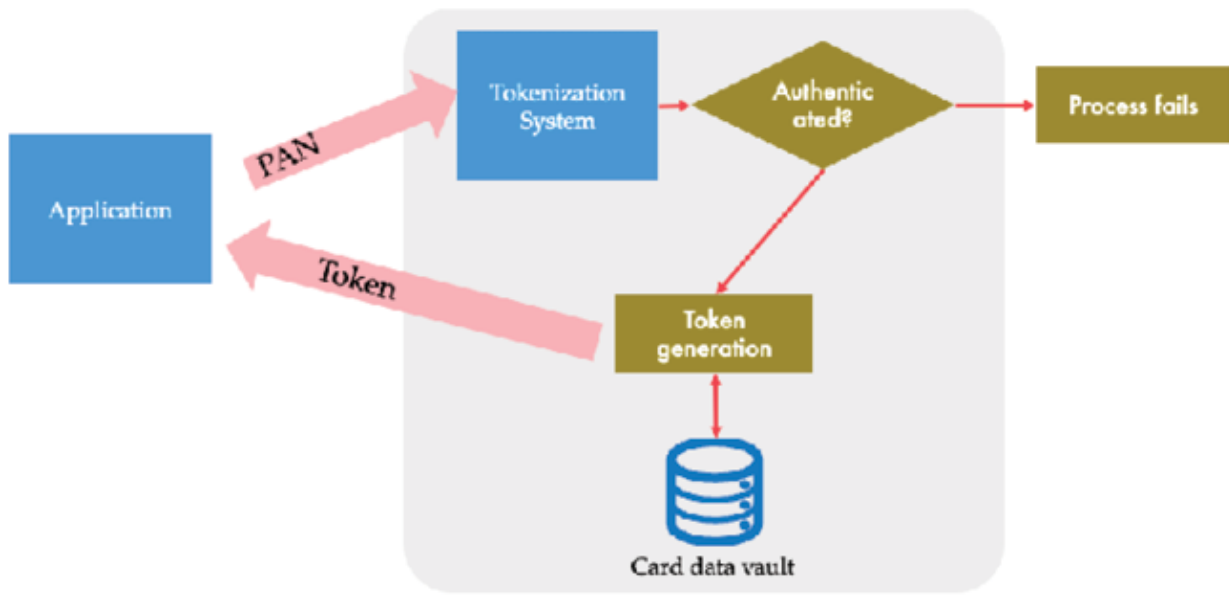


Figure 6.6: Tokenization process [13]

Real-world relevance

Tokenization is best suited for simplifying compliance requirements by separating sensitive and non-sensitive data, or for meeting data localization requirements (where sensitive data must be stored in specific regions). It is prevalent when building a **Payment Card Industry Data Security Standard (PCI DSS)** compliant system. Using tokenization reduces the scope of the system(s) (in this case, the data vault and tokenization process) that must comply with standards such as PCI DSS, as the rest of the systems store tokens^[13]. The process is as follows:

1. Identify and categorize sensitive data elements for tokenization.
2. Define the tokenization process and data vault.
3. Store the original data securely in a separate, encrypted data/token vault.
4. In applications and databases, tokens are used instead of the original data.
5. Implement strict access controls to the token vault.
6. Ensure the tokenization system is isolated from untrusted networks and access controls.

Data masking

Data masking is a similar process but with slightly different objectives. It focuses on replacing sensitive data with realistic data, but not actual data. As

a result, it protects sensitive information while maintaining a functional format. [Table 6.7](#) shows examples of masking techniques used for data based on the expected use cases^[14]:

Original data	Masked	Comment
08054 (customer id)	10004	Replace a customer ID with a new series for test data to preserve sequencing
4959 0059 0172 3389 (PAN)	**** * 3389	Mask the first 12 digits when showing data in the application

Table 6.7: Data masking examples^[14]

There are three standard techniques used for masking:

- **Substitution:** Replace sensitive data with realistic but fake values.
- **Shuffling:** Rearrange data within the same column.
- **Encryption:** Use the reversible encryption we saw previously.

Real-world relevance

Data masking is best suited to developing and testing data where data formats must match, but it does not require live data. A common approach is to refresh your testing environments with production data, and masking is widely used to ensure no sensitive data is in testing. The other use case is for analytics and artificial intelligence; since these use cases require aggregation and statistical analysis, it is essential to maintain the data format, sequencing, etc. Follow these steps to implement data masking during development and testing:

1. Define data masking requirements based on information classification levels and use cases for the masked data (e.g., analytics vs. testing).
2. Choose appropriate masking techniques based on the guidance shared previously.
3. Implement data masking controls based on the defined requirements.
4. Test the masking controls to ensure they are working correctly.
5. Keep records of masked data and original data relationships.

Transport layer security

TLS is the de facto cryptographic protocol designed to provide

communications (data in transit) security over a computer network. It ensures privacy and data integrity between communicating applications. There have been multiple manifestations of this protocol. People may also talk about SSL, the predecessor of Netscape, which was released in the 1990s. Netscape published three versions of SSL. In 1996, the **Internet Engineering Task Force (IETF)**, a standards organization for the Internet, established a TLS working group to standardize the TLS protocol. The latest version of TLS is 1.3. ^[15] However, TLS 1.2 is still widely used and should be deprecated in the coming years.

Figure 6.7 covers the high-level flow of the protocol. As can be seen, a combination of asymmetric and symmetric encryption is used. Asymmetric encryption is used initially during the handshake phase to establish a shared session key. During the handshake phase, the client and server exchange cipher suites (which include the name of the protocol and authentication algorithm for data) and server certificates. The client verifies the authenticity of the server's certificate and agrees on the encryption algorithms. The session key is then generated for symmetric encryption to protect all subsequent communication between the client and server.

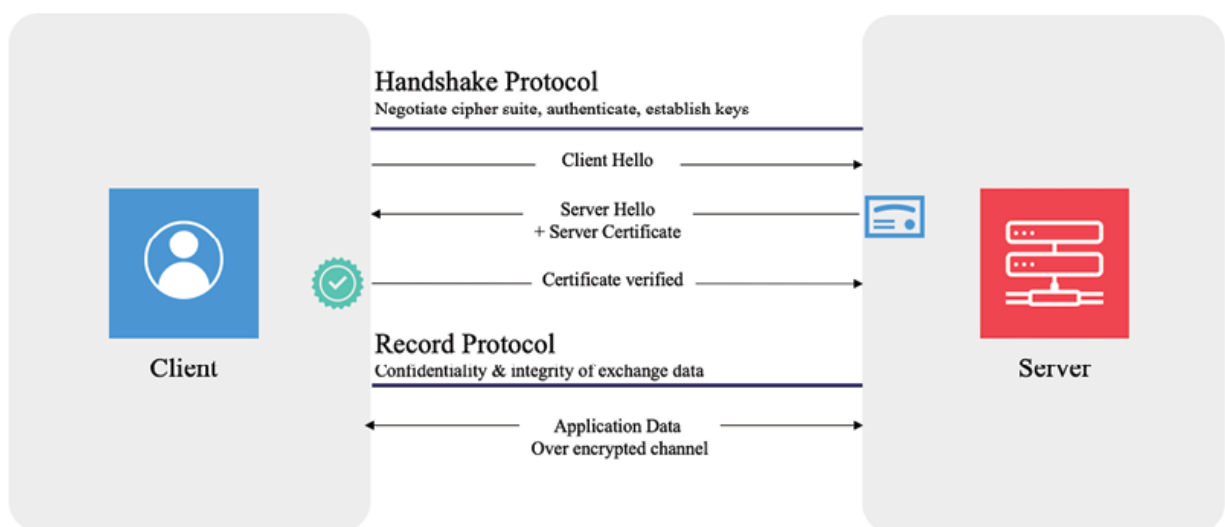


Figure 6.7: TLS ^[16]

Another similar protocol used for secure file transfers is **Secure File Transfer Protocol (SFTP)**. This was also developed by the IETF in 1997. The protocol is used for accessing, transferring, and managing large files

over the web. This protocol operates slightly differently from TLS, as it works on top of SSH communication, which enables secure access to computers across any network.

Real-world relevance

The real-world relevance of TLS is as follows:

- When building solutions, only use TLS 1.2 or higher. Enable **HTTP Strict Transport Security (HSTS)** for web applications using HTTP.
- Ensure proper certificate validation on both the client and server sides. Generate certificates using certificates from trusted **Certificate Authorities (CAs)**.
- Configure servers and your services to use strong cipher suites, following NIST guidelines.
- Regularly update and patch TLS implementations to address known vulnerabilities.

Mutual transport layer security

As we saw, the client validates the server's authenticity using a certificate. Modern systems extend this further by allowing the client to present a certificate that the server verifies. This process is called **mutual TLS (mTLS)** and is a key method within the Zero Trust framework for establishing the authenticity of requests. mTLS is an extension of TLS where both the client and server authenticate each other, providing an additional layer of security. *Figure 6.8* illustrates the differences between this approach and standard TLS. As you can see, after the Server presents its certificate, the client also presents its certificate, which the client verifies.^[16]

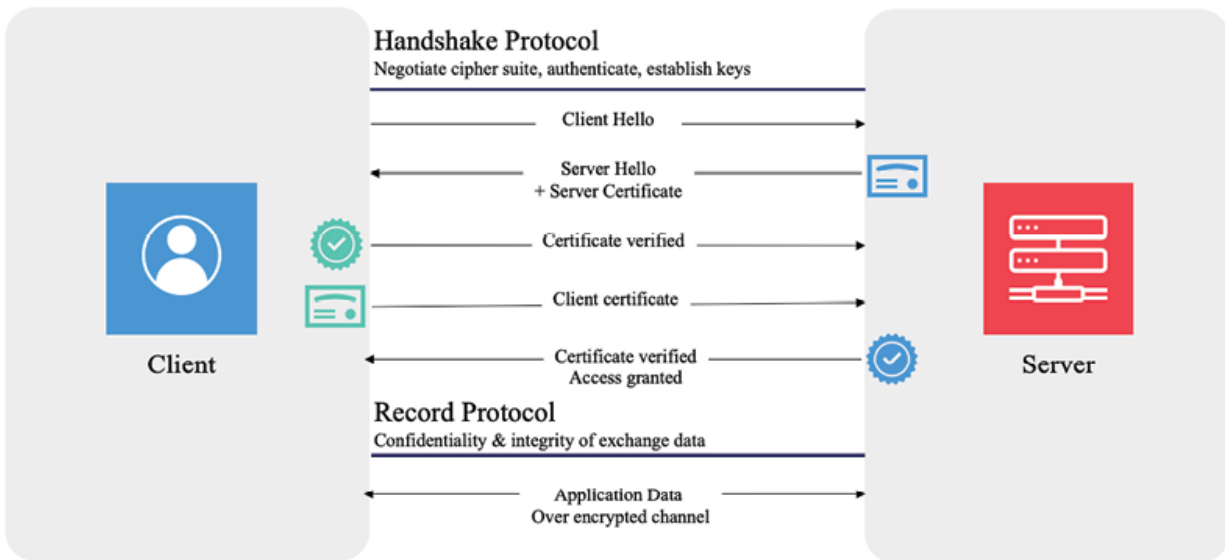


Figure 6.8: mTLS^[16]

Real-world relevance

The real-world relevance of mTLS is as follows:

- Adopt an automated certificate management, generation, and renewal process using **AWS Certificate Manager (ACM)**, Azure Key Vault, and HashiCorp Vault.
- Generate and securely manage client and server certificates. Implement proper certificate validation on both the client and server sides.
- Use a private CA for internal services (CAs are discussed later in this chapter). This is the most crucial consideration within mTLS, as the legitimacy of the certificates exchanged depends on the CA.
- Implement certificate revocation mechanisms (**certificate revocation list (CRL)**, or Online certificate status protocol).

Digital certificates

We discussed certificates in [Figure 6.2](#) in the TLS flow; a digital certificate is an electronic document that proves the ownership of a public key, containing information about the key, its owner, and the issuing authority. With TLS, one of the essential controls is protecting the binding of public keys to entities so their authenticity can be established. Public key is the cryptographic key that can be shared publicly and is used for encrypting messages or verifying digital signatures. The most common form of digital

certificate is X.509^[17], which helps bind the identity to a public key and facilitates digital signatures and encryption. X.509 is an **International Telecommunication Union (ITU)** standard defining the format of public key certificates. *Figure 6.9* shows a sample certificate issued by *DigiCert* to *amazon.co.uk*, which you can find when using their website in a web browser:

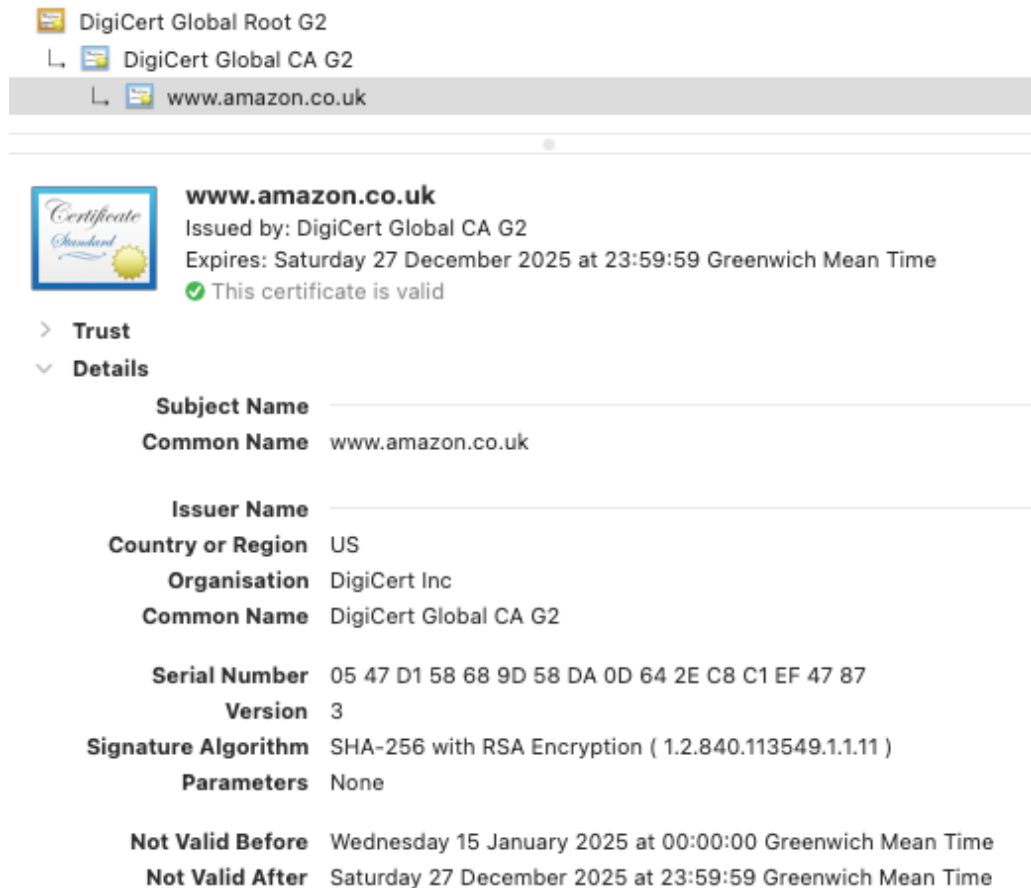


Figure 6.9: X.509 certificate example

An X.509 is ubiquitous in TLS communications, email, virtual private networks, and code signing. They help verify the identity of the individual, organization, or device. However, managing X.509 requires a secure key infrastructure, which we will cover next.

Public key infrastructure

PKI aims to solve the problem of binding keys to entities. It enables confident electronic communication by verifying that the sender or the

receiver is the exact entity they claim to be without compromising data integrity. PKI is derived from public key cryptography (discussed under asymmetric encryption), the technology behind PKI.

The following are the main functional elements of PKI:

- The CA confirms the identity of the parties sending and receiving data. *DigiCert*, *Entrust*, *GlobalSign*, and *Let's Encrypt* are commonly used PKI authorities.
- The CA trusts the Registration Authority to verify the user's identity.
- A PKI repository is a database of all digital certificates issued by the CA. It stores, distributes, and manages certificate updates.
- The CA issues a public key certificate for each identity to confirm it. The certificate includes information about the party holding the private key (or identifying party), the period of validity, and the CA's digital signature for authenticity.
- CAs also issue **certificate revocation lists (CRLs)** for revoked certificates. This can be due to the loss of a private key by the owner or changes to the owner's name, etc.

PKI encompasses everything used to establish and manage public key encryption, including software, hardware, policies, and procedures for creating, distributing, managing, storing, and revoking digital certificates.

CA hierarchy

Protecting the CA is essential to the entire infrastructure. If the CA (or its private key) is compromised, attackers could generate certificates as the CA and impersonate legal entities. Hence, certificate authorities are set up as a hierarchy. Typically, the trust is delegated from a root CA to the following levels: intermediate CAs. Intermediate CAs generate certificates for end users, devices, and services. This is done for enhanced security, scalability, and granular control. The top level is the root CA, typically kept offline using HSMs. [Figure 6.10](#) shows a typical CA hierarchy with root and intermediate CAs. The root CA keys are managed by an offline HSM that is not connected to the network and requires access to the hardware. The other CAs are managed by an online HSM for ease of use.

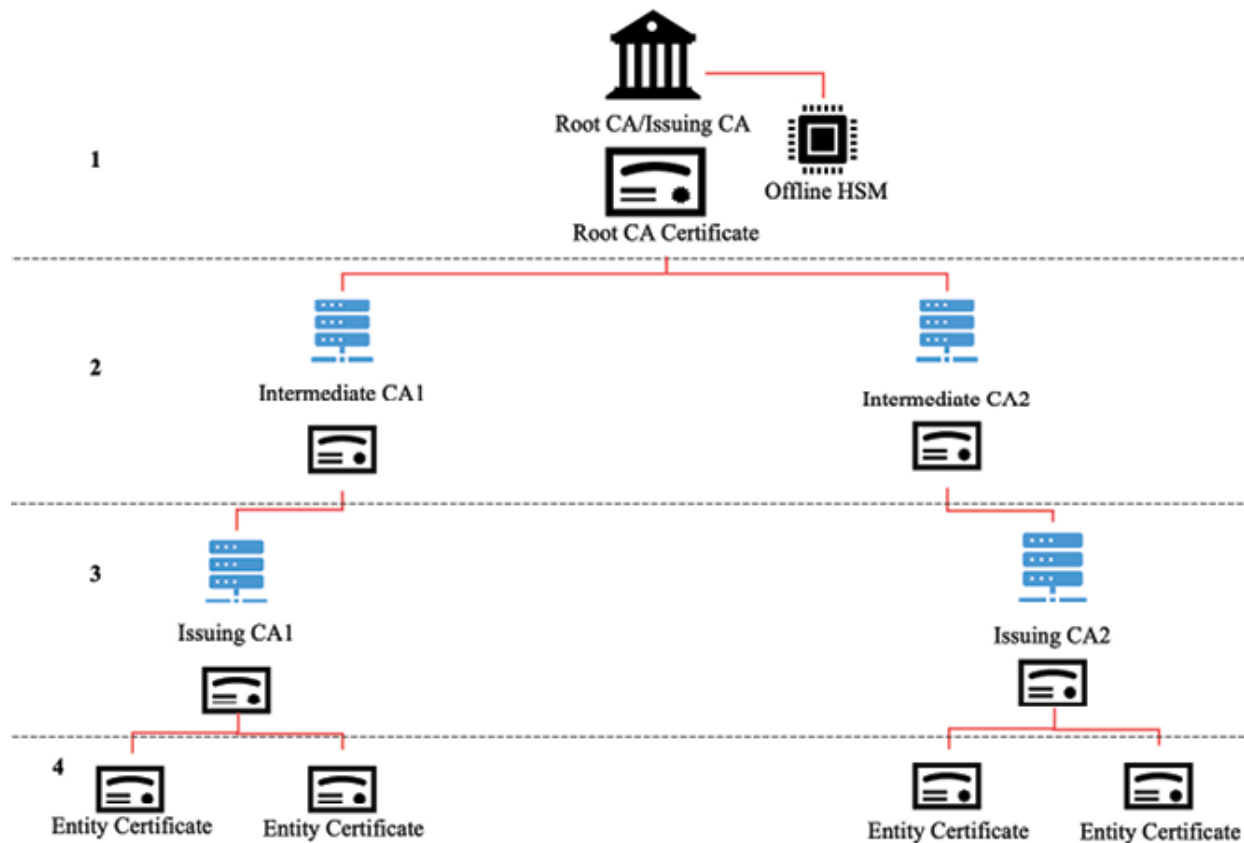


Figure 6.10: CA hierarchy

Another critical part of PKI is private PKI, a subset owned, managed, and used exclusively within the organization. The primary difference is that the organization serves as the certification authority, and a root CA generates the certificates.

Table 6.8 covers all the terms in a PKI and private PKI environment:

Term	Definition
Certificate Authority (CA)	The main component that issues, manages, and revokes digital certificates.
Registration Authority (RA)	Verifies the identity of entities requesting digital certificates from the CA.
Certificate revocation list (CRL)	A list of digital certificates that have been revoked before their expiration date.
Root CA	The top-level CA in a PKI hierarchy, which issues certificates to subordinate CAs.
Intermediate CA	A CA that is subordinate to the root CA and can issue certificates to end-entities or other subordinate CAs.

Internal CA	CA operated and managed within an organization for internal use.
Self-signed certificate	A certificate signed by the same entity whose identity it certifies, often used for root CAs in private PKIs.
Enterprise PKI	A PKI deployment designed to meet the security needs of a specific organization.

Table 6.8: Public key infrastructure terms

Real-world relevance

Most organizations typically have a predefined process, set of roles, policies, hardware, software, and procedures for creating, managing, distributing, using, storing, and revoking digital certificates and managing public-key encryption. However, the following are the key considerations and areas when creating your own CA:

- For public PKI, use trusted CAs such as DigiCert and GlobalSign. For private PKI, design a hierarchical CA structure with an offline root CA (using HSMs) and online subordinate CAs.
- Use HSMs to protect CA private keys. A secure key management (AWS Key Management System, Azure Key Vault) solution can manage and distribute keys.
- Implement automated certificate lifecycle management. Most modern KMS solutions include support APIs to automate key generation and data encryption.
- Regularly audit PKI operations and conduct security assessments.

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Encryption in transit is only required when communicating outside:** This is not true; with the Zero Trust approach, one must plan for when there is a compromise, so every flow should be encrypted end-to-end.
- **Key management is super expensive and not mandatory:** Although on-premises key management and hardware security modules are costly and viable, cost-efficient key management solutions are available on the cloud. Secure key management is not optional when protecting confidential data. Also, the segregation of duties is essential to ensure

that distinct people operate key management and store the data. Automate as much as possible so that no user intervention is required.

Access control

We covered access control in [Chapter 4, Secure Your Identities](#); refer to the chapter for additional details on the techniques. However, it is essential to mention this due to its essential role in data security. Implementing least-privilege access using **role-based access control (RBAC)** to limit data access to authorized personnel is critical to protecting data. When building access controls in your solution, implement strong identity, secure authorization, and privileged access management where appropriate. It is also essential to understand that access controls must be implemented at various levels: first, the infrastructure and platform used to build your software or solution, and second, the access controls for software users.

Secure data lifecycle

Data is the most crucial asset that almost every Zero Trust pillar aims to protect. The present-day world has seen a proliferation of data and information generated across various organizations and communities. Big data is often used to reference the full scale of the large, complex datasets we currently deal with. Understanding the data lifecycle and effective management of the data across the lifecycle is vital to the data's confidentiality, integrity, and availability, as the accuracy of data is critical to most decision-making today, including in artificial intelligence. It is thus crucial to maintain a secure data lifecycle.

Let us explore this with an example. When you visit a doctor or a hospital, you are usually required to answer a lengthy questionnaire. The idea is to gather as much background information as possible when identifying your proper treatment. Have you ever wondered why they ask many of these questions repeatedly? One of the main reasons is that circumstances and people's health change over time. The clinical staff does not want to miss out on a crucial medication or condition that could impact your treatment. For instance, mixing alcohol with pain medication is not recommended, and hypertension medication can affect the types of treatment options available.

As a result, you provide current and up-to-date information about your conditions and medication before any significant treatment. This shows how understanding the available data, its source, location, accuracy, and retrieval methods is necessary.

Data lifecycle management (DLM) and governance help organizations create a comprehensive approach to managing their data assets, ensuring data remains accurate, secure, and compliant across all stages. Ensuring the availability of accurate data is critical in decision-making, and in some instances, this could be a matter of life or death. However, what is this data lifecycle, and what are its key stages? *Figure 6.11* shows the key stages of this lifecycle:



Figure 6.11: Data management lifecycle^[18]

Table 6.9 describes all stages of the data and the key processes involved in maintaining a secure lifecycle:

Stage	Description
Create	Data is either created or collected from external sources.

Process	Data is processed to meet quality standards to ensure it is ready for use.
Store	It is then stored and organized into short and long-term storage based on its classification and desired use.
Use	Users of the data leverage analytics tools and machine learning algorithms to generate insights from the data.
Share	Data and insights are then shared and communicated.
Archive	Once data is not required actively, it is stored in an archive.
Reuse	Reuse data from the archive if required.
Destroy	Data is destroyed when no longer required.

Table 6.9: Data management lifecycle stages^[19]

For each of the stages, there are cross-cutting concerns that impact policies and should be considered to mitigate potential challenges and issues during the lifecycle ^[19]:

- The purpose and value of the data should be identified during collection to ensure it is aligned with the overall organizational goals.
- Privacy is an important consideration across all steps and is meant to ensure sensitive information is protected throughout the lifecycle in compliance with regulations such as GDPR.
- Data ownership defines who owns the created or collected data and is also essential.
- Liability due to a breach or unauthorized access affects most stages.
- Public perception applies to how the data is used and how it is perceived by the public, stakeholders, or other entities. It is critical for maintaining public trust.
- Security applies to all stages and ensures the proper protections are in place.
- Standards and data quality ensure accuracy, reliability, and integrity throughout the lifecycle.

Real-world relevance

Table 6.10 shows the key security controls to be applied at various stages of the data management lifecycle:

Stage	Security
Create	Classification and labelling of data based on sensitivity and regulation Strong authentication and authorization mechanisms Apply encryption at rest and in transit (end-to-end)
Use, process	Strong access controls, end-to-end encryption Least privilege access to users and applications Real-time monitoring and anomalous data access detection Apply strong access controls, end-to-end encryption
Store	Apply strong access controls, end-to-end encryption Network segmentation, isolation of data storage environments
Share	Apply strong access controls, end-to-end encryption
Archive	Establish and enforce data retention policies Implement secure archiving solutions Regular review and update of policies and access
Reuse	Apply strong access controls, end-to-end encryption
Destroy	Adopt secure data sanitization techniques, such as encryption and data masking, prior to disposal Use manufacturer certified destruction methods for physical media Maintain audit trails for destruction activities Apply strong access controls, end-to-end encryption

Table 6.10: *Data management lifecycle stages and security controls*

Figure 6.12 shows a sample lifecycle management policy you can configure for an S3 bucket on S3 that addresses backups, archival, and deletions. The lifecycle rules show that current versions will be moved to a different tier after a certain days, and non-current versions will be permanently deleted after a configured number of days.

Objects | Properties | Permissions | Metrics | **Management** | Access Points

Lifecycle configuration

To manage your objects so that they are stored cost effectively throughout their lifecycle, configure their lifecycle. A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. Lifecycle rules run once per day.

Default minimum object size for transitions
All storage classes 128K

Lifecycle rules (1)

Use lifecycle rules to define actions you want Amazon S3 to take during an object's lifetime such as transitioning objects to another storage class, archiving them, or deleting them after a specified period of time. [Learn more](#)

Lifecycle rule name	Status	Scope	Current version actions	Noncurrent versions actions	Expired object delete mar...	Incomplete multipart upl...
zerotrust-	Enabled	Filtered	Transition to Standard-IA, then	Permanently delete	-	-

[View lifecycle configuration](#)

Replication rules (0)

Use replication rules to define options you want Amazon S3 to apply during replication such as server-side encryption, replica ownership, transitioning replicas to another storage class, and more. [Learn more](#)

Replication rule name	Status	Destination bucket	Destination Region	Priority	Scope	Storage class	Replica owner	Replication Time Control	KMS-encrypted objects (SSE-KMS or DSSE-KMS)	Replica modification sync
No replication rules You don't have any rules in the replication configuration.										

[Create replication rule](#)

Inventory configurations (1)

You can create inventory configurations on a bucket to generate a flat file list of your objects and metadata. These scheduled reports can include all objects in the bucket or be limited to a shared prefix. [Learn more](#)

Name	Status	Scope	Destination	Frequency	Last export	Format
zerotrust-	Enabled	/	s3://s3-us-east-1-zerotrust-	Daily	-	CSV

[View all inventory configurations](#)

Figure 6.12: AWS S3 lifecycle policies

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Data protection compliance is usually an afterthought:** It leads to a lack of or inconsistent handling of archival and destruction processes. Include these in the upfront requirements process and ensure all components support the entire lifecycle management stages.
- **Lack of backup and recovery procedures and testing:** It is one thing to architect or design for backup and recovery and another to test regularly. Besides ensuring your solution is routinely backed up and recovery procedures are documented, they should also be tested periodically to ensure you can recover from them.
- **Global organizations and differences in governance policies:** There are usually subtle differences in data protection across various jurisdictions, specifically concerning where data is stored, processed, transferred, retained, deleted, data breach notifications, etc. Understanding the nuances and planning data flows and architecture when dealing with citizens across various jurisdictions.
- **Awareness and training:** It is critical to ensure data ownership and

custodianship are clearly understood and data owners know their roles. In addition, all individuals should be regularly updated on their role in managing a secure data lifecycle and changing regulations (if any).

Data governance

DLM and data governance are closely interrelated within data security. They have a symbiotic relationship: data governance provides the overarching framework, policies, and standards that guide how data should be managed throughout its lifecycle, and DLM focuses on the practical implementation of guidelines across.

Data protection regulations, such as the GDPR, mandate specific requirements for the protection of personal data. Data governance provides the overarching structure, policies, and standards that guide the management of data throughout its lifecycle. Aside from establishing clear objectives, scope, guiding principles, roles, and responsibilities (including data owners, custodians, and consumers), policies, and procedures, data governance involves the following key capabilities:

- Data discovery and classification identify and classify data across an organization's digital estate, usually through automation.
- Data protection and privacy management provide tools for setting data protection policies and ensuring compliance with regulations such as GDPR and HIPAA.
- Data cataloging and metadata management include data discovery, classification, catalogs, and data assets throughout the organization's entire data estate.
- Data lineage tracks data sources, movement, and transformation across systems.
- Integration and interoperability enable seamless integration with services across organizations and select third-party systems.
- Data insights and analytics generate data usage, patterns, and compliance metrics information to evaluate the effectiveness of data governance programs and policies.

Data governance framework

Follow these steps to implement a data governance framework:

1. Develop a comprehensive data governance framework and policies based on business needs and regulatory requirements.
2. Implement data classification and cataloging to track all data assets and metadata.
3. Establish metrics for data quality, accuracy, completeness, and consistency, and automate quality checks.
4. Implement robust DLM practices for creation, storage, archival, and deletion.
5. Ensure data security and privacy measures, such as encryption and access controls, are in place.
6. Maintain accurate and up-to-date data inventories.
7. Establish transparent processes for handling individual rights requests.

Secure archival and deletion

It is good to protect the data being stored and transferred, but what happens when you no longer need it? Most users only need access to their data for a definite amount of time; for example, users typically view their bank statements for the past three months; beyond that, the data can be archived. Certain regulations may require data storage for multiple years for audit purposes, so archived data must persist according to the regulatory requirements. Data protection regulations, such as the GDPR, require the minimization of personal data use and storage for the purpose for which it is used. This means it must be destroyed when the data is no longer necessary (beyond the standard audit storage requirements). When destroying data, it is essential to ensure that all sensitive data is completely and irretrievably removed from storage media when it is no longer needed. Follow these steps to implement safe archival and deletion:

1. Develop a formal policy for data sanitization, archival, and destruction.
2. Use appropriate methods for deletion based on media type and data sensitivity:
 - a. **For storage disks:** Use manufacturer-provided secure erase and multiple overwrites with random data.

- b. **For physical media:** Employ shredding or degaussing.
3. Maintain records of all sanitization and destruction activities.
4. Verify the effectiveness of archival and deletion methods through audits. Equally, test your archive recovery procedures to ensure the data can be retrieved when required.

Common pitfalls and solutions

The common pitfalls and potential solutions are given as follows:

- **Failing to maintain documentation and audit:** Maintaining supporting documentation and an audit for all archival and deletion actions is critical. Most modern systems support audit actions by default, but they may need to be set up.
- **Not implementing destruction and archival:** If these requirements are overlooked, it can lead to higher costs as data accumulates over the system's lifetime. Therefore, consider them from the beginning and implement solutions as soon as possible.
- **Ignoring encryption key management:** If encryption keys are not destroyed when cryptographic erasure is used, keys can be compromised or retained, and encrypted data can be compromised.
- **Improper handling during equipment disposal:** Not securely disposing of physical devices (e.g., hard drives) before recycling or reselling them can result in sensitive information falling into unauthorized hands.

Datastore security

Most organizations utilize numerous data stores across various applications. These typically range from storage devices (disks), relational databases such as SQL Server, PostgreSQL, and MySQL, and NoSQL databases such as Cassandra and MongoDB, to specialized data warehouses or data lakehouses. These databases and stores store a lot of sensitive information, ranging from personal details about consumers to their health and financial records. Consumers also rely heavily on the availability of this data in this always-on world. Hence, in today's digital landscape, ensuring the security

of data stores is critical to an organization's overall data security strategy.

Datastore security encompasses a range of measures, tools, and practices designed to protect data from unauthorized access, malicious attacks, and data breaches. *Table 6.11* summarizes the techniques already covered in various sections of this book:

Configuration hardening	The Centre for Internet Security (CIS) benchmarks for each datastore serve as a good starting point for securing the database
Authentication	Stronger identities and passwords MFA and SSO Certificate-based authentication for apps, where possible Store credentials in secret management solutions Regular rotation of credentials or short-lived credentials, where possible
Access controls	Least privilege access, granting users only the minimum necessary permissions Separate users from different applications Implement privileged access management for production data
Query controls	Implementing appropriate query controls to help prevent unauthorized data access and potential SQL injection, including: Input validation to filter out malicious SQL commands Parameterized queries to separate SQL code from user input Query rate limiting to prevent denial-of-service attacks
Encryption at rest	Encrypt the entire storage device to protect the device if it is physically stolen or compromised. Typically, this uses symmetric encryption such as AES. Full disk encryption (FDE) is used for any storage disks attached to systems or workstations, permitting access only upon successful authentication to the FDE solution. Field-level encryption is used to encrypt sensitive data attributes or fields within the database for extra protection beyond data at rest and in transit. This is covered in more detail later in the chapter.
Auditing and monitoring	Track user actions and queries (especially around sensitive data) Monitor system performance and resource usage Analyze access patterns to identify anomalies Generate alerts for suspicious activities Maintain detailed logs for forensic analysis and reporting
Vulnerability management	Conduct periodic vulnerability scans and penetration testing to identify potential exploits Promptly applying security patches and updates Identify and address misconfigurations and weak points in the database architecture
Backup strategies	Regular, automated backups of databases Secure storage of backup files, preferably off-site locations

	Encrypt backups to prevent unauthorized access Periodic testing of backup and restore procedures

Table 6.11 : Datastore and database security

Two key topics related to data security require specific mention as they are essential to the adoption of cloud-based technologies: encryption at rest using customer-managed keys and field-level encryption.

Customer managed keys for encryption

We previously covered data protection at rest, which applies to database encryption. All cloud providers make this very simple; in most cases, encryption-at-rest is enabled with keys managed by the cloud provider. However, they also offer the **customer managed keys (CMK)** option. CMK provides control of the lifecycle of keys, including creation, deletion, enabling or disabling, and rotation policies. So, if you need better control of these and better monitoring and tracking of usage of keys, you should set up customer-managed keys and use them when encrypting data at rest. At a minimum, CMKs should be used for all confidential classification data; public and internal classification can use cloud provider-managed keys.

Figure 6.13 shows sample settings when creating a database on AWS; note that in the encryption section, encryption is enabled, and an AWS KMS key is set to a customer-managed key used for encrypting all its data at rest.

7 days

☒ Copy tags to snapshots

Encryption

☒ Enable encryption
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. [Info](#)

AWS KMS key [Info](#)
aws:kms

Account
KMS key ID

Maintenance

Auto minor version upgrade [Info](#)

☒ Enable auto minor version upgrade
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

Maintenance window [Info](#)
Select the period you want pending modifications or maintenance applied to the database by Amazon RDS.

☒ Choose a window
☐ No preference

Start day: Monday

Start time: 00 : 00 UTC

Duration: 0.5 hours

Deletion protection

☒ Enable deletion protection
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

[Info](#) You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

[Cancel](#) [Create database](#)

Figure 6.13: AWS encryption using customer-managed key

Field-level encryption

In addition to the at-rest encryption, sensitive data can be further protected by encrypting specific attributes or fields stored in the database. Field-level encryption is an advanced technique that protects sensitive data by encrypting individual fields within the database. This approach provides granular control over data security.

The key benefits of this technique are as follows:

- Segregation of duties as system and database administrators cannot see data quickly without access to the key used to encrypt fields.
- Protection against insider threats. Even when an attacker gains database access, encrypted fields are not readable.
- Better performance is achieved when encryption is targeted to specific fields.

Field-level encryption typically works as follows:

1. Data is encrypted on the client side (your application) before sending to the database.
2. Encrypted data is stored in the database as ciphertext.
3. When retrieved, the data remains encrypted until it reaches an authorized client.
4. The client decrypts the data using the appropriate encryption key.

Although it has benefits, field-level encryption leads to certain restrictions you should be aware of. One is the inability to perform certain operations on encrypted data, including search, sorting, and filtering. This could mean changing application logic or duplicating specific data fields to support sorting and filtering functionality. The other is the performance costs associated with encrypting multiple fields.

Real-world relevance

The exact implementation of field-level encryption depends on the type of database, use case, and cloud provider. The following are a few examples:

- SQL Server supports the **Always Encrypted** feature; this is configured in the **data definition language (DDL)** statements used to create database tables. The following is a sample creating table statement that encrypts customer names and national identifier fields ^[20]:

```
CREATE TABLE Customers
(
    Name NVARCHAR (60) ENCRYPTED WITH (
        COLUMN_ENCRYPTION_KEY = MyCEK,
        ENCRYPTION_TYPE = RANDOMIZED,
        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
    ),
```

```

        NationalId    VARCHAR    (11)    COLLATE
Latin1_General_BIN2 ENCRYPTED WITH (
        COLUMN_ENCRYPTION_KEY = MyCEK,
        ENCRYPTION_TYPE = DETERMINISTIC,
        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256'
    ),
    Age INT NULL
);

```

- **Mongo DB** supports client-side field-level encryption. The following is a sample code [\[21\]](#) that shows how encryption settings are configured (using settings and **schemaMap**) and how encrypted patient data can be stored in the database:

```

String dekId = "<paste-base-64-encoded-data-
encryption-key-id>";
//create an encryption json schema
Document jsonSchema = new Document().
    append("bsonType", "object").
        append("encryptMetadata", new
Document().append(<keys>))
            append("properties", new
Document().append(<properties>))
//for encrypted properties append the properties in
the following example
//.append("ssn", new Document().append("encrypt",
new Document()
//    .append("bsonType", "int")
//    .append("algorithm",
"AEAD_AES_256_CBC_HMAC_SHA_512-Deterministic")))
//initialize schemaMap
HashMap<String, BsonDocument> schemaMap = new
HashMap<String, BsonDocument>();
schemaMap.put("medicalRecords.patients",
BsonDocument.parse(jsonSchema.toJson()));
//populate client settings with schema configuration
MongoClientSettings clientSettings =

```

```
MongoClientSettings.builder()
    .applyConnectionString(new
ConnectionString(connectionString))

    .autoEncryptionSettings(AutoEncryptionSettings.builder()
        .keyVaultNamespace(keyVaultNamespace)
        .kmsProviders(kmsProviders)
        .schemaMap(schemaMap)
        .extraOptions(extraOptions)
        .build())
    .build();
//create and store patient data for more details see
reference link \[21\]
```

Encryption at the edge

AWS Cloud Front^[22] (a CDN) also supports encrypting sensitive data from the client to the service or application. This feature enables encryption at the edge, where data remains encrypted throughout the entire application stack. Specific data is protected throughout your system, allowing only certain applications to access it.

Continuous monitoring

Continuous monitoring is the ongoing surveillance and analysis of data-related activities to detect and respond to potential security threats, compliance violations, and anomalous behaviors. It is a critical part of robust data security. This comprehensive approach ensures that organizations can maintain the confidentiality, integrity, and availability of their data assets in real-time. *Chapter 8, Continuous Monitoring*, will discuss this in further detail and includes, but is not limited to, data loss prevention, data activity monitoring, incident response, and data recovery procedures.

Emerging technologies

Various new technologies are impacting and will impact data security. Here

is a list of some leading technologies impacting data security and protection:

- **Quantum computing and encryption:** Quantum computing, as we saw in *Chapter 1, Understanding Zero Trust* and *Chapter 5, Secure Your Network*, poses a significant threat to many security protocols, procedures, and frameworks. This space is moving rapidly, with breakthroughs from *Google* and *Microsoft*. Within data security, the most significant impact is on encryption; quantum computers can, in a minute, solve problems that a classical computer would take thousands of years to solve. This makes attacks like brute force entirely plausible; present-day computers require thousands or even millions of years to crack algorithms in use today. NIST 2022 has published guidance on quantum-resistant cryptographic algorithms^[23], which includes CRYSTALS-Kyber for general encryption to secure websites and CRYSTALS-Dilithium, FALCON, and SPHINCS for digital signatures. The current guidance for symmetric encryption is to continue using AES-256. In August 2024, NIST also published post-quantum encryption standards^[24], FIPS-203 for general encryption and FIPS 204 and 205 for digital signatures. In addition, organizations may have to invest in specialized **quantum key distribution (QKD)** systems to implement quantum encryption and integrate them into their existing security infrastructure. QKD uses a quantum system rather than mathematics and transmits what is known as qubits instead of bits. The technology is still in its early stages and faces challenges related to scalability and practical implementation.
- **Blockchain for data integrity:** Blockchain is the technology behind cryptocurrencies, but also offers a robust framework for ensuring data integrity by providing a decentralized, immutable ledger of transactions. It is crucial for data security because it allows for transparency, traceability, and tamper resistance, making it difficult for malicious actors to alter or manipulate data without detection. Blockchain can be instrumental in scenarios where multiple parties must collaborate and maintain a shared, trusted data record, which is common in many regulated industries, such as finance, trading, and healthcare. Organizations need to choose an appropriate blockchain platform to implement blockchain for data integrity, design their data structure, and integrate blockchain validation into their existing processes.

Organizations can choose between public and private blockchain platforms depending on their use cases. When implementing blockchain solutions, it is crucial to consider scalability, consensus mechanisms, and privacy factors. Quantum computing also poses a threat to some of blockchain's core foundations based on current encryption standards. There will be a significant change in this space as quantum computers become prevalent.

- **AI/ML data protection:** AI and ML technologies have a dual-edged impact on data security as they are being used to enhance data security and the sophistication of attacks. AI/ML can help organizations analyze vast amounts of data in real-time to detect anomalies, predict potential threats, and automate security responses. In data protection, it can identify complex attack patterns and adapt to evolving threats more quickly than traditional rule-based systems. At the same time, attackers with access to machine learning models can launch more advanced attacks. Using generative AI, it is easy to compose more realistic phishing emails, and machine learning models can assess large amounts of data and adapt their attacks more quickly and automatically.
- **Confidential computing:** Confidential computing protects data while it is being processed by isolating sensitive data and computations in a hardware-based **trusted execution environment (TEE)**. Security enclaves on *Apple iPhones* and *Android* phones are classic examples. These are used to protect sensitive data such as biometrics, private keys, and any sensitive data that needs to be encrypted. Memory for the enclave is entirely isolated, and keys are maintained at the hardware level. The technology is crucial because it addresses the vulnerability of data during processing, complementing existing encryption methods for data at rest and in transit. This concept is also being applied to the cloud by adopting **private cloud computing (PCC)**. PCC extends the protection of the user data to the cloud while it is being processed and then deletes the data after processing. Confidential computing enables organizations to process sensitive data in shared or untrusted environments, such as public clouds, while maintaining data confidentiality.

Guidelines and checklists

This section covers specific guidelines and checklists for the three roles typically responsible for maintaining the security posture of products and applications, specifically focused on data security.

For developers and engineers

This subsection guides anyone developing software or solutions for customers (internal or external). Product managers, designers, and architects can use this to better understand the key principles their team needs to adopt or the features that need to be built. The guidelines are as follows:

- **Requirements, design and training:**
 - Understand the classification of the data stored and processed by the system.
 - Design appropriate consent and rights management processes.
 - Separate subsystems of applications that store or process different levels of classification.
- **Cryptography and encryption:**
 - Follow NIST guidance for cryptography and encryption.
 - Use data masking when refreshing production data into lower environments or creating test data.
 - Apply tokenization to separate sensitive data where appropriate.
 - Adopt a secure key management lifecycle and store keys separately from data.
- **Data storage security:**
 - Use strong encryption algorithms for sensitive data and hash all passwords.
 - Use current and up-to-date encryption libraries and natively supported encryption processes.
 - Implement field-level encryption as an added measure for sensitive fields.

- **Data in transit:**
 - Encrypt data in transit using TLS/SSL.
 - Use secure ciphers in applications.
 - Adopt mTLS for client authentication.
- **Secure data access:**
 - Strict access controls for users accessing your system.
 - Use parameterized queries to prevent SQL injection.
- **Monitoring, audit and DLP:**
 - Logging and auditing of sensitive data access and changes.
 - Integrate data loss prevention to ensure sensitive data is encrypted and not leaked or accessed by unauthorized users and applications.
- **Data lifecycle, backup and recovery:**
 - Implement backups of all critical configuration and transactional data.
 - Implement sufficient redundancy to deal with failures.
 - Implement secure data erasure and deletion to meet compliance needs.
 - Test recovery procedures to ensure data can be restored from backups.

For infrastructure engineers

This subsection guides anyone who deals with infrastructure (either on-premises or cloud). The guidelines are as follows:

- **Requirements, design and training:**
 - Understand data classification for data stored and processed by the systems.
 - Separate subsystems that store or process different levels of classification.
- **Cryptography and encryption:**
 - Follow NIST guidance for cryptography for encryption.
 - Set up a secure key management system to secure key storage.
 - Implement automated and secure key lifecycle, including periodic

rotation.

- **Data storage security:**

- Ensure strong encryption algorithms are used for sensitive data and hashing for passwords.
- Configure all data stores to be encrypted at rest based on data classification requirements.
- Integrate data stores with secure key management.

- **Data in transit encryption:**

- Encrypt data in transit using TLS/SSL and configure secure ciphers.
- Adopt mTLS for client authentication.
- Configure policies on the cloud and containers to adopt secure communications.
- Public key interchange and key management.
- Set up secure key management systems and processes.
- Set up key rotation regularly (automated where possible).
- Set up private PKI infrastructure automation to manage internal keys.

- **Monitoring, audit and data loss prevention:**

- Set up centralized logs and metrics for all security audit logs, including database activity logging.
- Integrate data loss prevention with your centralized logging solution.
- Set up alerting for key KPIs and anomalies.

- **Data lifecycle, backup and recovery:**

- Implement backups of all critical configuration and transactional data.
- Implement secure data erasure and deletion to meet compliance needs.
- Implement sufficient redundancy to deal with failures.
- Test recovery procedures to ensure data can be restored from backups.

For leaders

This subsection guides anyone who leads or manages a team, including

product managers, designers, and architects. It also benefits people supporting organizational functions like risk and compliance:

- **Security planning and requirements:**
 - Ensure all planning includes key security requirements and considerations.
 - Ensure all security and compliance SMEs are part of the planning.
 - Prioritize sufficient technical debt remediation and automation.
- **Security culture, training and awareness:**
 - Plan regular security training for teams.
 - Identify security champion(s) and raise awareness.
 - Foster a culture of security awareness.
- **Compliance and risk:**
 - Develop and maintain risk register and asset register, and conduct regular risk assessments and audits.
 - Ensure adherence to appropriate compliance standards, internal and external, viz GDPR, PCI DSS.
 - Raise appropriate risks and review risk mitigations and exceptions regularly.
- **Metrics and KPIs, monitoring and incident response:**
 - Incident response planning.
 - Review and monitor key metrics such as mean time to detect and respond, incident security levels, false positive rates, policy compliance rates, incidents, and alert volumes.
 - Report key KPIs to senior leaders.
- **Data governance:**
 - Establish comprehensive information security policies and regular reviews.
 - Ensure data classification and risk assessment are clearly understood.
 - Define clear roles and responsibilities for data security.
- **Tools and process:** Invest in automated data loss prevention, key

management, and other tools for data security, and adopt the latest security standards and tools.

- **Data lifecycle, backup, and recovery:**
 - Plan backup and redundancy of all critical configuration and data.
 - Ensure secure data erasure and deletion to meet compliance needs.
 - Regularly test recovery procedures from backups.

Conclusion

Data is an organization's most valuable asset, requiring robust protection measures. This chapter covered data security management practices, including classification, encryption, access controls, and backup strategies. Case studies such as those of Marriott and Yahoo highlighted the importance of data protection. In the next chapter, we will learn about securing your network.

Points to remember

- Data classification is crucial for data security, with different levels requiring varying protection measures. Regulatory compliance, such as the GDPR and HIPAA, drives data protection requirements, emphasizing the importance of consent, rights management, and de-identification.
- Cryptography involves encryption and hashing to protect data. Symmetric encryption uses a single key for encryption and decryption, while asymmetric encryption uses a public key for encryption and a private key for decryption. Tokenization replaces sensitive data with unique identifiers, while data masking replaces sensitive data with realistic data but not actual data.
- Access controls and key management are fundamental aspects of safeguarding encrypted data, with more sophisticated algorithms and secure key storage enhancing security.
- Encryption at rest and field-level encryption protect sensitive data, but

key management and performance considerations are crucial.

- TLS ensures secure communication, and mTLS adds an extra layer of security by requiring client and server authentication. PKI is crucial for data security, using a hierarchy of CAs for trust delegation. Private PKI, managed within an organization, offers enhanced security and control.
- DLM and governance are linked to data security. DLM focuses on implementing guidelines across the data lifecycle, while data governance provides the framework and policies. Data protection laws, such as GDPR, outline individuals' rights and place specific requirements on data management, necessitating efficient and standardized data storage, management, retrieval, erasure, and request mechanisms.
- Continuous monitoring, which involves data loss prevention, activity monitoring, incident response, and recovery procedures, is crucial for maintaining data security.^[2]
- Emerging technologies, such as quantum computing, blockchain, AI/ML, and confidential computing, significantly impact data security, requiring organizations to adapt and implement robust security measures.

References

1. <https://www.csoonline.com/article/567795/marriott-data-breach-faq-how-did-it-happen-and-what-was-the-impact.html>
2. https://www.researchgate.net/publication/372524901_Marriott_International_Data_Breach
3. <https://www.ftc.gov/news-events/news/press-releases/2024/10/ftc-takes-action-against-marriott-starwood-over-multiple-data-breaches>
4. <https://www.marriott.com/marriott/aboutmarriott.mi>
5. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2883607
6. <https://www.reuters.com/article/technology/yahoo-security-problems-a-story-of-too-little-too-late-idUSKBN1480AM>

7. <https://blog.lastpass.com/posts/security-incident-update-recommended-actions>
8. <https://www.gdprregister.eu/news/hm-gdpr-fine/>
9. <https://www.101computing.net/caesar-cipher/>
10. <https://www.nist.gov/standardsgov/compliance-faqs-federal-information-processing-standards-fips>
11. <https://csrc.nist.gov/publications/fips>
12. <https://www.techslang.com/definition/what-is-a-key-management-system/>
13. https://listings.pcisecuritystandards.org/documents/Tokenization_Guidelines_Info_Supplement.pdf
14. https://www.researchgate.net/publication/230639917_Design_of_Data_Masking_Architecture_and_Analysis_of_Data_Masking_Techniques_for_Testing
15. https://owasp.org/www-chapter-london/assets/slides/OWASPLondon20180125_TLSv1.3_Andy_Brodie.pdf
16. <https://www.cloudflare.com/learning/access-management/what-is-mutual-tls/>
17. <https://www.ssl.com/faqs/what-is-an-x-509-certificate/>
18. <https://www.flaticon.com>
19. https://www.researchgate.net/publication/372663298_Secure_Data_Management_Life_Cycle_for_Government_Big-Data_Ecosystem_Design_and_Development_Perspective
20. <https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine>
21. <https://www.mongodb.com/docs/manual/core/csfile/>
22. <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/field-level-encryption.html>

23. <https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms>
24. <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>

CHAPTER 7

Manage Vulnerabilities

Introduction

In early 2018, the cybersecurity world was unsettled when researchers unveiled the Spectre^[1] and Meltdown^[2] vulnerabilities, which affected nearly every modern processor. These hardware-level flaws allowed malicious programs to steal sensitive data processed on affected systems. Specter attacks involve inducing the victim to perform operations that would not occur under normal circumstances, potentially leaking confidential information. Conversely, Meltdown used out-of-order executions on modern processors to access memory locations, including personal data and passwords. The discovery of these vulnerabilities was particularly alarming because they exploited fundamental aspects of CPU design, specifically speculative execution used as a performance optimization technique in most modern processors. Since it impacted most modern-day processors, it was widespread, affecting desktops, laptops, cloud servers, and even smartphones. Most cloud providers had these patched automatically within days.

This example underscored the criticality of addressing hardware-level security issues and the need for a more comprehensive approach to vulnerability management. Keeping vulnerabilities at bay is how you try to keep attackers away. This chapter will cover the basics of vulnerabilities and their impact on the security of your system and organization based on real

examples. It will also cover tools (such as SCA, OSS, runtime protection, and security testing) and prevention techniques and processes you must implement. By understanding the nature of vulnerabilities like Spectre and Meltdown and implementing robust vulnerability management practices, organizations can better protect themselves against potential threats and maintain the integrity of their systems and data.

Structure

The following topics are covered in the chapter:

- Case study: Capital One
- Case study: Equifax
- Core concepts
- Emerging technologies
- Guidelines and checklists

Objectives

After securing all the aspects we covered previously in the book, namely code, build, and deploy, identities and access, network, and data, managing vulnerabilities in your system continuously is an essential part of ensuring a trusting environment. This chapter introduces the basics of vulnerability and typical vulnerability management, followed by different tools and processes to detect and protect you from vulnerabilities. It will also introduce concepts around assessing risks of vulnerabilities, followed by how to remediate and verify. We will also cover emerging technologies and threats, enabling attackers and organizations to protect their systems.

Case study: Capital One

Let us look at a case study:

- **Date:** July 2019, **Country:** US
- **Summary:** An external individual obtained unauthorized access to

certain types of personal information from Capital One that it routinely collects from individuals during credit card applications. This was identified when an outsider reported that the data was available on a GitHub page.^[3]

- **Root cause:** A firewall misconfiguration allowed the attacker to execute commands on Capital One's server, enabling unauthorized access to their **Amazon Web Services (AWS)** servers.
- **Impact:** The hacker obtained the names and addresses of about 100 million individuals in the United States and 6 million in Canada. 140,000 Social Security numbers and 80,000 linked bank account numbers were compromised. Capital One had to pay \$80 million penalty to a U.S. regulator.^[4]

Detailed timeline

Capital One^[3], a large consumer bank in the United States with current revenues of \$30 billion, was one of the early adopters of the cloud since 2016 and invested heavily in migrating from on-premise datacenters to the cloud. In July 2019, an external individual gained unauthorized access to certain types of personal information from Capital One credit card customers and individuals. The company discovered this incident through the Responsible Disclosure Program, in which an outsider reported that Capital One's customer data was available on a GitHub page. *FBI* investigations arrested a Seattle woman in this regard, who had previously worked at *Amazon*.

Further analysis showed that the attacker created a scanning software tool that they developed to identify servers with misconfigured firewalls, allowing the execution of commands from outside to penetrate and access servers. The attacker found a misconfigured **web application firewall (WAF)** running on a virtual machine used to protect web applications. This particular firewall, known as ModSecurity, is run on an open-source Apache Web Server to protect against various vulnerabilities. The firewall was not properly restricted and allowed requests to a back-end AWS service, known as the metadata service, with temporary credentials to the environment.

The attacker then used a **server-side request forgery (SSRF)** attack to bypass the firewall and gain access. SSRF involves passing a **Uniform**

Resource Locator (URL) that the server does not expect or deal with safely (say, a URL or file field in the parameter). The attacker then tricks the server into executing commands on behalf of the user and gaining access to internal systems to leak sensitive data. The attacker used this to relay commands to the back-end resource within AWS, known as the metadata service. The attacker used this to obtain credentials for an AWS role.

Once they had access, the credentials were used to list files from AWS **Simple Storage Service (S3)** buckets of the compromised account. They then transferred 30GB of Capital One credit application data to the attacker's local machine. This was done for more than 700 buckets.

Learnings

The following are the learnings:

- **Least privilege access:** The server with ModSecurity seems to have had broad permissions to hundreds of S3 buckets. The specifics of the server are not precisely known; however, it is not typical for a firewall to have direct access to data. In addition, there should have been a need to expose hundreds of buckets; the team should have granted only necessary access.
- **Secure configurations and vulnerability management:** What caused the server misconfiguration is unknown; however, multiple measures could be taken to ensure server configurations are safe and detect misconfigurations if they happen. When it comes to the security of firewalls, adopting the **Center for Internet Security (CIS)** benchmarks for web servers and firewalls can help ensure configurations are secure.
- **Ongoing security testing is necessary:** Critical devices and servers, including firewalls and critical network devices, should be part of manual or automated penetration testing and ongoing security reviews. This can help uncover any misconfigurations and vulnerabilities that may have been left behind during implementation.
- **Continuous monitoring:** A robust intrusion detection and monitoring system could help detect anomalous IAM API calls and spurious access to S3 buckets. Transferring data from 700 buckets should ideally have triggered some alarms that could have triggered appropriate incident response procedures.

- **Automatic remediation:** With continuous monitoring, automated workflows can be implemented to generate alerts and to remediate vulnerabilities and misconfigurations automatically.

Case study: Equifax

Let us look at the case study:

- **Date:** September 2017, **Country:** US
- **Summary:** In March 2017, a critical vulnerability in Apache Struts software, which was quite popular, was disclosed. One of Equifax's systems built on this technology was not patched. At the same time, attackers began a cyberattack on Equifax that lasted 76 days. As part of the attack, they gained unencrypted credentials to access sensitive data from multiple databases. This was one of the largest cybercrimes related to identity theft due to its massive scale.^[5]
- **Root cause:** Equifax used an outdated software component and failed to patch a known vulnerability in the Apache Struts framework.
- **Impact:** Massive data breach compromising data of 148 million Americans (nearly half the US population) and 11 million British citizens ^[6]. The Federal Trade Commission in the US settled for \$ 425 million to help people affected by the breach. The UK regulator fined Equifax £11 million for allowing attackers to access the data of 13 million plus UK customers.

Detailed timeline

Equifax is a global credit reporting agency that provides credit scores and reports, among other financial services, across more than 20 countries. Founded in 1899 and with over 14,000 employees, it is one of the three largest credit reporting agencies. The breach was caused by an Apache Struts vulnerability.

In March 2017, Apache Struts software disclosed a critical vulnerability (CVE-2017-5638), a zero-day exploit. The vulnerability allowed remote command injection attacks by incorrectly parsing Content-Type HTTP headers. Attackers could execute commands under the privileges of the web

server it was running on and receive data back from the server.

Although Equifax used this software in many applications, it failed to patch its systems. One consumer dispute portal built in the 1970s was running this vulnerable software.

On May 13, 2017, attackers began a cyberattack that lasted for 76 days and dropped *web shells* (a web-based backdoor) to obtain remote control over Equifax's network. During the attack, they found a file containing unencrypted credentials (usernames and passwords) that they used to access 48 unrelated databases. Attackers sent 9,000 queries on these 48 databases, successfully locating unencrypted **personally identifiable information (PII)** data 265 times. The data was transferred out of the Equifax environment. Equifax did not see the data exfiltration because the device used to monitor network traffic had been inactive for over a year due to an expired security certificate.

It was not until July 2017 that Equifax updated the expired certificate and immediately noticed suspicious web traffic from an IP address originating in China. The suspicious traffic exiting the application potentially contained image files related to consumer credit investigations. After the discovery, they were under active attack, and Equifax immediately launched an incident response effort to shut down the web portal for emergency maintenance.

Learnings

The following are the learnings:

- **Vulnerability management:** A robust vulnerability management process that scans and identifies such vulnerabilities in software across the organization, recommends fixes, and assists with prioritization and verification could help resolve these issues promptly.
- **Timely updates and patching:** The primary cause of the breach was the inability to promptly fix vulnerabilities in the Apache Struts framework. Another problem was the expired certificate, which impacted their ability to detect the attack on time. These issues could have been managed if timely updates and patching processes had been implemented.
- **Least privileged access:** The scale of the impact was compounded because around credentials were unencrypted, allowing attackers to gain

access to 48 unrelated databases. Organizations must adopt the least privileged access across systems and ensure no single component has such broad access or credentials across their estate.

- **Secure credentials and encryption:** Exactly why the credentials were unencrypted is unclear, but these should be managed securely using an external secret store that encrypts the data.

Other notable examples

The following are other notable case study examples:

- **Kubernetes:** Kubernetes is an open-source container orchestration platform that is now globally adopted by many organizations. We will cover more about containers and Kubernetes in *Chapter 9, Container Security*. In August 2019, a critical vulnerability was identified based on an external audit conducted on Kubernetes that revealed that HTTPS connections were not authenticated.^[7] Another report from Aquasec^[8] in 2023 suggested they found over 350 exposed Kubernetes clusters that could be exposed to potential threats and attacks of various types. For a technology that is ubiquitous today with modern software development, this paints a scary picture of how vulnerable organizations are.
- **Zoom zero-day vulnerability:** Another solution that saw a massive adoption during the pandemic due to remote work was virtual meetings and videoconferencing solutions. *Zoom* was one of the largest beneficiaries of the boom. However, in 2020, vulnerabilities were found in Zoom^[9]. One of these vulnerabilities allowed the injection of the **Universal Naming Convention (UNC)** path from the chat feature when using the application on Windows. Clicking on the link would allow limited remote code execution and lock users' credentials. Another vulnerability allowed attackers to install malware on the Mac operating system.

Core concepts

A vulnerability is a weakness or flaw in a system that cybercriminals can exploit to gain unauthorized access or cause harm. These vulnerabilities often stem from coding errors, misconfigurations, or design flaws and can be

leveraged by attackers to disrupt operations, steal data, or compromise security.

Vulnerabilities can be classified into various types based on when and how they occur. The following are some commonly known types:

- Software vulnerabilities occur due to known bugs or unresolved issues within software that attackers can exploit. **Structured Query Language (SQL)** injection in MOVEit, covered in [Chapter 3, Secure Your Code, Build and Deploy](#), or the Apache Struts vulnerability in the Equifax case study, are all examples of software vulnerabilities.
- Hardware vulnerabilities are typically hardware design vulnerabilities in servers, routers, and peripheral devices that attackers can abuse.
- Misconfigurations are default or incorrectly configured settings or unnecessary services that can open doors for unauthorized access. The Capital One example included a misconfigured firewall.
- Network vulnerabilities are typically misconfigurations in network devices or controls such as firewalls.
- Zero-day vulnerabilities are those that attackers exploit before being discovered or fixed by the vendor or provider of the software, hardware, or solution.
- Lack of encryption is caused by the absence of encryption practices that allow easy access to confidential data once they access the data store or system.
- Access vulnerabilities are related to identity or access that grants access to unauthorized users.

We will now explore the key areas of vulnerability management, from understanding their lifecycle to various processes and techniques for identifying, managing, and addressing them to stay fully protected.

Vulnerability management lifecycle

Vulnerability management is the continuous process that enables organizations to systematically identify, prioritize, and mitigate vulnerabilities in information security. Vulnerabilities serve as an entry point for attacks and cause much harm if left unattended. Equally, as the infrastructure estate expands, it can be challenging for organizations to stay

on top of vulnerabilities across the enterprise. It is thus essential to understand the implications of these vulnerabilities and prioritize how and when they are addressed. *Figure 7.1* shows the key stages in the lifecycle:



Figure 7.1: Vulnerability management lifecycle

The key to successful vulnerability management is to ensure early identification, regular assessments, compliance, and cost efficiency through automation where possible. *Table 7.1* describes the key stages of the lifecycle:

Stage	Description
Discover	The first stage of the lifecycle is to identify and discover all assets and critical systems within the organization. These are to be categorized and prioritized based on their criticality, data classification, and potential threats. This becomes the overall scope and generates an inventory of all assets that need to be protected. Once you have a holistic inventory, you should identify vulnerabilities in these systems through the use of vulnerability scanners across the network, applications, and systems, and manual tests and audits such as penetration testing. The outcome is a full set of assets and their associated vulnerabilities that are currently known.
Prioritize	A typical organization will have hundreds, if not thousands, of assets, each with multiple known vulnerabilities. The next important step is to prioritize them to help focus the remediation that will follow. Priorities will be based on potential impact to business, likelihood, and legal and compliance implications. This is where frameworks such as Common Vulnerability Scoring System (CVSS) come in

	handy, although they still need to be applied in the organization's context. Prioritization plays a critical role in assigning resources and remediating the vulnerabilities to bring the risk within the organization's risk appetite.
Remediate	Once priorities are clear, you can get down to action in order of priority. Actions can be one of the following: • Remediate or treat is resolving the problem at the source. This is the most likely outcome for most vulnerabilities. • Mitigate by applying compensatory controls; typically, this means you apply internal controls, such as limiting access and segregation of duties. • Accept or tolerate the risk associated with the vulnerability in case there is no action possible or the cost of taking an action is disproportionate to the potential benefit. When accepting the risks, they need to be surfaced at the appropriate level within the organization, depending on impact and likelihood.
Validate	Once all critical vulnerabilities have been addressed, you need to validate and verify the resolution. There are multiple ways this can be performed, such as using internal monitoring systems and vulnerability scanners where the vulnerability was reported, internal testing, external testing, and audits. This stage is not necessarily a one-time activity but may have to be repeated regularly based on the criticality of the asset or system; for instance, penetration tests may be performed annually for critical customer-facing systems.
Report	Once validated, a summary of the status of vulnerabilities is to be generated again, classifying them based on impact, probability, and the type of action plan. This is a crucial part of accepting any vulnerabilities that cannot be addressed and requires thorough review and justification, especially for the ones that need to be accepted or tolerated.
Monitor and improve	The last and essential part of the cycle is to monitor and improve the entire process, starting with the discovery of the vulnerabilities through to resolution and reporting. The focus is on the areas of tools and processes that can improve the process and make it faster and effective.

Table 7.1: Vulnerability management lifecycle stages

Real-world relevance

Let us use the example of the Apache Struts vulnerability (CVE-2017-5638) we saw in the Equifax case study at the beginning of the chapter:

- **Discover:** Ideally, such a vulnerability would come up in multiple areas, from vulnerability management tools to **continuous integration/continuous deployment (CI/CD)** pipelines. Being a critical vulnerability, it would also be broadly reported on most forums, and one can understand this through subscriptions to security forums, CVE, and exposure databases. Once you are aware of the vulnerability,

it is essential to understand all impacted assets. Ideally, the asset inventory would already be available within the organization, and the use of Apache Struts would be known. If not, the vulnerability scanning tools can provide insight into impacted areas.

- **Prioritize:** The CVE had a base score of 9.8, making it a critical vulnerability that must be addressed. The next aspect would be reviewing the impacted software systems. In the case of Equifax, although Struts was used across many systems, the specific system in question was the customer dispute portal, which seemingly does not raise any alarms, but the fact that it could enable access to a server with unencrypted credentials should have been known. This would increase the priority of patching this vulnerability.
- **Remediate:** Given the vulnerability's impact and urgency, this specific case would require remediation by assembling the team to resolve the issue and test any associated issues. Mitigating or accepting the vulnerability risk would not have been an option.
- **Validate:** The next step is to validate that the fix works. This would mean a penetration test to ensure the vulnerability has been resolved and does not allow attackers to execute commands through file-upload attempts.
- **Report:** The latest status of the fix would then be reported internally and externally.
- **Monitor and improve:** The final step is to ensure the monitoring is in place to catch any intrusions or anomalies. Lastly, any learnings from this experience should be captured to resolve any issues with the process or the tools. This would potentially mean resolving the certificate issue to help identify any potential attacks.

Zero-day vulnerability

Sometimes, vulnerabilities have been disclosed but not patched, either because they are discovered before researchers or developers are aware, or before a patch can be issued. These are the most dangerous attacks and pose a higher risk as they can be exploited. The MOVEit transfer service exploits we saw in [Chapter 3, Secure Your Code, Build and Deploy](#), were a classic example of a zero-day vulnerability affecting many organizations across

multiple countries. Until the fix for a zero-day exploit is released, organizations can either mitigate the risk by deploying compensatory controls or accept the vulnerability risk.

Discover

This subsection will cover various ways of discovering assets and vulnerabilities across an organization's information technology estate. The first steps are discovering your assets, creating an inventory, and securing their configuration. This is followed by protecting them against potential weaknesses, which requires identifying vulnerabilities and environmental threats. This requires scanning for vulnerabilities and comprises many different techniques and solutions. There are two types of vulnerability scanning solutions:

- Application scanning (or software) solutions are typically used to scan for vulnerabilities in the software being built and allow you to stop the build and deployment of the software. Most of these were covered in [Chapter 3, Secure Your Code, Build and Deploy](#), but will be discussed in a bit more detail, and they include **software composition analysis (SCA)**, **open-source software (OSS)**, and **static application security testing (SAST)**.
- Runtime protection scans for vulnerabilities in running software and its configuration.

Asset discovery and inventory management

Asset discovery is a foundational part of effective management and cybersecurity, and it helps identify the full scope of an organization's cybersecurity landscape. The asset inventory generated based on the discovery further enables the identification of threats and vulnerabilities in those assets. The devices connected to the organization's network have grown exponentially with the increase in remote work and the use of personal and mobile devices. Hence, the traditional manual asset management is no longer scalable. Automated asset discovery tools (such as *SolarWinds*, *Rapid7*, *CloudQuery*, and *Asset Panda*) streamline identifying and cataloging all hardware, software, and network-connected devices within an organization's infrastructure. These tools use active techniques,

such as network scanning and device probing, and passive methods, like monitoring network traffic, to detect assets.

Once discovered, detailed information such as IP addresses, operating systems, installed applications, and device configurations is collected and organized into a centralized inventory system. Real-time asset discovery ensures continuous visibility into the IT environment, enabling organizations to track changes, identify vulnerabilities, and manage risks effectively. Once discovered, it is also crucial to integrate it with other IT management systems, such as **Configuration Management Databases (CMDBs)** or ticketing platforms; automated asset discovery enhances operational efficiency, reduces manual errors, and supports compliance with regulatory standards. A proactive approach is critical for addressing dynamic attack surfaces and safeguarding against threats posed by shadow IT or unapproved devices.

Assets identified in the inventory can be classified into a few common types listed in the following:

- Hardware assets are physical, tangible components such as desktops, laptops, mobile devices, and peripherals such as printers, scanners, network equipment, and storage devices.
- Software assets include software applications such as operating systems (Windows, Linux, and Mac), productivity software such as Microsoft Office, and specialized enterprise resource planning or customer relationship management systems.
- Cloud assets are services or resources accessed over the Internet, such as Microsoft OneDrive. *AWS*, *Azure*, *Google*, and **software as a service (SaaS)** applications like Salesforce offer cloud computing. Many assets today fall into this category.
- Digital assets are databases and repositories on devices that store customer data, financial records, and employee data.

Figure 7.2 shows an example of a CloudQuery console with resources EC2 (virtual machines) across various teams:

CLOUD	TYPE	ACCOUNT	NAME	REGION
aws	EC2 Instance	12345678	finance-prod	eu-north-1
aws	EC2 Instance	12345678	finance-staging	us-east-1
aws	EC2 Instance	12345678	finance-staging	eu-north-1
aws	EC2 Instance	12345678	finance-prod	eu-west-1
aws	EC2 Instance	12345678	api-staging	us-east-2
aws	EC2 Instance	12345678	api-staging	sa-east-1
aws	EC2 Instance	12345678	api-prod	eu-central-1
aws	EC2 Instance	12345678	api-prod	ap-northeast-3
aws	EC2 Instance	12345678	api-prod	ap-northeast-3

Figure 7.2: Cloud asset inventory (CloudQuery)^[10]

Configuration management

With a consolidated inventory of assets, the next step is to ensure they are configured correctly and securely. The CapitalOne case study shows how a misconfiguration can enable attackers to access critical data. **Configuration management (CM)** systematically manages, maintains, and documents the configuration of systems, applications, networks, and other IT assets. It ensures that all components in an environment are set up according to predefined standards and remain consistent over time. CM involves tracking **configuration items (CIs)**, monitoring changes, and enforcing baseline configurations to maintain stability, security, and compliance. Configuration management is critical for reducing misconfiguration risks, which are a leading cause of vulnerabilities. It enables organizations to identify deviations from secure baselines and rectify them promptly.

Configuration management aims to reduce the likelihood of misconfigurations and configuration drift. Misconfigurations occur due to default passwords, default settings, open ports, or weak encryption settings that can expose systems to attacks; configuration risk typically occurs due to unauthorized or undocumented changes that can lead to vulnerabilities that go unnoticed until exploited.

Real-world relevance

One of the benchmarks that can be followed by teams that are globally recognized security guidelines is the CIS Benchmarks developed by the CIS, which provide prescriptive recommendations for securely configuring operating systems (Linux, Windows), cloud platforms (e.g., AWS, Azure), network devices, and applications and databases (Kubernetes, PostgreSQL, Oracle, NGINX, etc.).

Follow these steps for a secure configuration baseline for your software applications:

1. When starting to configure software and applications, look for standards and benchmarks as they provide the following benefits:
 - **Security hardening:** Reduce attack surfaces by eliminating insecure settings.
 - **Compliance alignment:** Map to frameworks like **National Institute of Standards and Technology Cybersecurity Framework (NIST CSF)**, **Payment Card Industry Data Security Standard (PCI DSS)**, and **Health Insurance Portability and Accountability Act (HIPAA)**.
 - **Operational efficiency:** Balance robust security with minimal impact on performance.
2. During implementation, the configurations will be automated based on these benchmarks using infrastructure as code, which we learned in [Chapter 3, Secure Your Code, Build and Deploy](#). This eliminates manual errors. If multiple baseline levels exist, apply level 1 profiles for basic security requirements or level 2 for high-security environments.
3. Use tools for assessing your benchmarks, such as the **CIS Configuration Assessment Tool (CIS-CAT)** or Vulnerability Manager Plus, to automate audits against CIS Benchmarks. Depending on the software you use, you may have to use additional tools; for instance, for Kubernetes, tools such as kube-bench assess CIS your configurations against benchmarks.
4. Regularly update configurations based on new versions of CIS Benchmarks.

[Figure 7.3](#) shows a sample dashboard from the AWS Security Hub service;

once configured, this shows all the standard controls for a secure baseline across your AWS account. The scorecard shows the percentage of controls passing, and the list in the following shows all the failed controls, along with their severity:

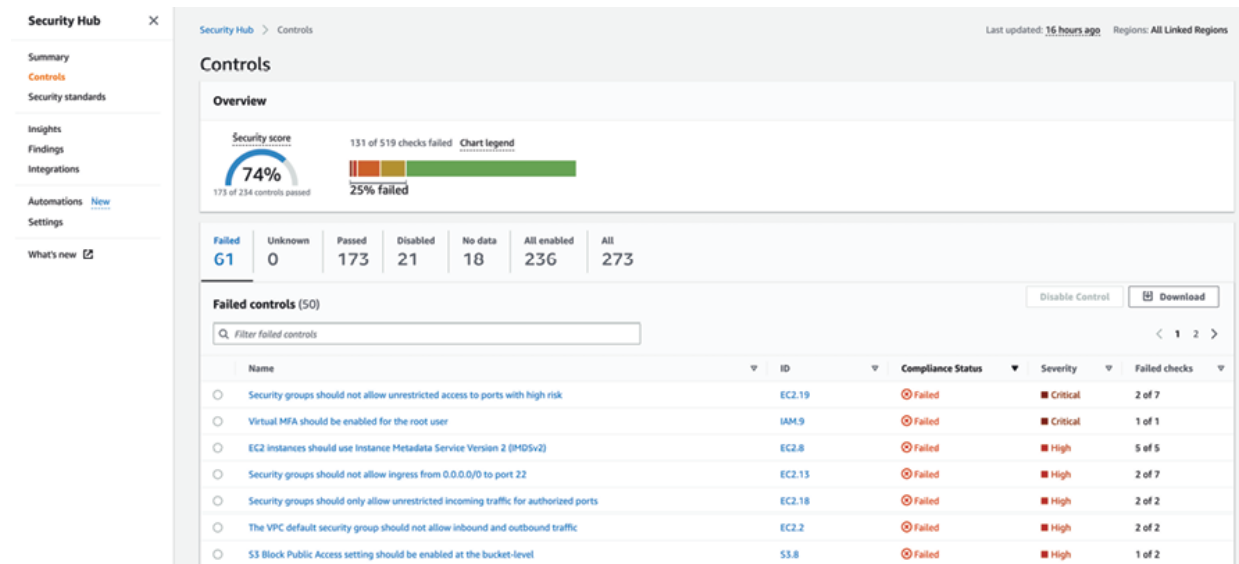


Figure 7.3: AWS Security Hub baseline controls^[1]

Software composition analysis

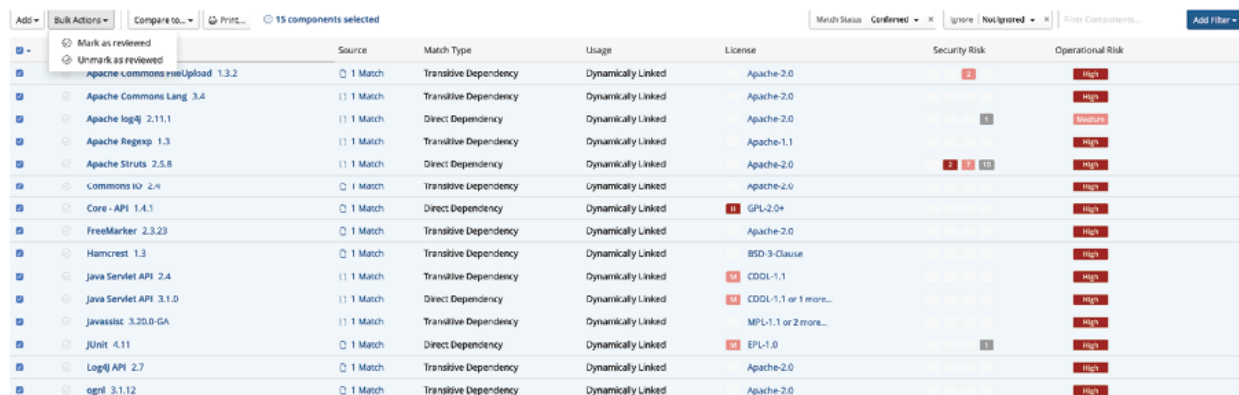
Software composition analysis (SCA) scanning tools analyze all software components and their dependencies to identify potential vulnerabilities and compliance issues. They analyze software dependencies and compare them against databases of known vulnerabilities to highlight potential risks. By integrating with asset inventory systems, SCA tools can ensure that all software components are regularly scanned and monitored for emerging threats. They typically offer the following features:

- **Identify components:** The tools create a SBOM by scanning the application's build artifacts, dependencies, and metadata. For instance, in the case of Java applications, these tools will look at the Gradle or Maven build config and analyze the generated **Java Archive (JAR)**.
- **Detect vulnerabilities:** SCA then compares the dependencies against a database of known vulnerabilities to identify known vulnerabilities in the libraries and open-source components. Full vulnerability details (such as CVE) are provided, helping developers and security teams

address potential risks.

- **Reporting and remediation:** SCA tools provide reports that highlight vulnerabilities, license issues, and potential risks, enabling organizations to prioritize remediation efforts.
- **Policies and risk management:** The tools also help define policies for tolerance and quality gates around vulnerability types and levels, enabling organizations to manage risks associated with open-source software proactively.
- **Supply chain security:** SCA plays a crucial role in securing the software supply chain by identifying and mitigating vulnerabilities in third-party components.

Some examples of tools are GitHub Advanced Security, Snyk, JFrog Xray, BlackDuck. *Figure 7.4* shows the Black Duck, which shows the various dependencies, security, license risks (OSS scan in the following figure), and operational risks:



	Source	Match Type	Usage	License	Security Risk	Operational Risk
☐ Mark as reviewed ☐ Unmark as reviewed	Apache Commons FileUpload 1.3.2	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High
☐	Apache Commons Lang 3.4	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High
☐	Apache log4j 2.11.1	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	Medium
☐	Apache Regexp 1.3	1 Match	Transitive Dependency	Dynamically Linked	Apache-1.1	High
☐	Apache Struts 2.5.8	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	High
☐	Commons IO 2.4	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High
☐	Core - API 1.4.1	1 Match	Direct Dependency	Dynamically Linked	GPL-2.0+	High
☐	FreeMarker 2.3.23	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High
☐	Hamcrest 1.3	1 Match	Transitive Dependency	Dynamically Linked	BSD-3-Clause	High
☐	Java Servlet API 2.4	1 Match	Transitive Dependency	Dynamically Linked	CDL-1.1	High
☐	Java Servlet API 3.1.0	1 Match	Direct Dependency	Dynamically Linked	CDL-1.1 or 1 more...	High
☐	javassist 3.20.0-GA	1 Match	Transitive Dependency	Dynamically Linked	MPL-1.1 or 2 more...	High
☐	JUnit 4.11	1 Match	Direct Dependency	Dynamically Linked	EPL-1.0	High
☐	Log4j API 2.7	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High
☐	ogni 3.1.12	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High

Figure 7.4: SCA and OSS Scans Black Duck^[12]

Open-source software

Most software built today requires open-source components, libraries, and tools. However, not all open-source licenses are the same; some can be more restrictive when used in enterprise products and solutions. Some licenses, for instance, require that any derived work be distributed under the same license. Hence, license compliance ensures the software meets all legal requirements. Typically, this feature is part of most SCA tools and helps organizations understand and comply with the licenses of their open-source

components, preventing legal issues. The key features include understanding all open-source licenses in use, defining policies around using approved license types, and monitoring license compliance. Some examples of tools include *OpenChain*, *IFrog*, *BlackDuck*.

Static application security testing

SAST analyzes source code, binaries, or bytecode to identify vulnerabilities. It is a white box testing method that does not execute the code. It is typically used during development to detect issues such as SQL injection, **cross-site scripting (XSS)**, etc. These tools usually use techniques such as pattern matching to detect insecure coding patterns, analyze data flows based on how data moves through the application, and control flows using loops and conditionals. Some solutions also allow the creation of custom rules. Some examples of tools are *Sonar*, *CodeQL*, *HCL AppScan*, *Checkmarx*, *Checkmarx*, and *Fortify*. [Figure 7.5](#) shows a sample finding done by Snyk based on static code analysis:

The screenshot displays the Snyk Code Analysis interface for a project named 'snyk/snyk-goof'. The interface includes a sidebar with filters for SEVERITY, PRIORITY SCORE, STATUS, LANGUAGES, and VULNERABILITY TYPES. The main area shows a search bar and a list of 11 issues. The selected issue is a 'NoSQL Injection' vulnerability with a score of 828. The vulnerability is located in the file 'routes/index.js' at line 39. The code snippet shows a function 'exports.admin' that uses 'User.find' with unsanitized input from the HTTP request body. The description states: 'Unsanitized input from the HTTP request body flows into find, where it is used in an NoSQL query. This may result in an NoSQL Injection vulnerability.' The interface also includes a 'Learn about this type of vulnerability and how to fix it' link and buttons for 'Ignore' and 'Full details'.

Figure 7.5: Snyk code vulnerability example[13]

Dynamic application security testing

Dynamic application security testing (DAST), on the other hand, is a black box testing method that tests running applications as it evaluates runtime behavior by simulating attacks. It is used to identify vulnerabilities such as authentication flaws, insecure configurations, and insufficient controls. These tools typically interact with the system as an external user, simulate attacks using malicious payloads, and analyze responses to detect weaknesses. Some examples include *Zed Attack Proxy (ZAP)*, *Burp Suite*, *Acutenix*, *Veracode*, *Rapid7*, and *HCL AppScan*. These tools analyze the application response under simulated attacks to uncover its weaknesses, such as:

- Input and output validation errors
- Server or infrastructure configuration issues
- Authentication and session management issues
- Data exposure issues

In this way, these tools mimic external attacks and provide a realistic assessment of the security posture of the application, allowing you to detect real vulnerabilities and provide comprehensive coverage. These tools can all be integrated into your deployment pipelines and delivery workflows.

Network discovery

Network discovery identifies and maps all devices and systems connected to an organization's network. This process is crucial for maintaining an up-to-date asset inventory and ensuring comprehensive vulnerability scanning coverage. Advanced network discovery tools can detect managed and unmanaged devices, providing visibility into potential security blind spots, shadow IT devices, and outdated systems. These tools work by discovering devices connected to the network, mapping running services and applications, and detecting any vulnerabilities, unpatched systems, or misconfigurations. Some examples are *Nmap*, *SolarWinds*, and *Tenable Nessus*.

Prioritize and assess

Large organizations have complex and massive infrastructure estates with

hundreds, if not thousands, of applications and vulnerabilities that various tools will continuously discover. Not all vulnerabilities can be addressed simultaneously, nor are they all equally important. It is crucial to optimize resource allocation, reduce risk, protect critical assets, and comply with regulations. This subsection will cover how to prioritize and assess identified vulnerabilities. Most of the tools and solutions that help discover these vulnerabilities will automatically do this for you. Still, it is always good to have a sufficient understanding of how to decide what matters. We will discuss some assessment frameworks and impact analysis that contribute to risk assessment by helping organizations understand their software and systems' potential vulnerabilities and threats that feed into prioritization and evaluation.

Asset criticality

Asset criticality is the importance of an asset in supporting business operations, safety, compliance, and overall success. The criticality score ranks assets based on their impact on production services, revenue loss, or safety risks; this helps ensure high-priority assets and equipment receive frequent maintenance or immediate attention during failures. Assets are typically tracked in the asset inventory system we saw previously in this chapter. These assets should be linked to a criticality that can then be used to prioritize vulnerabilities identified against them.

Common Vulnerability Scoring System

The **Common Vulnerability Scoring System (CVSS)** is a standardized framework for assessing the severity of security vulnerabilities. We discussed this in [Chapter 3, Secure Your Code, Build and Deploy](#). It is a standardized framework used to assess and quantify the severity of security vulnerabilities, providing a numerical score that helps organizations prioritize risks. CVSS evaluates vulnerabilities based on several metrics, including exploitability, impact on confidentiality, integrity, availability, and user interaction required for an attack, and then assigns numerical scores (0 to 10). This scoring system allows security teams to objectively assess the risks associated with various vulnerabilities and prioritize them according to their potential impact on the organization's assets. The system is based on

three metrics^[14]:

- **Base metrics:** Evaluate intrinsic properties related to exploitability and system and subsequent system impact, such as attack vector, complexity, privileges required, and impact on confidentiality, integrity, and availability.
- **Threat metrics:** Reflect vulnerability related to a threat that may change over time, such as proof that the vulnerability has not been exploited. Values in this category can change over time.
- **Environmental metrics:** Tailors scores to reflect the criticality of assets within an organization's specific environment. This includes considerations such as the presence of a security control that may mitigate some or all consequences of a successful attack.

Table 7.2 summarizes the CVSS score ratings and their implications:

CVSS score	Rating	Implication
9.0 to 10.0	Critical	Immediate remediation is recommended as the exploit can allow attackers to gain complete control over systems or networks. CVE-CVE-2017-5638 with Apache Struts that affected DP World Australia.
7.0 to 8.9	High	Although not urgent, these are critical issues and still pose a significant risk and should be addressed promptly. CVE-2022-22965 (Spring4Shell) in Java Spring Framework enables RCE under specific deployment conditions on Java Development Kit (JDK) 9+.
4.0 to 6.9	Medium	Less likely to be exploited, but can still lead to moderate damage if left unaddressed.
0.1 to 3.9	Low	These pose minimal risk, and unless they affect high-value assets, they can be deprioritized.
0.0	None	No exploitable impact or risk identified unless the environmental factors change the risk factor.

Table 7.2: CVSS scores and ratings

Business impact analysis

The measures we have seen show only some attributes associated with vulnerability risk, but understanding the impact of data breaches and incidents is essential. **Business impact analysis (BIA)** helps simplify

decision-making in risk management processes by identifying the potential consequences of disruptions to critical business functions. It evaluates tangible and intangible assets, processes, personnel, and data pathways to determine their importance. It helps adjust the external demands and the impact of vulnerabilities with changes to the market and conditions, and internal drivers such as costs and people.

BIA typically involves the steps listed in *Table 7.3*:

Step	Details
Goals and scope	Define the objectives and scope of the business impact analysis, for instance, which departments or systems are part of the scope.
Critical business functions and assets	Within the scope, identify products, services, and business processes that are essential to the successful operation of the organization. Based on this scope, identify the assets that enable those functions.
Analyze dependencies	Map interdependencies between assets, suppliers, and processes to identify single points of failure and the impact of vulnerabilities across systems.
Assess potential impacts	Quantifies the financial and operational impact of losing specific assets to ensure organizations focus on protecting systems vital for operations and survival. Impact could be financial loss due to the impact on revenue, operational disruption, and productivity loss, reputational damage, and non-compliance with regulations, leading to penalties.
Continuous improvement	Periodically review the impact analysis process to account for changes in business processes, technology, or external threats.

Table 7.3: Business impact analysis

There are a few prioritization techniques that are integral parts of business impact analysis; two of the common ones are listed as follows:

- **Contextual prioritization** involves assessing vulnerabilities within the unique context of an organization's environment. It considers factors such as the business criticality of affected systems, third-party dependencies, and regulatory requirements. Asset criticality, business impact, data sensitivity, exploitability, threat intelligence, incident history, operational dependencies.
- The **risk-based prioritization** approach focuses on the likelihood and impact of the risks by combining vulnerability severity (e.g., CVSS scores) with contextual factors like business impact and exposure level. It balances technical severity with organizational priorities for efficient risk mitigation. For instance, a vulnerability with a lower CVSS score

may be prioritized if it affects a high-critical asset and could have a higher impact.

Risk matrices are a good way to visually represent risks based on likelihood and impact. They help organizations categorize vulnerabilities into low, medium, high, or critical risk levels, as shown in [Table 7.4](#), and help prioritize. The matrix in the following is typically used in risk-based prioritization, which combines likelihood vs. impact:

		Impact				
		Negligible	Minor	Moderate	Significant	Severe
Likelihood	Very likely	Medium	High	High	Very high	Very high
	Likely	Medium	Medium	High	Very high	Very high
	Possible	Low	Medium	Medium	High	Very high
	Unlikely	Low	Low	Medium	High	High
	Rare	Low	Low	Low	Medium	High

Table 7.4: Sample risk matrix

A few more prioritization and assessment terms, listed in the following, are critical to understand when analyzing vulnerabilities:

- **Exposure assessment** evaluates an organization's vulnerability to external threats based on attack surface size and accessibility. The aim is to reduce the attack surface by addressing exposed entry points. It is a continuous process to help quantify and prioritize vulnerabilities, misconfigurations, and security risks across the organization's digital landscape. For instance, it includes poorly configured or end-of-life systems, in addition to detected vulnerabilities.
- **Risk appetite** is the level of risk an organization is willing to accept to pursue its objectives. It reflects the organization's strategic priorities and defines boundaries for acceptable risks. Risk appetite determines which vulnerabilities to prioritize based on their alignment with organizational goals.
- **Risk tolerance** is the degree of variability or uncertainty an organization can handle within its risk appetite. It is more granular than risk appetite and often tied to operational thresholds. It defines

acceptable downtime or data loss for critical business functions during disruptions and establishes thresholds for remediation timelines (e.g., fixing critical vulnerabilities within five days).

- **Residual risk** refers to the level of risk that remains after implementing all possible mitigation measures. It acknowledges that no system can be completely risk-free. Residual risk helps understand the lingering risks that could impact critical business functions even after applying safeguards. Legacy systems may retain residual risks due to unpatched vulnerabilities despite mitigation efforts. It drives continuous monitoring and reassessment of security controls to address evolving threats.

Remediate

Once vulnerabilities are prioritized, it is time to remediate, that is, address the vulnerability to help protect the organization's assets and maintain compliance. It is made of multiple steps outlined as follows:

1. **Remediation planning** is the first activity that establishes structured processes and controls, laying the groundwork for successful risk reduction.
2. **Patch management** is a key component of the remediation process. It addresses identified vulnerabilities by applying necessary updates and fixes to address known vulnerabilities.
3. **Implement the alternative strategies if required:**
 - **Mitigation strategies** involve temporary or alternative measures to reduce risk when immediate patching is not feasible, as a fix may not be available.
 - **Compensating controls** are security measures, such as network segmentation or enhanced monitoring, that offset risks when direct remediation is impractical.
4. **Change management** is the formal process of reviewing, approving, and documenting remediation actions, minimizing operational disruptions.
5. **Testing procedures** involve rigorously validating patches and configuration changes in controlled environments, which ensures they

do not introduce new issues.

6. **Rollback planning** is the preparation of contingency plans to revert changes if remediation efforts cause unintended consequences.

The implementation process translates these plans into action, focusing on the technical execution of remediation tasks. It includes a few key activities:

- Configuration hardening adjusts system and application settings to reduce attack surfaces and enforce security best practices.
- System updates focus on upgrading operating systems, applications, and firmware to supported and secure versions.
- Patch deployment is applying updates across affected systems in a timely and coordinated manner.
- Dependency management identifies and updates vulnerable libraries, plugins, and third-party components.
- Version control is maintaining records of changes and ensuring the ability to track, audit, and revert to previous states if necessary.

Validate

Once patches and fixes are implemented, they must be verified and tested. This includes security and regression testing and runtime protection, verifying that vulnerabilities have been successfully addressed and that systems remain secure and compliant during operation. Post-remediation is a standard process where testing is conducted to confirm that vulnerabilities have been properly fixed and that no new issues have been introduced during the remediation process. It spans multiple activities:

- Performance testing assesses whether systems continue to operate efficiently after security changes, ensuring that remediation actions do not negatively impact business operations or user experience.
- Regression testing is essential to confirm that previously functioning features remain intact after updates or patches, preventing the recurrence of old vulnerabilities or introducing new ones.
- Security validation involves systematically testing controls and configurations to ensure they function as intended and provide the expected level of protection.
- Penetration testing simulates real-world attack scenarios to validate the

effectiveness of security measures and uncover any residual or hidden vulnerabilities that automated tools might miss. It provides the most comprehensive assurance that vulnerability management processes are robust, effective, and aligned with organizational objectives.

- Configuration audits assess systems and applications against established security baselines and best practices. Linking configuration audits to asset inventory can ensure that all assets are configured securely and consistently, helping identify misconfigurations that could lead to vulnerabilities, even without software flaws.
- Compliance checking is performed to verify that all security measures align with relevant regulatory requirements, organizational policies, and industry standards, helping organizations maintain audit readiness and avoid penalties.
- Generate a remediation report detailing the vulnerability, actions taken, test results, and compliance status for audit and stakeholder review.

Regular security assessments, including penetration testing and vulnerability scans, should be performed to identify and address potential weaknesses in an organization's infrastructure and application workloads. Depending on the application's criticality and impact on business, either annual or more frequent tests should be performed to ensure compliance.

As organizations move applications into dynamic, cloud-native, modern, container, serverless, and microservices-based architectures, static and pre-deployment security controls alone cannot address emerging threats. As a result, runtime protection is essential in the validation phase of vulnerability management because it provides continuous, real-time monitoring and defense against active threats and vulnerabilities that may only become apparent when applications, containers, or workloads are running. Runtime protection solutions monitor the behavior of applications and systems as they operate in a live environment, allowing them to block malicious or anomalous behavior. Some examples include *Falco*, *Sysdig Secure*, and *Aqua Security*. **Cloud Native Application Protection Platforms (CNAPP)**, which we will discuss later, also offer runtime protection.

Report

Reporting is essential to ensuring a robust security posture. Comprehensive

documentation and reporting processes ensure that every stage of the vulnerability management lifecycle, from discovery and prioritization to remediation and validation, is meticulously recorded and communicated. This encompasses the following elements that create a holistic reporting framework supporting operational security and aligning with business objectives and regulatory requirements:

- Configuration management is critical in tracking asset changes and addressing vulnerabilities in evolving system configurations.
- Vulnerability tracking systems maintain detailed records of identified issues, remediation status, and recurring patterns, supporting operational efficiency and historical analysis.
- Metrics and **key performance indicators (KPIs)** such as mean time to remediation, risk scores, and asset coverage are essential for measuring the effectiveness of vulnerability management efforts and driving continuous improvement.
- Trend analysis leverages historical data to identify recurring vulnerabilities and predict future risks, informing proactive security strategies.
- Compliance reporting automates the generation of audit-ready documentation, demonstrating adherence to regulatory standards like **Payment Card Industry Data Security Standard (PCI DSS)** and **Global Data Protection Regulation (GDPR)**, and streamlining the audit process.
- Executive dashboards offer real-time, visual summaries of the organization's security posture, remediation progress, and key risk indicators, enabling informed decision-making at the leadership level.
- Audit trails record all vulnerability management activities, supporting accountability, forensic investigations, and regulatory compliance by offering transparent evidence of security controls and process adherence.

Monitor and improve

This subsection will cover some solutions for continuously monitoring vulnerabilities and improving the management lifecycle. Some of these

solutions are excellent discovery tools as well. The three solutions covered here play a critical role in enforcing policies and procedures, performing automated and advanced monitoring of vulnerabilities, and building decoy systems to gather intelligence on threats. Combining these techniques can help provide a robust, multi-layered vulnerability management strategy. However, this is only possible with the continuous improvement mindset and learning from the findings provided by these solutions.

Cloud Security Posture Management

As the CapitalOne case study shows, misconfiguration detection and remediation are extremely important. **Cloud Security Posture Management (CSPM)** tools cover infrastructure-level security configurations by comparing them against secure baselines, which help detect resource misconfigurations. Solutions in this space continuously monitor cloud environments to ensure adherence to security best practices and regulatory standards. These tools typically do not have any runtime capabilities (that is, looking at running applications) and primarily focus on infrastructure compliance. However, some also include workload and application protection capabilities, which are covered in the following two sections.

Real-world relevance

Some examples include *SentinelOne*, *Check Point CloudGuard*, *AWS Security Hub*, *Azure Security Center*, and *Policy*. The following is how the cloud provider services perform posture management:

- AWS Security Hub also performs automated checks on cloud infrastructure security configurations against benchmarks, identifying misconfigurations and compliance risks. AWS Security Hub was discussed under the *Configuration management* section; [Figure 7.4](#) shows the dashboard. [Figure 7.6](#) shows the various security standards it supports when setting up a Security Hub. In addition to AWS Standard foundational security practices, it also includes the CIS Benchmarks, NIST publication, and PCI DSS. Once these are set up and configured, the dashboard shows compliance and security posture, as shown in [Figure 7.4](#). The tool also allows alerting capability and automated remediation actions.

Security standards

Enabling AWS Security Hub grants it permissions to conduct security checks. [Service Linked Roles \(SLRs\)](#) with the following services are used to conduct security checks: Amazon CloudWatch, Amazon SNS, AWS Config, and AWS CloudTrail.

- ☒ Enable AWS Foundational Security Best Practices v1.0.0
- ☐ Enable AWS Resource Tagging Standard v1.0.0
- ☐ Enable CIS AWS Foundations Benchmark v1.2.0
- ☐ Enable CIS AWS Foundations Benchmark v1.4.0
- ☐ Enable CIS AWS Foundations Benchmark v3.0.0
- ☐ Enable NIST Special Publication 800-53 Revision 5
- ☐ Enable PCI DSS v3.2.1
- ☐ Enable PCI DSS v4.0.1

AWS Integrations

Enabling Security Hub grants it permissions to import findings from AWS services that you have enabled.

[Learn more](#)

Figure 7.6: AWS Security Hub standards

- Azure Policy enforces security baselines and compliance standards (based on CIS Benchmarks and NIST) across Azure resources. It forms the backbone of CSPM capabilities by ensuring resources adhere to organizational policies and regulatory requirements. [Figure 7.7](#) shows a sample Azure Policy dashboard that shows compliance against these policies:

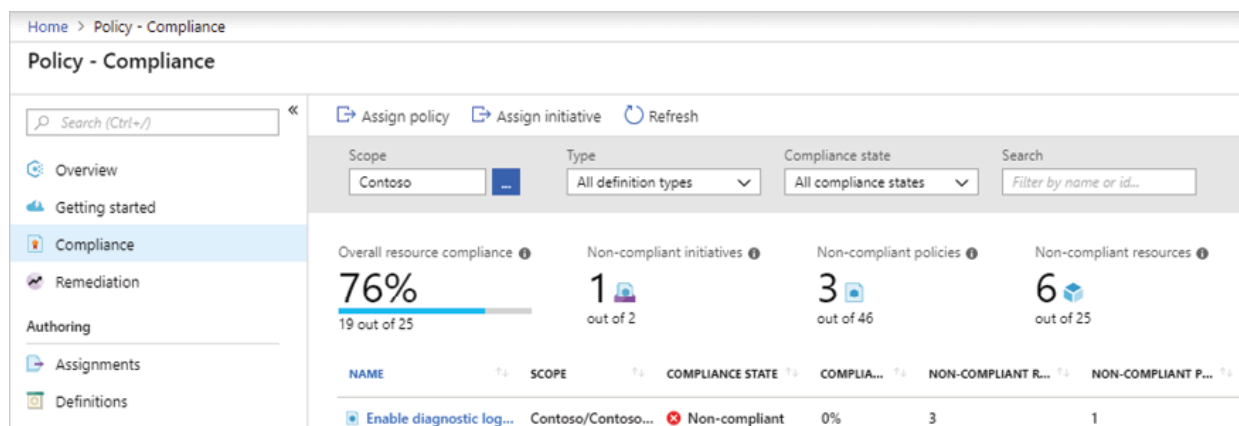


Figure 7.7: Azure policy dashboard^[15]

- Google Cloud Security Command Center also includes CSPM

capabilities for managing risks and ensuring compliance across multi-cloud environments.

Cloud Workload Protection Platform

Cloud Workload Protection Platforms (CWPPs) are essential security solutions that safeguard cloud-based workloads such as virtual machines, containers, and serverless functions. In a Zero Trust network architecture, where implicit trust is eliminated, CWPPs play a crucial role by continuously monitoring workloads, enforcing least privilege access, and applying runtime security controls. These platforms leverage behavioral analytics, identity-based policies, and micro-segmentation to detect and prevent unauthorized access, malware, and misconfigurations. CWPPs provide consistent visibility and threat protection across multi-cloud and hybrid environments, ensuring that workloads remain secure regardless of location or underlying infrastructure.

Real-world relevance

Some solutions that offer CWPP include Palo Alto Prisma Cloud and CloudStrike Falcon Cloud. In addition, cloud providers offer some native services, such as Inspector and GuardDuty from AWS and Microsoft Defender for Cloud from Azure. These tools provide holistic security across applications and infrastructure through scanning and runtime protection of workloads. Here is a quick overview of the scope they cover:

- Amazon Inspector automatically scans workloads like EC2 instances, Lambda functions, and container images for software vulnerabilities and network exposure and provides contextual risk scores using CVE data to prioritize remediation. AWS GuardDuty, on the other hand, focuses on anomaly detection and threat prioritization to help detect malicious activity and unauthorized behavior.
- Microsoft Defender for Cloud is a holistic solution across CSPPM and CWPP. Workload protection includes scanning Azure SQL databases for misconfigurations, excessive permissions, and unprotected sensitive data. It provides actionable remediation steps based on Microsoft's best practices. [Figure 7.8](#) shows the workload protection dashboard. It shows vulnerabilities across many different resources, including servers, Kubernetes, and Azure SQL databases; alerts are then generated and

prioritized as high, medium, and low severity:

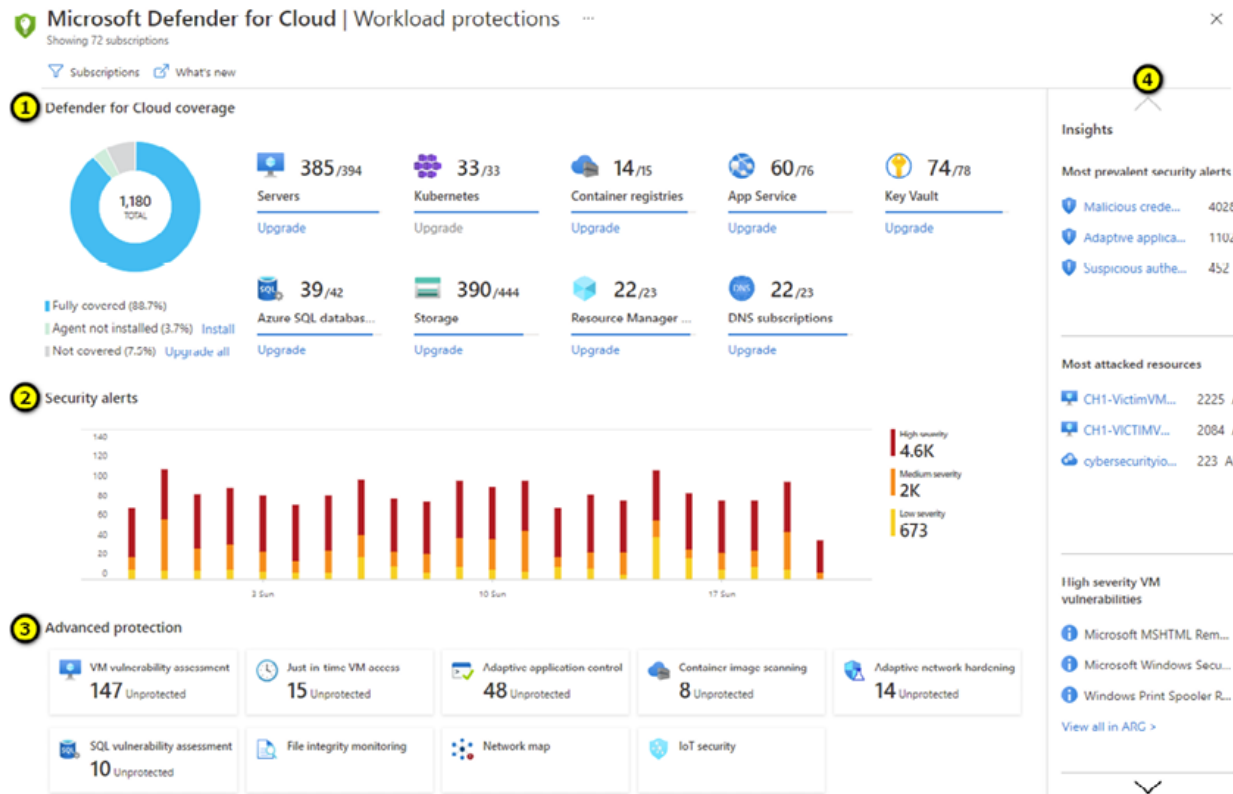


Figure 7.8: Microsoft Defender for Cloud^[16]

Honeypots

Honeypots are decoy systems that attract attackers, allowing security teams to monitor their behavior, identify attack methods, and gain valuable intelligence on emerging threats. They help prioritize and address vulnerabilities in the real system; however, they should not be solely relied upon for vulnerability management and must be used as part of a comprehensive security strategy. The following is a list of some of the benefits of using honeypots:

- **Early detection:** Honeypots can detect attacks early by luring attackers to a seemingly vulnerable system, giving security teams time to respond before real damage occurs.
- **Attacker behavior analysis:** By monitoring attacker activity on the honeypot, security teams can gain insights into attacker tactics, techniques, and tools.

- **Threat intelligence gathering:** Honeypots can provide valuable intelligence by revealing new attack methods and trends, enabling proactive security improvements.
- **Prioritization of vulnerabilities:** Security teams can prioritize patching by understanding which vulnerabilities are most targeted by attackers.

Important considerations when using honeypots:

- Honeypots should not replace active patching of known vulnerabilities.
- Careful monitoring is crucial to avoid false positives from legitimate traffic.
- Skilled attackers may be able to identify honeypots, so they should be designed to appear as realistic as possible.

Some examples of honeypot solutions include Canary by Thinkst, **Modern Honey Network (MHN)**, and Cowrie.

Emerging technologies

Various new emerging technologies are impacting and will impact data security. Here is a list of some leading technologies impacting data security and protection. As organizations increasingly adopt cloud infrastructure, vulnerability management tools focus on securing multi-cloud and hybrid environments; features like automated asset discovery, vulnerability detection for cloud infrastructures, integration with CI/CD pipelines, and automated incident response are becoming standard.

Software bill of materials

SBOMs provide a detailed inventory of application components, helping organizations track vulnerabilities in third-party libraries or dependencies, making them a vital part of addressing supply chain attacks and risks. A growing emphasis is on enhancing software supply chain security and mitigating risks associated with vulnerabilities and license violations. The focus has shifted from merely generating SBOMs to robust SBOM management, which includes validation, data augmentation, normalization, and automated workflows for efficient handling. This evolution is driven by increasing regulatory mandates, such as Executive Order 14028 in the U.S.,

and industry-specific requirements in healthcare, automotive, and critical infrastructure sectors. Additionally, private industries are leading SBOM adoption by embedding requirements into supplier contracts, accelerating adoption timelines ahead of regulatory enforcement.

Automation and integration with DevOps pipelines are key trends enabling seamless SBOM generation and analysis throughout the software development lifecycle. Standardization efforts, particularly around formats like **Software Package Data Exchange (SPDX)** and CycloneDX, are improving interoperability and fostering collaboration across organizations. Furthermore, adopting cloud-native architectures and containerization technologies has amplified the need for SBOMs to manage complex software ecosystems effectively. Challenges such as high implementation costs, interoperability issues, and a shortage of skilled personnel persist but are being addressed through innovation in user-friendly tools and AI-powered vulnerability analysis. As awareness of supply chain risks like malicious code injection and license violations grows, SBOMs are increasingly recognized as critical components of proactive security models in modern software ecosystems.

Integration of incident response

Automation and orchestration are becoming indispensable in both vulnerability assessment and threat detection, and there is also an increased emphasis on integrating incident response with vulnerability management. Automated systems streamline processes such as auditing compliance, patching vulnerabilities, and isolating compromised endpoints during incidents. This reduces manual effort while improving efficiency and accuracy in mitigating risks. AWS and Azure provide automated patching solutions such as AWS Systems Manager and Azure Update Manager; their vulnerability detection services also enable automation of responses to vulnerabilities using serverless capabilities; for instance, AWS Inspector can trigger AWS Lambda functions to process vulnerabilities raised. Together, these advancements will drive a more proactive and adaptive approach as we advance. The following is a sample automation use case using AWS capabilities:

- Let us assume a critical CVE is detected in an EC2 instance.

- AWS Inspector sends a finding to Security Hub, which uses AWS Lambda to trigger the following operations:
 - Tag the instance as vulnerable.
 - Block outbound traffic with a security group update.
 - Creating a ticket in the incident management system.
 - Notify the security team and others who need to know.

AI-based prioritization

Artificial intelligence is in use in many aspects, but one key trend within vulnerability management is the adoption of AI-driven prioritization, where AI analyzes vast datasets to identify vulnerabilities based on real-world exploitability rather than severity scores alone. This approach enhances risk-based vulnerability management by integrating business context, environmental factors, and predictive models like **Exploit Prediction Scoring System (EPSS)**, allowing organizations to focus remediation efforts on vulnerabilities most likely to be exploited.

AI-assisted scanning and updates

Modern software leverages varied technologies and is composed of many different distributed components. The advent of automation in deployment has meant teams can now release software more often. Although there are clear benefits of this, as software can be constantly updated with the latest and more secure versions, vulnerability management, assessment, and updates are challenging in a continually changing environment. The recent advancements in AI, and specifically generative AI, are helping vulnerability scanning tools in scanning and adding context to the vulnerabilities, suggesting and explaining fixes. Tenable, Snyk, and others also offer AI-powered chat assistants to help security and development teams gain more structured insight and simplified explanations of complex vulnerabilities by sourcing data from across the technology estate as well as approved vulnerability databases. Lastly, these solutions can be integrated into an automated pipeline to prioritize, review, and auto-fix vulnerabilities.

Continuous assessment and threat intelligence

One of the key metrics related to vulnerabilities is the **time-to-exploit (TTE)**^[17], which is the average time to exploit a vulnerability before or after a patch is released. This metric has significantly shifted over the past five to six years, decreasing from 63 days to an average of five days. This necessitates faster detection and response capabilities, and continuous monitoring systems enable real-time detection of vulnerabilities across IT environments, reducing the exposure window and replacing traditional periodic scans. They help detect emerging vulnerabilities and instantly verify remediation efforts. Continuous assessment and monitoring also provide ongoing visibility into security posture, enabling early identification of misconfigurations and minimizing exposure.

In addition, the rise of cloud-native security approaches further highlights the need for continuous assessment. Dynamic cloud environments require tools that assess configurations, scan web applications, and test APIs in real time to address unique cloud-specific vulnerabilities. These tools ensure comprehensive protection across SaaS platforms, cloud workloads, and hybrid environments. We have already covered native tools from cloud providers such as AWS and Azure and third-party tools like Palo Alto, Aquasec, and Rapid7 InsightVM; all these offer automated workflows for remediation tracking, compliance reporting, and automation streamlining processes such as vulnerability scanning, patching, and reporting.

Beyond real-time detection and response, threat intelligence integration helps systems analyze user behavior and network traffic to detect deviations from standard patterns, flagging zero-day exploits, unauthorized access, or malware infiltration. Predictive analytics powered by machine learning also enables organizations to forecast attack trends based on historical data, proactively fortifying defenses against future threats. Incorporating threat intelligence into vulnerability management helps organizations identify vulnerabilities actively exploited by attackers, enabling faster response times and more precise prioritization of remediation efforts. AWS services integrate into Security Hub, Microsoft Defender, and Sentinel for a holistic view of threats and correlation across the entire estate, as well as threat intel and behavior analytics.

Cloud Native Application Protection Platform

Cloud Native Application Protection Platforms (CNAPPs) integrate **Cloud Workload Protection Platforms (CWPPs)** and **Cloud Security Posture Management (CSPMs)** to provide comprehensive security across the entire application lifecycle. These platforms ensure continuous code scanning during development, detecting vulnerabilities early, and reducing risk before deployment. CNAPPs also deliver automated runtime protection for cloud-native environments, securing containers, Kubernetes clusters, and serverless functions against evolving threats. They proactively identify and mitigate risks in real-time, leveraging AI-powered threat detection and response.

Real-world relevance

Some of the leading CNAPP solutions include Microsoft Defender for Cloud, Check Point CloudGuard, CrowdStrike Falcon Cloud Security, and Aqua Security. These solutions offer end-to-end security, compliance enforcement, and enhanced visibility across hybrid and multi-cloud environments. The two cloud-centric capabilities are covered in the following:

- **Microsoft Defender for Cloud** integrates CNAPP capabilities, including CSPM, CWPP, and DevOps security, providing unified security management across multi-cloud. Some example exploits are listed as follows:
 - A zero-day exploit targets containerized applications in **Azure Kubernetes Service (AKS)**.
 - Exploits a misconfigured identity permission to access sensitive data. Utilizes attack path analysis to visualize potential exploitation routes by integrating identity and access management insights.
- **AWS GuardDuty** is an intelligent threat detection service that helps protect AWS accounts and workloads from malicious activity and unauthorized behavior. It uses artificial intelligence and machine learning to monitor events generated by data services such as AWS S3 and Aurora, compute services such as **Elastic Kubernetes service (EKS)**, **Elastic Compute Cloud (EC2)**, and **Elastic Container Services (ECS)**. It generates a prioritized list of alerts based on findings; some sample use cases are listed in the following:

- Detect sensitive data being exfiltrated due to anomalous transfers based on S3 buckets and data access logs.
- Access to AWS resources using compromised credentials based on unusual API calls, access patterns, or unauthenticated user access.

Guidelines and checklists

This section covers specific guidelines and checklists for the three roles typically responsible for maintaining the security posture of products and applications. Use these with the checklists for a safer code, make, and deployment process. They may not be exhaustive, and you would still benefit from reading the rest of the topics and creating your own versions as an extension of these checklists.

For developers and engineers

This subsection summarizes guidance for anyone developing software or solutions for customers (internal or external). Product managers, designers, and architects can also use this to better understand the key principles their team needs to adopt or the features that need to be built:

- Understand the criticality of the system built and the impact of any vulnerabilities.
- Build systems using standard benchmarks for applications and databases such as CIS, cloud provider best practices, and OWASP.
- Regularly review and update code to address security weaknesses and vulnerabilities.
- Conduct code reviews focused on security, not just functionality.
- Integrate automated security scanning (SCA/OSS/SAST/DAST) into build pipelines.
- Respond promptly to vulnerability findings and apply remediation or mitigation as required.
- Document all vulnerability fixes and ensure traceability in the codebase.
- Test code changes for security regressions after remediation.
- Stay current with libraries, dependencies, and frameworks to minimize

exposure to known vulnerabilities.

For infrastructure engineers

This subsection summarizes guidance for anyone who deals with infrastructure (either on-premises or cloud):

- Maintain a comprehensive, up-to-date inventory of all infrastructure assets, including cloud and on-premises resources.
- Schedule and perform regular vulnerability scans across infrastructure, applications, and containers.
- Adopt secure configuration management based on benchmarks such as CIS and monitor for misconfigurations in all environments.
- Integrate vulnerability scanning and assessment into CI/CD workflows for rapid detection and response.
- Prioritize vulnerabilities using risk-based approaches (e.g., CVSS), focusing on business-critical assets.
- Develop and automate patch management and release processes for the timely remediation of vulnerabilities.
- Validate remediation by rescanning and conducting penetration tests or security validation.
- Monitor for new vulnerabilities and emerging threats, integrating CSPM, CWPP solutions, and capabilities.
- Document exceptions and accepted risks, ensuring leadership sign-off for deferred remediation.
- Coordinate with incident response teams to ensure vulnerabilities triggering incidents are addressed swiftly.

For leaders

This subsection summarizes guidance for anyone who leads or manages a team, including product managers, designers, and architects. It also benefits people supporting organizational functions like risk and compliance:

- Define the scope, objectives, and KPIs for the vulnerability management program, which are aligned with organizational goals, risk appetite and processes.

- Ensure compliance with regulatory, contractual, and industry requirements related to vulnerability management.
- Allocate resources and support cross-functional collaboration for vulnerability remediation.
- Set up regular reporting on vulnerability status, trends, and remediation progress for stakeholders.
- Support the integration of vulnerability management into product development and release cycles.
- Approve and review exception handling processes for vulnerabilities that cannot be immediately remediated.
- Evaluate the effectiveness of the vulnerability management program through periodic security and compliance audits and reviews.
- Stay informed about emerging technologies and evolving threats to guide strategic security decisions.

Conclusion

Organizations today face a complex landscape of vulnerability management challenges that threaten their digital assets and business continuity. Managing vulnerabilities in environments based on modern cloud-native technologies is increasingly difficult due to rapid technological changes, persistent misconfigurations, and the sheer complexity of multi-cloud setups, expanding the attack surface and making real-time visibility essential. At the same time, many large organizations still rely on legacy systems (such as mainframes and old monolithic systems) that present significant risks because they often lack modern security features, are incompatible with current security tools, and rarely receive critical updates, yet they remain integral to many businesses due to cost and operational dependencies. There is also a rise in zero-day vulnerabilities and the sophistication of attacks, which further complicates matters, demanding proactive threat intelligence, continuous monitoring, and agile incident response to detect and mitigate exploits before they cause harm. These challenges require a coordinated, adaptive approach to vulnerability management, which we covered in this chapter.

The next chapter will focus on *Continuous Monitoring*, which is the last pillar but brings and connects the previous pillars to provide an end-to-end approach to Zero Trust security.

Points to remember

- A vulnerability is a weakness or flaw in a system that cybercriminals can exploit to gain unauthorized access or cause harm. There are various types of vulnerabilities, from software and hardware to misconfigurations and access or encryption issues.
- Vulnerability management lifecycle is a continuous process involving discovery, prioritization, assessment, remediation, validation, and reporting of vulnerabilities to reduce risk.
- Asset discovery and inventory involve identifying and cataloging all IT assets, including hardware, software, and cloud and digital assets, to establish a comprehensive baseline for security management.
- Configuration management involves maintaining secure configurations based on information systems benchmarks, monitoring for deviations, and remediating misconfigurations to ensure they are configured securely and to minimize risk.
- Vulnerability scanning encompasses using various tools, including SCA, OSS, SAST and DAST, and network discovery.
- Risk assessment evaluates and prioritizes vulnerabilities based on their likelihood of exploitation and potential impact to focus resources on the most critical risks. Business impact analysis typically examines the enterprise context to evaluate the impact of vulnerabilities and help prioritize resources.
- Remediation and patch management, addressing and fixing vulnerabilities through patching, configuration changes, or other mitigation actions, starting with the highest-risk issues.
- Verification and testing; validating that remediation efforts have been successful through rescanning, audits, and continuous monitoring to ensure vulnerabilities are resolved.
- Ongoing monitoring and improvement can be simplified using solutions

such as CSPM, which monitors the secure baseline of cloud security configuration; CWPP, which continuously monitors workloads, enforces least privilege access, and applies runtime security controls testing; and honeypots.

- Reporting involves documenting identified vulnerabilities, remediation actions, and risk levels to inform stakeholders, support compliance, and drive ongoing improvement.
- Some of the emerging trends and technologies in this space include standardization and adoption of SBOM, integration of incident response with vulnerability management, AI-based prioritization, continuous assessment, threat intelligence, and CNAPP.

References

1. <https://spectreattack.com/spectre.pdf>
2. <https://meltdownattack.com/meltdown.pdf>
3. <https://cams.mit.edu/wp-content/uploads/capitalonedatapaper.pdf>
4. <https://www.reuters.com/article/business/capital-one-to-pay-80-million-fine-after-data-breach-idUSKCN2522D8/>
5. <https://oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf>
6. <https://www.fca.org.uk/publication/final-notice/equifax-limited-2023.pdf>
7. <https://github.com/kubernetes/kubernetes/issues/81112>
8. <https://www.aquasec.com/blog/kubernetes-exposed-one-yaml-away-from-disaster/>
9. <https://www.rapid7.com/blog/post/2020/04/02/dispelling-zoom-bugbears-what-you-need-to-know-about-the-latest-zoom-vulnerabilities/>
10. <https://www.cloudquery.io/>
11. <https://aws.amazon.com/blogs/security/consolidating-controls-in-security-hub-the-new-controls-view-and-consolidated-findings/>
12. <https://community.blackduck.com/s/article/How-to-tell-what->

changed-from-one-Black-Duck-scan-to-the-next

13. [**https://docs.snyk.io/scan-with-snyk/snyk-code/manage-code-vulnerabilities**](https://docs.snyk.io/scan-with-snyk/snyk-code/manage-code-vulnerabilities)
14. [**https://nvd.nist.gov/vuln-metrics/cvss**](https://nvd.nist.gov/vuln-metrics/cvss)
15. [**https://learn.microsoft.com/en-us/azure/governance/policy/how-to/get-compliance-data**](https://learn.microsoft.com/en-us/azure/governance/policy/how-to/get-compliance-data)
16. [**https://learn.microsoft.com/en-us/azure/defender-for-cloud/workload-protections-dashboard**](https://learn.microsoft.com/en-us/azure/defender-for-cloud/workload-protections-dashboard)
17. [**https://cloud.google.com/blog/topics/threat-intelligence/time-to-exploit-trends-2023**](https://cloud.google.com/blog/topics/threat-intelligence/time-to-exploit-trends-2023)

CHAPTER 8

Continuous Monitoring

Introduction

In this ever-changing world, implementing security across all the pillars without considering ongoing monitoring and management is insufficient. This means organizations must constantly monitor network traffic, user behavior, and device status in real-time to detect threats as they emerge and support the ongoing improvement of security policies and controls. A continuous improvement mindset ensures that security measures evolve in tandem with new threats, vulnerabilities, and business requirements within the context of your organization. This makes Zero Trust a dynamic, adaptive, and resilient security model, with continuous and protective monitoring serving as a cornerstone of Zero Trust.

Structure

The following topics are covered in the chapter:

- Case study: Colonial Pipelines
- Core concepts
- Monitoring strategy
- Emerging technologies
- Guidelines and checklists

Objectives

This chapter will cover various aspects of continuous protective monitoring and security information and event management, starting with a monitoring strategy and infrastructure, and followed by incident response, team management, reporting, and metrics. We will also cover the importance of backup and recovery procedures and processes.

Case study: Colonial Pipelines

Let us look at a case study for a better understanding:

- **Date:** May 2021
- **Summary:**
- **Root cause:** A dormant account password for a VPN account that did not require MFA^[1].
- **Impact:**
 - The six-day operational shutdown starting from May 7th 2021, disrupted 45% of the East Coast's fuel supply chain, affecting 50 million customers across 13 states. 44% average price increases of \$0.88 per gallon were seen across affected areas, with a peak of \$1.97. ^[1]
 - Colonial Pipelines lost \$8.46 million daily, while the cascading costs of supply chain disruptions were \$27.2 million per day. The initial ransomware payment was \$4.4 million, though up to 77% was retrieved by law enforcement. ^[1]
 - The total economic impact, accounting for direct and indirect costs, exceeded \$2.1 billion. ^[1]

Detailed timeline^[1]

Colonial Pipeline was founded in 1962. It is the most extensive infrastructure, spanning 5,500 miles through 13 states, making it critical to North America's energy distribution system. The network relies on thousands of distributed sensors and a couple of hundred automated pumps

coordinated by 18 regional centers. This system exchanges over 4 million data points hourly for optimal operational efficiency, achieving 98.7% operational efficiency under standard operations with an annual delivery of 921.5 million barrels.^[1]

In May 2021, Colonial Pipeline suffered a ransomware attack, which was considered the most significant cyberattack on an oil infrastructure target in the United States' history. The company relied on a legacy VPN infrastructure lacking fundamental security controls such as MFA. Further investigations revealed that the initial breach occurred on April 29, 2021, and exploited a vulnerable VPN account that retained permissions to the AD, even though it had been dormant since January 2021. These credentials were traded on underground forums since March 2021.

Once in, the threat actors deployed PowerShell-based persistence mechanisms and established seventeen backdoor accounts across the network. They also leveraged compromised domain controller credentials, compromising 142 endpoints spanning 24 network segments as part of lateral movement. The threat actors then modified group policy objects on domain controllers, methodically scanned thousands of IP addresses, and mapped the network architecture. The attackers then went on to exfiltrate 101.325 GB of compressed corporate data through encrypted channels, followed by deployment of ransomware payloads across 29 critical billing systems and IT infrastructure on days four and five of the attack. The last phase on day six was ransomware payload deployment across 29 critical systems controlling billing and IT infrastructure.

The attack was noticed at 05:30 a.m. on May 7th, 2021, and the company immediately initiated an emergency response, which was followed by a complete operational shutdown of its pipeline infrastructure.

The initial response involved 85 internal cybersecurity personnel supplemented by 37 external incident response specialists from four major cybersecurity firms. A centralized command center was established, and twelve federal agencies were also involved, and secure communication channels were established. The team reconfigured network devices and enhanced segmentation protocols to create 23 isolated security zones. An enhanced monitoring infrastructure incorporating next-generation security sensors and advanced intrusion systems was deployed. Within 48 hours,

recovery teams had restored core business applications and implemented robust security controls, including multi-factor authentication and advanced endpoint protection. On day four, recovery operations started with 42% of normal capacity to allow for sufficient monitoring and integrity checks, and were scaled up to 75% on day five, and by day six, back to full operational capacity.

The entire outage led to a six-day operational shutdown of the pipeline network; fuel availability dropped to critical levels across many states, leading to a regional emergency declaration for 17 states and Washington, D.C., to keep fuel supply lines open. The company had to pay the amount that was demanded by the hacker group (75 bitcoin or \$4.4 million). A direct impact was fuel shortages at Douglas Airport, requiring American Airlines to change flight schedules temporarily, and at least two flights (to Honolulu and London) had fuel stops or plane changes added to their schedules for four days. Over the days, this led to fuel shortages across multiple states.

Learnings

Let us look at the learnings of this case study:

- **Network segmentation:** Attackers deployed encryption payloads across multiple network segments, forcing Colonial Pipeline to cease operations across its entire pipeline network. Post-incident analysis also identified that 47% of their active VPN sessions originated from unauthorized IP ranges. 84% of remote access traffic bypassed modern intrusion detection systems. These issues highlight significant gaps in network segmentation, monitoring, and access control mechanisms.
- **Privileged access:** There were also lapses in managing privileged access, with 189 user accounts maintaining elevated privileges without documented business requirements. Sound privileged access management is essential to limit their usage and reduce the risk of breaches.
- **Multi-factor authentication:** The lack of MFA on virtual private network connectivity allowed attackers to gain access to a dormant account.
- **Monitoring:** Attackers successfully exfiltrated 100GB of sensitive data within 12 hours of initial access. Ideally, such large data movements

should have triggered a few alerts and a more proactive response from teams.

- **Response:** Incident response drills and continuity tests are critical to faster recovery, and organizations with regular drills, updated recovery plans and procedures show a lower mean time to respond and recover.
- **Security awareness:** Research across organizations after the incident also indicated that only 31% of critical infrastructure personnel had received comprehensive security training within the previous year, and 43% operated without formalized incident response procedures.

Other notable examples

In the last eight chapters, we have covered various case studies. Let us combine them to understand how continuous monitoring could have been essential to ensuring security. [Table 8.1](#) summarizes all the case studies, the root causes behind each attack or breach, and how continuous monitoring could have helped in each situation:

Case study	Root cause	Learnings related to continuous monitoring
Target (Chapter 2 , <i>Architecture and Pillars of Zero Trust</i>)	Weak device used by a third-party contractor firm	Enhanced security monitoring and intrusion detection systems could have helped in addition to security awareness
SolarWinds (Chapter 3 , <i>Secure Your Code, Build and Deploy</i>)	Supply chain attack on the build system	Centralized logging and better monitoring to assist with investigation and forensics
Uber (Chapter 4 : <i>Secure Your Identities</i>)	Publicly posted access keys to AWS S3 storage	Inadequate security incident management and reporting
NotPetya (Chapter 5 , <i>Secure Your Network</i>)	Malware entered through third-party compromised systems spread across Maersk's global infrastructure	Tried and tested incident response plans help respond rapidly and limit impact Regularly updated offline backups, Maersk found a single offline backup in a remote location that was crucial to recovery Redundancy in IT systems and multi-region resilience, due to the global nature of the Maersk estate, resources from unaffected regions could be involved in recovery

Case study	Root cause	Learnings related to continuous monitoring
Marriott (Chapter 6, Secure Your Data)	Compromised server allowing attackers to access customer data	Customer data retention should be limited to the required duration
Capital One (Chapter 7, Manage Vulnerabilities)	Firewall misconfiguration allowed the attacker to execute commands, enabling unauthorized access	Continuous monitoring, robust intrusion detection, and monitoring systems could help detect anomalous access Automatic remediation workflows are integrated with a monitoring system to remediate misconfigurations
Equifax (Chapter 7, Manage Vulnerabilities)	Outdated software component that was not patched in a timely manner	The solution used to monitor network traffic had been inactive due to an expired security certificate and could potentially have alerted teams had it been active

Table 8.1: Case study summary across chapters

Core concepts

All the pillars we have covered help keep your applications and infrastructure secure, regardless of where they are deployed. Beyond implementing these pillars, one also needs to ensure that the controls implemented as part of these pillars are continuously monitored and improved—the easiest way to understand this is on your vehicle’s dashboard. A vehicle dashboard has various key indicators—speedometer, fuel gauge, warning lights for oil, tire pressure, etc. Now imagine driving a car without any of these indicators or warning lights. Your car will run and function until it runs out of fuel or oil, but without the real-time feedback of the current state of the key indicators, you would have to monitor all of these controls regularly before you use the vehicle, or your vehicle will either stop abruptly or break down unexpectedly. Every vehicle’s dashboard provides real-time information about its health and performance. It offers continuous visibility to monitor vital systems and alert you if something needs action. This helps maintain the vehicle's health and prevents minor issues from causing a catastrophic failure.

We need a similar monitoring solution for the various pillars of security we have covered so far, for instance, to review the state of our code and

vulnerabilities in code, the health of devices connected to our network, the safety of credentials and privilege access, and anomalies in network activity. In addition to reviewing the current state, it should also help us improve the implemented solutions and raise awareness among our teams, so they are better equipped. The continuous monitoring pillar covers all of these aspects.

The National Institute of Information Security defines **information security continuous monitoring (ISCM)** as *maintaining ongoing awareness of information security, vulnerabilities, and threats to support organizational risk management decisions*^[2]. Continuous monitoring is a foundational component of Zero Trust security, enabling real-time visibility into user activity, device health, and network behavior to enforce dynamic access controls and detect anomalies. It will collect and analyze data across systems, establish baselines for regular activity, and identify deviations that may indicate threats. However, collecting data and generating alerts alone will not help; we would also need the ability to respond to mitigate risks, ensure compliance, and continuously improve security and safety overall.

Let us first examine what we need to monitor, what types of data we collect, and from what sources:

- **User and Entity Behavior Analytics (UEBA)** involves monitoring user actions and behaviors to establish baselines and detect anomalies. This applies to two sets of users: enterprise users or employees and end users or customers who use your organization's products and solutions. A sound monitoring system will need to monitor both types of users and their activities.
- **Device and endpoint** access the health of devices and endpoints, firmware and software versions, patch status, and overall security posture. With the growth in the Internet of Things and connected devices, this area is continuously expanding and critical to overall security posture.
- **Network activity monitoring** involves reviewing and tracking traffic and data flows, protocol use, and unusual patterns. Many attacks include data exfiltration, and a good network activity monitoring solution can help identify these anomalies more proactively.
- **Geolocation and contextual** signals use the device's or user's location and contextual data to inform access and detect suspicious activity.

- **Privilege and credential monitoring** examine access rights, escalation, and usage. Previous case studies show that attackers typically use unused, dormant, and compromised credentials to breach networks, making credential monitoring an essential part of security.

The monitoring data we capture can be classified into the types shown in *Table 8.2*:

Type	Examples
Application	• WAF and API monitoring • App server logs from servers such as Tomcat, Node.js, Django • Error logs including exceptions and stack traces • Transaction logs from APIs, payment flows, and critical workflows • Business events such as user sign up, order placed, booking confirmed, payment made
Data	• Query performance metrics (slow queries, locks, deadlocks) • Audit and activity logs (who accessed what and when) • Replication and backup status • File integrity and data loss prevention (DLP)
Cloud specific	• Cloud service metrics such as AWS CloudWatch, Azure Monitor • Cost and usage reports (to watch for sudden spikes or possible misuse)
Infrastructure	• System logs: syslog, journalctl outputs from systems • Server metrics such as CPU, memory, disk I/O, and network usage • Cloud resource events such as service spin-up and down, autoscaling events • Container logs: Docker logs and Kubernetes events
Network	• Traffic flows such as NetFlow, VPC flow logs • DNS queries and responses • Load balancer logs from the likes of AWS Elastic Load Balancer, Azure Application Gateway, etc. • Firewall and traffic inspection logs • Service mesh and micro-segmentation
User activity	• Synthetic transactions such as fake users doing logins, checkouts, etc. • Real user monitoring (RUM), such as actual user load times, application errors • Mobile app monitoring related to crashes and performance issues
Endpoint	• Device trust evaluation—status of compliance, patching, encryption • Endpoint telemetry (login, logout, access, changes, etc.) • Secure endpoint posture checks
Change	• Logs and events from change systems, such as continuous integration and deployment tools • Build and deployment logs from build and deployment tools such as Jenkins,

	GitHub • IaC changes (Terraform plans/applies) • Configuration drift detection
Security logs	• Authentication and authorization logs related to login success/failure, multi-factor authentication events • Firewall logs, such as traffic allowed/blocked • Intrusion detection/prevention logs and alerts • Vulnerability scan outputs
Operational	• Events based on golden signals in operational monitoring • Uptime of availability of a service and downtimes • Traffic is the amount of demand on the system or service • Errors and failures in processing requests • Saturation is information in resource usage and exhaustion
Compliance	• Policy changes and logs such as IAM changes, firewall rule updates • Access control logs, encryption, and key management logs

Table 8.2: *Principles of continuous monitoring*

A continuous monitoring solution needs to be based on some key principles that are important in today's world; these are summarized in [Table 8.3](#):

Principle	Description
Always-on	The uptime of a continuous monitoring system is as crucial as the system it monitors, and this includes not just the tools but also the people supporting it.
Real-time	Due to the nature and volume of data, periodic checks are no longer sufficient. It is crucial to have continuous monitoring to have access to live information.
Continuous	Monitoring is not a point-in-time response anymore, as threats are everywhere and can occur at any time of the day; it requires constant vigilance and continuous review, response, and compliance.
Holistic	A good system needs to integrate monitoring across infrastructure, network, user experience and security and provide a single view of all threats and alerts.
Rapid response	Detecting early signs and reacting before they become outages or breaches is critical and requires a rapid response process in place.
Collaboration	For the system and process to be effective, monitoring and response procedures should be a shared responsibility across developers, security, product, business and operations teams.
Accessible	Collaboration is only possible if the system is accessible to everyone involved; if the system is only available to a select few, it will restrict people from responding to failures and threats.
Awareness	Making sure teams are aware of the systems and processes will ensure they can respond in time

Table 8.3: Principles of continuous monitoring

Figure 8.1 shows all the core activities, capabilities, and processes that comprise a continuous monitoring system within an organization:

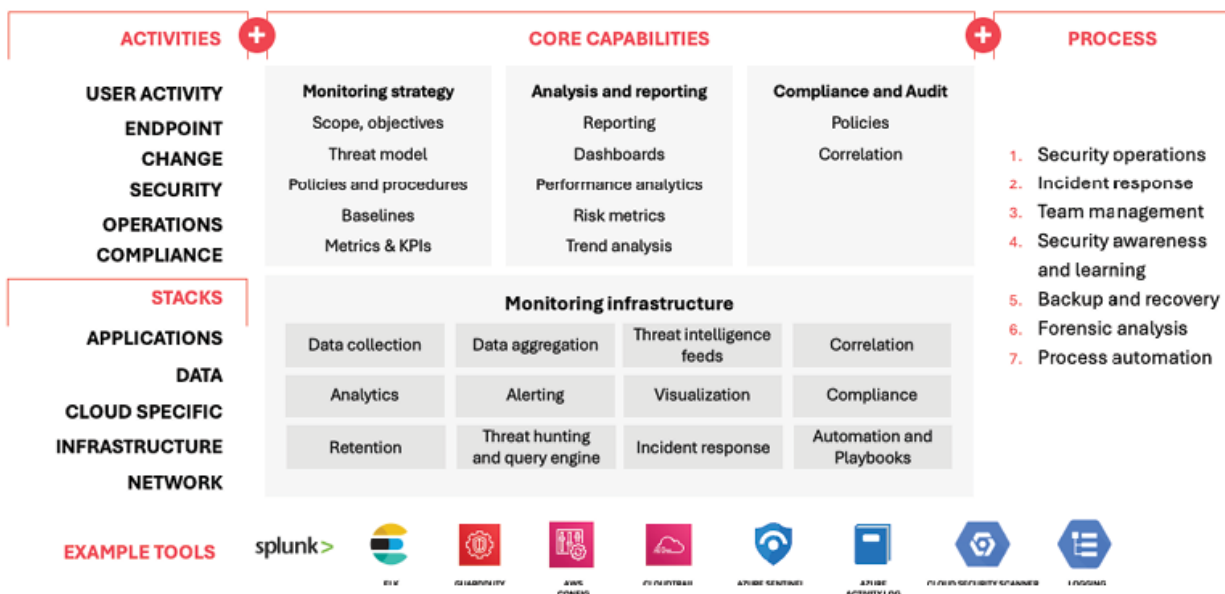


Figure 8.1: Continuous monitoring and compliance overview

The following are the core areas of continuous monitoring as depicted in the figure:

- **Activities** refer to the various types of data, events, and logs we capture to build a comprehensive view of security across the organization. **Stacks** represent the other layers, tiers of information from which we capture these activities. Both of these are covered in [Table 8.2](#).
- **Core capabilities** include the strategy and objectives, technical infrastructure, and features supported by the monitoring solution, and comprise of:
 - **Monitoring strategy** sets up the overall scope, objectives, and establishes baselines and KPIs for monitoring.
 - **Monitoring infrastructure** encompasses key tools and integration with security tools, including connections to SIEM, SOAR, and other platforms, for comprehensive visibility and response.
 - **Analysis and reporting** leverage tools for pattern recognition, anomaly detection, and automated reporting.

- **Compliance and audit** support ensure continuous monitoring, support compliance reporting, and audit readiness.
- These are then combined with organizational **processes** listed as follows, for end-to-end operations:
 - **Security operations** involve utilizing a centralized facility or a third-party service to monitor, detect, analyze, and respond to cybersecurity threats.
 - **Incident response** helps detect, investigate, and mitigate security incidents to minimize impact.
 - **Team management** involves coordinating and collaborating on incident response across teams to ensure sufficient coverage around the clock.
 - **Security awareness and learning** to improve security awareness throughout the organization, including with third parties, and building a shared understanding of the organization's risks and threats
 - **Backup and recovery** involve setting up offline backups and a recovery process to restore operations after breaches and incidents while maintaining data security.
 - **Forensics** analyzes and reconstructs what happened during an incident to understand its root cause and improve future responses.
 - **Process automation** of repetitive tasks related to detection and response to improve efficiency and reduce risk.
 - **Continuous improvement** cycle utilizes monitoring insights to adapt and strengthen security posture over time.

We will now cover the core capabilities and processes. A detailed review of organizations' monitoring strategies and their execution is beyond the scope of this book; we will only cover concepts relevant to understanding what needs to be in place and provide practical advice on how to apply them in the day-to-day implementation of your applications and infrastructure.

Monitoring strategy

A well-defined strategy includes what, how, and why to monitor; it states the

objectives, scope, and processes for collecting, analyzing, and responding to security-related data across the organization. It encompasses the selection of monitoring targets, the establishment of baselines, the development of use cases, alert configuration, and the integration of automated tools for data collection and analysis. The strategy also sets metrics and **key performance indicators (KPIs)** to measure effectiveness and guide continuous improvement. Here are some steps to follow to create a monitoring strategy at the organization or team level:

1. Define the **scope and objectives** of what needs to be monitored and why, aligning monitoring activities with organizational risk tolerance and business goals, and involving stakeholders to ensure objectives are practical and relevant.
2. Establish a **threat model** that covers internal and external threat actors, the attack surface, and critical assets. Map out the flow of data and the trust boundaries.
3. Document **policies and procedures** such as data collection standards, roles and responsibilities, alerting and response procedures, escalation procedures, and reporting requirements.
4. **Prioritize** the monitoring scope by focusing on high-risk areas first.
5. Assess **data collection and integration** needs by selecting the sources of various signals, logs, and events to be captured, and identify existing systems to integrate.
6. Establish **baselines** for normal operations and behaviors so anomalies can be identified, using analytics from systems to determine normal.
7. Establish **metrics and KPIs** to measure detection speed, accuracy, and resolution.
8. **Iterate and adapt** to improve the strategy with evolving threats, changes to business, and lessons learned.

Monitoring infrastructure

The foundational technology for every organization's security monitoring infrastructure is a SIEM system. SIEM is a technology solution that aggregates, analyzes, and correlates security data and logs across multiple sources (including endpoints, servers, applications, networks, and other

appliances) to provide a centralized view of an organization's security landscape. Splunk Enterprise Security, IBM QRadar, Microsoft Sentinel are some of the widely used SIEM products.

The term was coined in 2005 in a *Gartner*^[3] report that proposed a new security information system based on two previous generations: **security information management (SIM)** for long term storage, analysis and reporting on log data and combined with threat intelligence; and **security event management (SEM)** for events from security systems such as antivirus, firewalls and intrusion detection systems. Many vendors started offering SIEM platforms that aggregate log data and real-time events and help identify anomalies, vulnerabilities, and incidents. *Figure 8.2* shows the core capabilities of a SIEM infrastructure:^[3]

Monitoring infrastructure			
Data collection	Data aggregation	Threat intelligence feeds	Correlation
Analytics	Alerting	Visualization	Compliance
Retention	Threat hunting and query engine	Incident response	Automation and Playbooks

Figure 8.2: Monitoring infrastructure capabilities ^[3]

Table 8.4 covers the details of each of these capabilities:

Component	Details
Data collection	Collect logs, telemetry, and events from endpoints, servers, and network devices. E.g. FluentD, Filebeat, CloudWatch agents
Data aggregation	Centralized collection and normalization of logs from various sources. E.g., Logstash, Fluent Bit, GrayLog
Threat intelligence feeds	Enriches security data with external threat indicators and intelligence feeds. MISP, ThreatConnect, Anomali ThreatStream
Correlation	Based on the data and events sourced from various sources, link events, logs and data to an incident, threat, or vulnerability
Real-time analytics	Use machine learning to identify relationships in data and detect suspicious deviations from normal user or system behavior
Alerting and notification system	Sends alerts through multiple channels based on detection rules or anomalies. PagerDuty, Microsoft Sentinel Alerts

Component	Details
Dashboard and visualization tools	Real-time visual dashboards to monitor security posture and incident trends. Kibana, Splunk Dashboards
Compliance	Gathering of compliance data and producing reports to adapt security and governance for compliance standards.
Retention	Long-term storage of historical data for analytics, tracking, compliance and forensic purposes.
Threat hunting and query engine	Enables searching using structured query languages, filters, and pivoting data to uncover threats and vulnerabilities. Splunk SPL, Azure KQL, Elastic Query DSL
Incident response	Supports teams to respond to incidents quickly through case management, collaboration, and a knowledge base
Automation and playbooks	Define workflows and integrate with other systems to automatically handle common incidents (e.g., isolate host, block IP)

Table 8.4: Monitoring infrastructure capabilities

Technology integration

Continuous monitoring integrations enable seamless data flow and coordinate defense across diverse systems. Integrating endpoint security, cloud services, IAM systems, and network devices with SIEM and SOAR platforms allows organizations to centralize visibility and streamline incident response. APIs, event forwarding, log collectors, and native connectors are commonly used to bridge tools, ensuring that telemetry from SaaS apps, on-prem infrastructure, and cloud environments feed into a unified threat detection and response framework. This interoperability is essential for enabling automation, reducing alert fatigue, and improving **mean time to detect (MTTD)** and **mean time to recover (MTTR)**. [Table 8.5](#) details some of the standard systems and capabilities and their scope; typically, they are all integrated into the SIEM. Many of these capabilities are available out of the box in modern SIEM solutions.

System or capability	Details
Security orchestration, automation, and response (SOAR)	Automates incident response processes using predefined playbooks. E.g., IBM QRadar, Splunk SOAR, Microsoft Sentinel. This capability is often part of the SIEM platform (for instance, SIEM, Sentinel).
Threat intelligence platform (TIP)	Enriches security data with external threat indicators and intelligence feeds. E.g., malware information sharing platform (MISP) , ThreatConnect, Anomali ThreatStream.

System or capability	Details
Alerting and notification system	Sends alerts through multiple channels based on detection rules or anomalies. E.g., PagerDuty, Microsoft Sentinel, and Azure Monitor Alerts.
Dashboard and visualization tools	Real-time visual dashboards to monitor security posture and incident trends. E.g., Kibana, Grafana, Splunk Dashboards, Microsoft Sentinel.
UEBA	Baseline normal behavior for users and devices, then detecting anomalies that deviate from those baselines, assigning risk scores, and detecting threats and compromised accounts. E.g., Splunk User Behavior Analytics (UBA) , Microsoft Defender.
Endpoint detection and response (EDR)	Provides visibility, detection, and remediation capabilities at the endpoint level (i.e., physical devices connected to a network, such as mobile phones, desktops, laptops, virtual machines, and Internet of Things (IoT) technology). E.g., CrowdStrike Falcon, SentinelOne, Microsoft Defender for Endpoint.
Extended detection and response (XDR)	Detection and responding to threats across wider security domains, including network, cloud, and email security. E.g., Microsoft Defender XDR, CrowdStrike Falcon.
Network monitoring	Monitoring network infrastructure such as firewalls, routers, and network flows.
Network intrusion detection/prevention (NIDS/NIPS)	Detects or prevents suspicious network activity based on traffic analysis. E.g., Palo Alto Networks IDS/IPS, Check Point Quantum IPS, Cisco Secure IPS.
Vulnerability management system	Identifies and reports vulnerabilities on assets through scanning. E.g., Tenable Nessus, Qualys, Rapid7 InsightVM, Microsoft Defender.
IAM and identity telemetry	Monitors authentication logs, single sign-on, role changes, MFA activity, and access anomalies. E.g., Okta, Azure Entra, Duo Security, AWS IAM logs.
Application telemetry and secure code insights	Collects app logs and performs code analysis for security issues during development. E.g., Sonar, Snyk, Elastic APM, GitHub Advanced Security, Checkmarx.
Cloud security posture management (CSPM)	Scans cloud environments for misconfigurations and compliance gaps. E.g., Prisma Cloud, AWS Security Hub, Microsoft Defender for Cloud.
Device posture and asset inventory	Monitors device configurations, patch levels, and security posture. E.g., CrowdStrike Discover, Tanium, Microsoft Intune.

System or capability	Details
Data loss prevention (DLP)	Prevents unauthorized data transfers and monitors for sensitive data exposure. For instance, to ensure that a customer service with access to personally identifiable information (PII) cannot copy or extract data outside approved tools. Three main types of DLP exist based on their area of focus: Network, Endpoint, and Cloud. E.g., Symantec DLP, Microsoft Purview, Forcepoint DLP.
Data activity monitoring	Track and analyze all interactions with sensitive data across various data stores that store sensitive data through audit trails in case of a breach or attack, and allow organizations to implement proactive monitoring to identify unauthorized access to sensitive information.

Table 8.5: Monitoring infrastructure integrations

Analysis and reporting

With all the integrations, the monitoring platform will capture tremendous amounts of data from various places; however, it still needs to be turned into actionable insights, and the following are some of the outputs it needs to provide:

- **Executive dashboards** that provide real-time visibility into threat landscapes and operational health, enabling leadership to understand the current issues and prioritize resources.
- **Compliance reporting** that tracks adherence to regulatory frameworks (e.g., GDPR, HIPAA) by systematically documenting controls, audits, and remediation efforts.
- **Performance analytics** that track key performance indicators of the security operations, such as mean time to detect and recover and automate workflows to optimize SecOps efficiency.
- **Risk metrics** that quantify vulnerabilities and threat exposure using severity scores, while trend analysis identifies patterns in attacks or policy violations over time.
- **Return on investment** metrics that show the cost-effectiveness of security investments by comparing operational savings (e.g., reduced downtime) against tooling and labor costs.

Figure 8.3 shows an overview dashboard provided by Sentinel with some of the standard metrics, such as Incidents reported by severity and data flows. Many more standard metrics and dashboards are available, and the ability to create custom dashboards and reports.

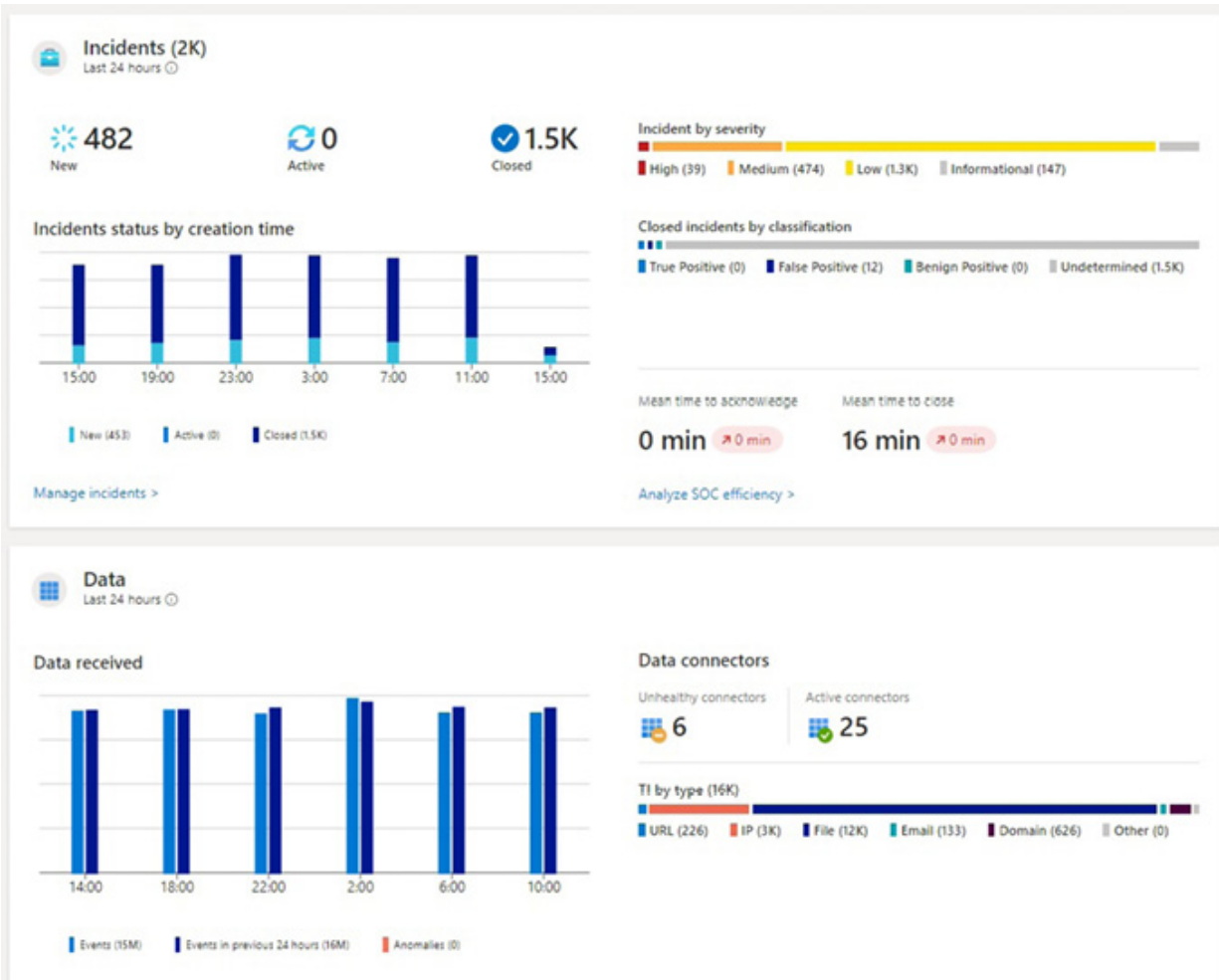


Figure 8.3: Microsoft Sentinel dashboard ^[4]

Compliance and audit

Organizations in regulatory industries such as financial services and healthcare must meet regulatory requirements and maintain trust and operational resilience, making compliance and audit an integral part of continuous monitoring. Specialized modules or platforms (e.g., AWS Config, Azure Policy, or third-party tools like Qualys) continuously assess the environment against regulatory standards (such as NIST, CIS, and FIPS), automating compliance checks for configurations, data handling, and access controls. These tools perform continuous control assessments, validating that security controls, such as least-privilege access, encryption, and patch management, are consistently enforced and effective. Real-time monitoring of user activity, system configurations, and network traffic generates audit trails, enabling organizations to identify swiftly and remediate deviations

(e.g., unauthorized changes or policy violations). For audits, these platforms provide pre-built reports and dashboards that map controls to specific regulatory frameworks, drastically reducing manual effort during compliance reviews. All of this helps ensure ongoing compliance and supports audit processes.

Security operations center

Security operations center (SOC) is a centralized facility staffed by IT security professionals dedicated to continuously monitoring, detecting, analyzing, and responding to cybersecurity threats across an organization's entire IT environment. The SOC team operates around the clock, leveraging a suite of security tools, including SIEM, IDS/IPS, endpoint detection, and network monitoring, to maintain real-time visibility and orchestrate rapid responses to incidents. Their responsibilities extend beyond threat detection and response to include vulnerability management, compliance monitoring, and the maintenance of security policies and technologies. SOC can be set up in-house, fully outsourced, or hybrid with a managed security service provider.

Managed detection and response (MDR) is an outsourced security service that combines advanced technologies, such as EDR, SIEM, and SOAR, with human expertise to provide proactive, 24/7 threat monitoring, threat hunting, and incident response. MDR providers analyze endpoint, network, and cloud data, using **artificial intelligence (AI)** and machine learning to identify sophisticated threats and automate remediation actions, often filling gaps for organizations lacking the resources or expertise to operate a complete in-house SOC.

Incident response

Despite applying security controls, incidents do happen, and establishing an incident response process with tailored procedures ensures the appropriate handling of security incidents promptly. An incident is defined by NIST publication^[5] as *an occurrence that actually or imminently jeopardizes without lawful authority, the integrity, confidentiality, or availability of information or an information system; or constitutes a violation or imminent threat of violation of law, security policies, security procedures, or*

acceptable use policies. Some examples cited in the publication include high volumes of connection requests (such as DDoS), compromising sensitive data by compromising administrative credentials, deploying ransomware, and using phishing to compromise emails. All of these have the potential to compromise data and cause damage to an organization, its employees, or consumers.

Incident response is a systematic approach to detecting, analyzing, and responding to security breaches or attacks to minimize damage, reduce recovery time and costs, and protect sensitive data. The NIST publication shows a high-level response lifecycle with key functions outlined in [Figure 8.4](#), to prevent and prepare to handle incidents that occur, reduce impact, improve response, and manage risk. The key functions are listed as follows:

- **Govern:** The risk management strategy, expectations, and policy are established, communicated, and monitored.
- **Identify:** The current risks are understood.
- **Protect:** Safeguards and protections are used to manage risks.
- **Detect:** Find and analyze attacks and compromises.
- **Respond:** Actions taken regarding the detected incident are taken.
- **Recover:** Restore affected assets and operations.
- **Improvement:** It continuously reviews lessons learned, root cause analyses, and other functions.

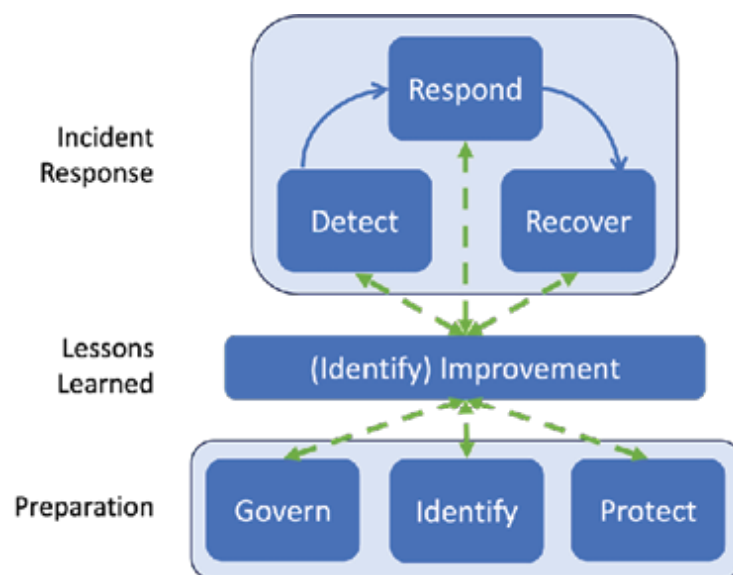


Figure 8.4: Incident response lifecycle ^[5]

Incident response policy provides a structured approach to dealing with security incidents and disruptions. A well-defined policy helps organizations recover quickly and resume normal operations. They typically cover the purpose, objectives, scope, definitions of events and incidents, roles, responsibilities, authorities, guidelines for prioritization, maintaining and restoring operations, and other key actions, and performance measures^[5].

Procedures and processes should be created based on the policy to explain the process to be followed during incidents. Some of the key activities under policies and procedures include:

- **Playbook** development provides standardized, repeatable actions for specific incident types, ensuring consistency, speed, and alignment
- **Triage** procedures help prioritize incidents based on risk and potential impact.
- A structured **investigation** workflow helps ensure thorough analysis using telemetry, behavioral context, and real-time data. Evidence must be collected in a forensically sound manner, preserving logs, access records, and affected system snapshots to support investigations, legal proceedings, or audits.
- **Containment** strategies help isolate affected systems, prevent lateral movement, and inform segmentation and dynamic access controls.
- **Recovery** procedures help gradually restore services in a controlled manner.
- **Post-incident analysis** includes capturing lessons learned and identifying gaps in policy, detection process, and updates to playbooks and workflows. Conduct a root cause analysis to determine the initial vulnerability or event that enabled the incident to occur, examining system behavior, threat actor tactics, and control failures.

Roles and responsibilities

As organizations increasingly rely on internal and external teams, incident response is no longer a central responsibility; it is shared across multiple teams. The organization can also transfer some of its responsibilities to a provider. *Table 8.6* covers some of the roles and their responsibilities:^[5]

Role	Responsibilities

Leadership	Oversight of incident response, funding, and decision-making.
Legal	Review and advice on the implications of applicable laws and regulations, right to privacy, third-party contracts, and incident response.
Human resources	Implement practices such as pre-employment screening, employee onboarding, and joiners-movers-leavers policies.
Business or asset owners	Review priorities of their affected assets, status of response and recovery efforts, and assess implications to assets they own.
Application teams	Assist in and own the response and recovery of the application or service provided to internal and external consumers.
Infrastructure and network teams	Assist in and own the response and recovery of infrastructure and network services used to host the applications and services.
Third parties	Maintain incident response and security standards for services and products offered to the organization.

Table 8.6: Roles and responsibilities

It is essential to understand the shared responsibility model and ensure that respective roles and teams are aware of their expectations. When defining roles and responsibilities, it is vital to comprehend notification requirements, including who must be informed—both internally and externally—based on the severity and scope of the incident, as well as legal, regulatory, and customer-facing obligations.

Data classification

Data classification, which we covered in [Chapter 6, Secure Your Data](#), helps prioritize organizational assets and can be an essential consideration in incident response. [Table 8.7](#) provides examples of how classification affects incident response:

Classification	Incident response
Public data	Typically, not a big threat as data is public, but it is important to document the incident, assess its impact, ensure that it was limited to public data, and implement the fix along with updates to necessary procedures.
Internal data	This is more serious than public data measures, which include whatever is required with public data; in addition to notifying internal stakeholders, internal communications, and providing better training around data handling procedures.
Confidential data	This is a serious breach and requires incident response procedures to be followed with containment measures, detailed forensics, as well as legal and regulatory counsel, and implications. A big part is also communication outside the organization.

Restricted data	This is the most serious, and in addition to confidential data measures, it will need more stringent containment and isolation, in-depth forensics, and assessment of impact.

Table 8.7: Data classification and incident response

Real-world relevance

Here is an incident response policy and process for a sample incident:

- **Objective:** To detect and mitigate risks from compromised user credentials by enforcing automatic access revocation, investigation, and guided recovery steps.
- **Scope:** This policy applies to all employees and contractors accessing cloud services via federated identity (e.g., Azure Active Directory, Okta, Google Workspace).
- **Statement:** If a user's credentials are suspected or confirmed to be compromised, the user must be immediately blocked from all systems, and incident response procedures must be executed. Reinstatement of access must follow successful re-authentication, risk mitigation, and post-incident review.

Process overview

The SIEM system will generate an alert for such scenarios; [Figure 8.5](#) shows a sample alert for a suspicious login in Elastic SIEM. This would then trigger the process shown in [Table 8.8](#):

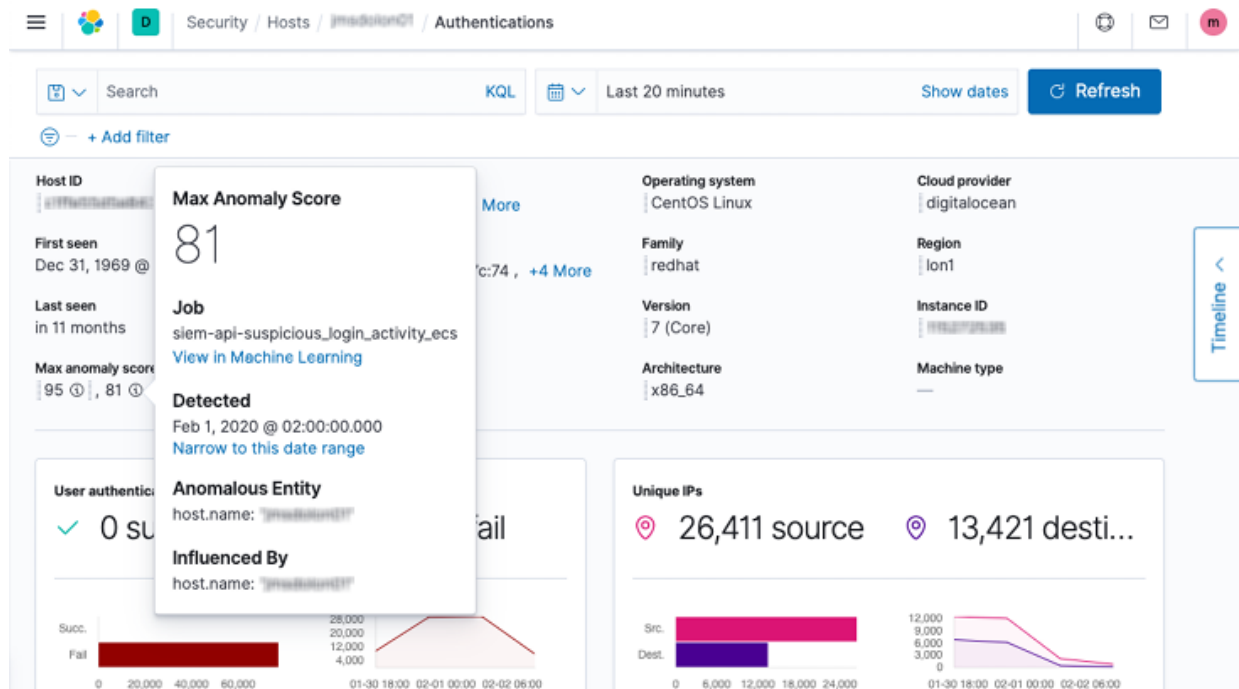


Figure 8.5: Authentication alert^[6]

Stage	Details
Detect	Use behavioral analytics or identity protection alerts (e.g., impossible travel, unusual login location, password spray) to trigger an alert. This could be done using, say, Microsoft Defender for Cloud, Elastic SIEM, or Splunk.
Triage	Determine severity: High (confirmed breach), Medium (suspicious behavior), Low (false positive likely). Defender for Cloud might do this automatically in most cases.
Investigate	Review login history, device posture, and access logs to determine the scope.
Contain	Disable account, revoke tokens/sessions, block access to sensitive apps. This response can be automated using Azure or Okta Graph APIs
Recover	Force password reset, reauthenticate with MFA, re-provision least-privilege access, send alerts and notifications.
Review and close	Generate the final report in the appropriate system. Document actions, update playbook, adjust detection rules if needed.

Table 8.8: Incident response process'

Team management

People are essential to security operations and compliance, requiring

effective team management. As most SOC teams work 24/7, shift planning must ensure well-structured rotations that prevent fatigue and maintain vigilance across all monitoring shifts. Training programs and skill development are vital, as they equip team members with up-to-date knowledge on current and emerging threats, tools, and evolving best practices. Regular, targeted training sessions and hands-on labs foster foundational and advanced competencies. Performance metrics, such as MTTD, MTTR, and alert accuracy, are tracked to evaluate individual and team effectiveness, drive accountability, and identify areas for improvement. Knowledge management practices, including maintaining detailed runbooks, playbooks, and knowledge bases, ensure that lessons learned and institutional expertise are easily accessible, supporting consistent and informed responses. Certification requirements, such as **Certified Information Systems Security Professional (CISSP)**, **Certified Information Security Manager (CISM)**, or vendor-specific credentials, are established to validate technical proficiency and encourage ongoing professional development. Together, these elements create a high-performing, resilient monitoring team capable of adapting to the dynamic demands of modern security environments.

Security awareness and learning

The success of Zero Trust Security depends on everyone in the organization; fostering a culture of security awareness throughout the organization, including with third parties, is crucial to the program's success. The primary objective is to offer learning opportunities to all and foster a shared understanding of the organization's risks and threats, as well as the role each individual must play in identifying, responding to, and mitigating those risks. This requires:

- Regular security and privacy training for all employees.
- Clear communication of security policies and best practices.
- Simulated phishing and social engineering exercises to test and improve awareness using tools like Gophish and Knowbe4.
- Encourage the reporting of suspicious activities or potential security incidents.
- Ongoing review and updates of security and privacy training programs

based on current and upcoming environmental threats and changes.

Backup and recovery

Recovery procedures aim to restore normal operations after breaches and incidents while maintaining data security. The exact details of recovery procedures differ based on the type of attack; for instance, ransomware that corrupts the data versus a DDoS attack will each need their specific ways of treatment and fixes. There are a few common aspects to all recovery procedures:

- Be prepared by implementing data redundancy and backup procedures based on availability, **recovery point objective (RPO)**, and **recovery time objective (RTO)** requirements. Secondly, procedures for different types of attacks and for each system must be documented. Regularly test the procedures through drills, tabletop exercises, and chaos tests.
- Service and data restoration should focus on recovering critical systems and data. When restoring, implement failover systems and procedures that are either automatically triggered or require little manual intervention.
- Communication is a big part of both internal and external processes. Internal communication ensures the recovery process is smooth and everyone knows what to do when restoring services quickly. External communication ensures people are aware of what to expect.
- Documenting and auditing all actions during recovery helps one understand the actions taken and areas for improvement. One can review them during post-incident postmortems and adopt measures to reduce the likelihood of the incident and its impact.

Backups

Organizations with regular, immutable backups stored offline can recover without significant data loss, ensuring data integrity and availability even during disruption. Most compliance frameworks we will cover in [Chapter 12, Policies, Frameworks, and Compliance](#) mandate regular backups to protect against data loss from attacks, hardware failures, and human errors. Automated backup solutions further streamline compliance by enforcing retention policies, geographical storage requirements, and audit-ready

documentation, reducing manual errors and ensuring consistency across hybrid environments. We discussed a few examples of data backup and recovery solutions as part of the secure data lifecycle in [Chapter 6, Secure Your data](#). Two types of data that need to be backed up are:

- **Configuration:** Ensure regular backups of critical configurations (such as network and infrastructure). Leveraging IaC ensures the configuration is always backed up in code repositories.
- **Transactional:** Backup your transactional data stores using capabilities provided by the datastores or third-party solutions.

By integrating backups into continuous monitoring frameworks, organizations gain real-time visibility into data protection gaps, ensuring alignment with compliance objectives.

Recovery and resilience testing

Defining incident response procedures and ensuring regular data backups alone do not prepare an organization or team for speedy recovery. **Disaster recovery (DR)** testing is essential to validate the effectiveness of the process and backups and meet compliance mandates. Testing usually involves simulating failure scenarios, outages, and attacks, and executing recovery procedures. Post the test recovery success rates, downtime, and data loss, metrics like MTTR and MTTD are tracked and documented to refine processes further. At the same time, automated reporting tools highlight consecutive backup failures or retention policy deviation. DR testing frequency and outcomes must align with enterprise and regulatory standards; most organizations conduct annual DR tests.

Automated chaos engineering tools like Chaos Monkey can be used to inject failures into systems, revealing vulnerabilities and testing response protocols for resilience testing. Game days are also another way of simulating real-world incidents to validate technical and human response capabilities, fostering a culture of preparedness. These exercises involve cross-functional teams (technology, security, operations) rehearsing scenarios like data breaches or network outages, with predefined objectives and a timeline. Conduct game days periodically to validate security controls; identify gaps in detection, response, communication, escalation procedures, and integrations; and enhance resilience against attacks, aligning with

frameworks like **Digital Operational Resilience Act (DORA)**.

Forensic analysis

A robust security posture requires the ability to scrutinize the detection and response to security breaches; this enables teams to identify, investigate, and respond to incidents systematically and understand the root cause(s). Forensics helps organizations trace the origins of threats, reconstruct timelines, and identify vulnerabilities that may be exploited. The process usually combines logs, network traffic, and artifacts with advanced analytics to identify anomalies, tampering, and unauthorized access. Monitoring platforms correlate telemetry across environments, apply multi-factor authentication, and enforce least-privilege access, ensuring all activities are auditable. However, it also requires meticulous tracking and auditing of actions performed during the response. Automated workflows and forensic tools streamline investigations, allowing security teams to validate digital artifacts, trace attack vectors, and mitigate risks proactively, ultimately helping improve incident response and security posture.

Process automation

Automation is a foundational element of most operational capabilities and is highly relevant in a Zero Trust security; it helps enforce consistency of controls, accelerates threat detection and response, and reduces the risk of manual errors. Automation plays a critical role across various areas in continuous monitoring and security operations, described as follows:

- Automated playbooks streamline incident response by executing pre-defined workflows for common threats, ensuring a rapid and standardized reaction to security events.
- Alert enrichment leverages integrated threat intelligence to provide analysts with critical context (for example, ownership of assets, severity of alert, runbook links, vulnerability details), improving the fidelity of alerts, making them more actionable, enabling faster and more informed decisions.
- Response orchestration coordinates actions across disparate security tools, ensuring policy enforcement and remediation are consistent and scalable throughout the enterprise.

- Case management tools such as ServiceNow track incidents across their entire workflow, support documentation, and collaboration features across teams, allowing for precise and up-to-date communication. Effective case management and workflow optimization (mostly built into the solutions) further enhance operational efficiency, enabling security teams to manage incidents from detection through resolution with minimal manual intervention.

Figure 8.6 shows a sample playbook within Microsoft Sentinel that uses Logic App workflows to automate incident response. The example blocks a user and their origin IP address based on an email approval.

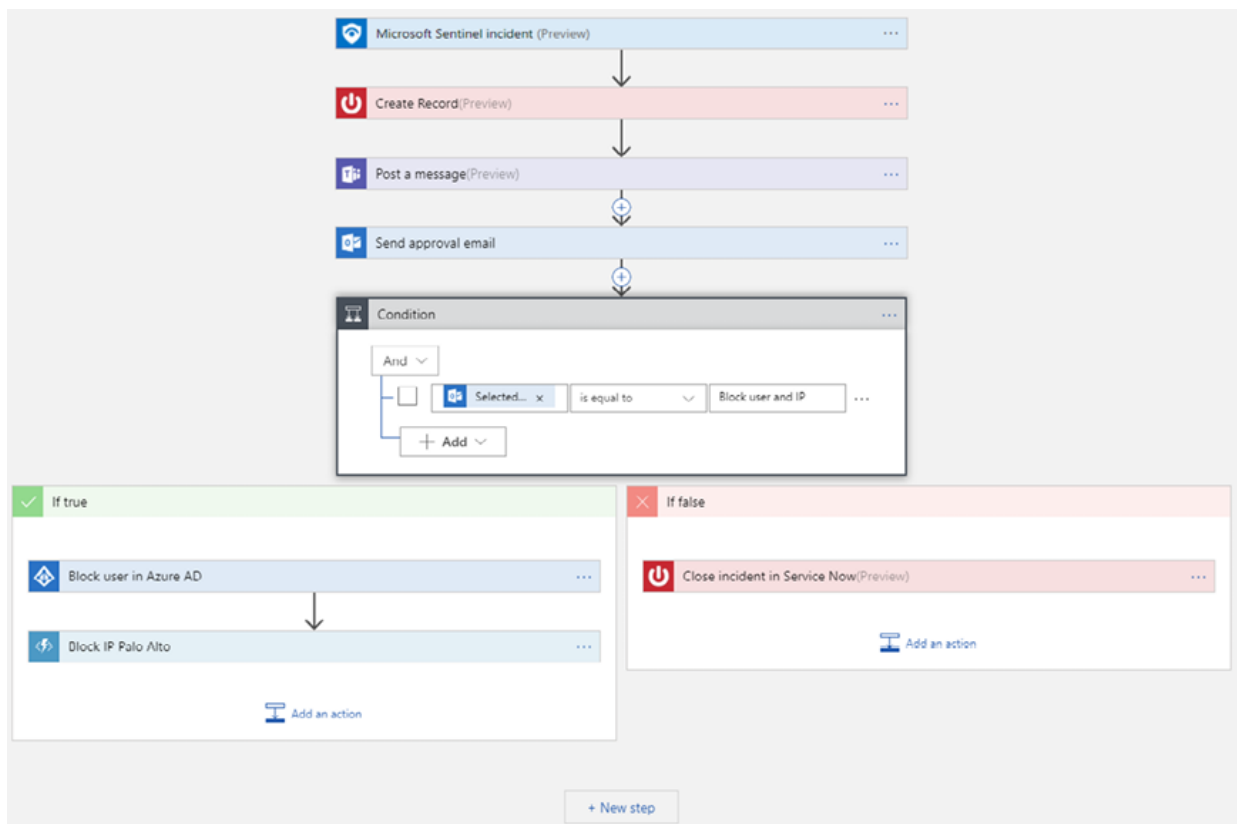


Figure 8.6: Microsoft Sentinel Logic App ^[7]

Real-world relevance

When implementing a new system, product, or application, you would follow the following steps for implementing continuous monitoring:

1. Define scope and objectives:

- a. Identify critical assets, compliance requirements (e.g., ISO 27001,

HIPAA), and business risk priorities.

b. Set goals: threat detection, audit readiness, data loss prevention, etc.

2. **Collect telemetry (logs and metrics) from all layers or stacks:**

a. **Endpoints:** Deploy **endpoint detection and response (EDR)** agents to monitor file, process, and user activity.

b. **Network:** Capture traffic data via intrusion detection and prevention systems and flow logs.

c. **Applications:** Integrate application performance monitoring and enable structured logging (e.g., JSON logs).

d. **Cloud and containers:** Use cloud-native services (e.g., AWS CloudTrail, GCP Audit Logs) and Kubernetes monitoring tools (e.g., Falco).

e. **IAM and identity logs:** Ingest single sign-on and multi-factor authentication data (e.g., from Okta, Azure AD).

f. **CI/CD pipelines:** Include secure code scanning tools and telemetry from build tools.

3. **Centralize logs and events:**

a. Use **log shippers** (e.g., Filebeat, Fluent Bit) to forward logs to a centralized location.

b. Feed logs into a **SIEM** (e.g., Splunk, Microsoft Sentinel) for normalization, storage, and correlation.

4. **Enable real-time detection:**

a. Define rules for known attack patterns (e.g., brute-force login attempts).

b. Apply UEBA and machine learning to detect anomalies.

c. Use threat intelligence to enrich events with external indicators.

5. **Automate response with SOAR:**

a. Implement playbooks to automatically respond (e.g., isolate host, disable account).

b. Integrate with ticketing systems (e.g., Jira, ServiceNow) for workflow orchestration.

6. **Visualize and monitor:**

a. Create dashboards to visualize real-time threats, system health, and

risk posture.

b. Schedule regular reporting for compliance and executive teams.

7. Review, test, and improve:

a. Run simulations (e.g., tabletop exercises, red team exercises, breach and attack simulations).

b. Regularly tune rules and alerts to reduce false positives.

Based on this flow, here is how a sample use case would be implemented:

- **Use case:** Suspicious login patterns in a trading platform.
- **Threat:** Repeated failed logins followed by a successful login from a new location.
- **Monitoring flow:**
 - Integrate identity and access management logs (from Azure AD, Okta) into your enterprise SIEM.
 - UEBA detects deviations in login behavior.
 - This triggers MFA revalidation and notifies the security operations center in case of repeated failed attempts.
- **Implementation guidance for engineers:**
 - Emit detailed login telemetry with user agent, IP, and device ID.
 - Implement consistent structured logging for all authentication flows.
 - Ensure the login API integrates with IAM event hooks.
 - Integrate logs into SIEM and tune thresholds for alerts or incidents (as needed)

Challenges and anti-patterns

Continuous monitoring is an essential part of detecting and mitigating threats and maintaining compliance. However, some practical challenges and anti-patterns could arise from the rush to monitor and alert everything. Let us discuss some of these challenges and anti-patterns:

- **Alert fatigue and false positives:** Excessive irrelevant alerts overwhelm analysts, causing critical threats to be missed or ignored.
- **Lack of contextual data correlation:** Isolated data sources hinder effective threat detection by failing to provide a complete picture of

security events.

- **Over-monitoring leading to privacy erosion:** Excessive surveillance can compromise user privacy and erode trust, undermining compliance and organizational culture.
- **Insider threats:** Malicious or negligent insiders remain a persistent risk, especially as organizations expand Zero Trust strategies and remote work increases access points
- **Limited visibility and observability:** The sheer volume of data, especially in cloud and multi-cloud environments, makes it challenging to maintain comprehensive visibility and timely detection of threats.
- **Geopolitical risks:** Nation-state actors and geopolitical tensions are contributing to a rise in targeted attacks, espionage, and disruption of operations, particularly in the critical infrastructure sector
- **Regulatory and compliance challenges and changes:** Fragmented and evolving cybersecurity regulations across regions create compliance burdens and complicate monitoring strategies for global organizations.

Emerging technologies

As covered in most of the chapters before, artificial intelligence has a profound impact in this space by helping continuously analyze anomalies and suspicious patterns for real-time detection, reducing false positives, and driving autonomous detection and response. Integrating various tools with DevSecOps pipelines also enhances teams' ability to proactively identify and manage security risks throughout the software development and release process. In addition, the following emerging technologies are also affecting this space:

- **Edge and IoT vulnerabilities:** The expansion of IoT and edge devices increases the attack surface, often introducing poorly secured endpoints that are difficult to monitor and protect. They are typically harder to monitor and maintain, as they can be installed in physically insecure and remote locations, subject to tampering, theft, and unauthorized access. Upgrading firmware and adopting modern, secure protocols is also challenging.

- **Cloud-based and integrated security platforms:** The shift to cloud-native, converged security management platforms enables organizations to monitor distributed environments, integrate data from information technology and operational systems, and orchestrate responses remotely and at scale. Most systems, like Microsoft Sentinel, support hybrid environments, allowing integrations across cloud providers and on-premises data centers. This enables organizations to use a single integrated platform for all their monitoring needs.
- **SOAR platforms:** Such as Splunk SOAR and Microsoft Sentinel, play a central role by automating repetitive and manual tasks such as alert enrichment, response orchestration, and case management. This alleviates alert fatigue and allows security teams to focus on higher-value analysis. By embracing process automation, Zero Trust environments achieve proactive, scalable, and resilient security operations that adapt to evolving threats and business needs.

Guidelines and checklists

Let us summarize some specific guidelines and checklists for the three roles typically responsible for maintaining the security posture of products and applications built. Use these with the checklists to secure identities and access to systems you create or manage.

For developers and engineers

This subsection summarizes guidance for anyone developing software or solutions for customers (internal or external). Product managers, designers, and architects can also use this to better understand the key principles that need to be adopted by their team or features to be built:

- **Monitoring and logs:**
 - Generate security logs for any sensitive events (such as authentication, authorization, data access, validation failures, config changes, and errors) within your applications.
 - Ensure appropriate log levels and correlation identifiers such as user, session, request, and transaction identifiers.

- Avoid or mask sensitive data in debug logging.
- Generate an audit trail of essential activities for security, compliance, and forensics.
- **Security testing:** Automate security testing and verification across these areas:
 - Automate security testing (static and dynamic application security tests) and integrate with continuous integration and deployment pipelines, capture or integrate logs with the monitoring infrastructure.
 - Automate functional and non-functional tests and use this to monitor anomalous behaviors.
 - Validate post-deployment security controls (e.g., secrets rotation, TLS enforcement).
 - Vulnerability management and resolution.
- **Application monitoring:** Generate events and monitor the following aspects of the application and data to ensure:
 - User, device, and session behavior, including login and logout timestamps, locations, and request volumes.
 - Authorization and access attempts, including privileged user access.
 - Input and request validation failures or rejections.
 - Data activity and audit trails; sensitive operations, exports, downloads, bulk data access, anonymizations, and deletions.
 - Configuration and code changes to application configurations, feature toggles, and deployment events, and access to secrets.
 - **API and integration:** Monitor abnormal API calls, high volumes, cross-service requests, and deprecated or vulnerable endpoints.
 - Compliance-specific monitoring, such as data retention and deletion, access to sensitive data, and consents.
- **Security operations:**
 - Ensure logs feed into centralized monitoring platforms and SIEM.
 - Identify and set up use cases for alerting for key application use cases.
 - Automate playbooks for security alerts.

For infrastructure engineers

This subsection summarizes guidance for anyone who deals with infrastructure (either on-premises or cloud):

- **Monitoring infrastructure:**
 - Configure access controls, audit trails, and integrate into a centralized logging and monitoring system
 - Centralize logs from apps, servers, databases, APIs, and firewalls. Use log shippers (e.g., Fluentd, Beats).
 - Implement and monitor EDR for virtual machines, containers, and workstations.
 - Deploy SIEM tools with necessary integrations to correlate logs and detect anomalies.
 - Continuously audit IAM roles, storage permissions, and API gateways
- **Integrate and automate infrastructure and runtime security events and logs, including:**
 - Application startup/shutdown logs.
 - Container or server resource abuse.
 - Monitor unusual network traffic, API gateways, port scanning, etc.
 - Access to secrets stores (e.g., AWS Secrets Manager, HashiCorp Vault).
 - Unexpected service connections (e.g., to unauthorized databases).
 - Vulnerability scan results and remediation logs.
 - Security patch and library update logs.
- **Configuration review and security testing:**
 - Monitor for configuration drift and outdated dependencies in base images or infra templates.
 - Automate security testing and integrate with continuous integration and deployment pipelines
 - Set up regular access review audits.
 - Perform privilege escalation testing, SAST, and DAST as part of

pipelines.

- Implement static analysis of infrastructure code using tools like CheckOV, Terraform Sentinel.
- **Alerting and automation:**
 - Configure alerts for any anomalies across the network and infrastructure estate.
 - Automate device and system health checks and patching.
- **Security operations:**
 - Monitor key metrics around authentication, access patterns, and failures.
 - Integrate SOAR platforms to automate threat containment (e.g., block malicious IPs).
 - Automate playbooks for security alerts and response procedures.
 - Conduct regular drills and simulations of incidents and remediate any identified issues and vulnerabilities.

For leaders

This subsection summarizes guidance for anyone who leads or manages a team, including product managers, designers, and architects. It also benefits people by supporting organizational functions like risk and compliance that interact with product teams:

- **Strategy and planning:**
 - Prioritize security and privacy-by-design by ensuring continuous monitoring is part of every feature, balance monitoring with data minimization and encryption to avoid over-surveillance.
 - **Foster cross-team collaboration:** Align security, development, infrastructure, and operations to standardize monitoring workflows.
 - **Define Zero Trust KPIs:** Track metrics like mean failed login rates, MTTD, false positives, and policy compliance.
 - Set up the right team structure and include sufficient coverage and an on-call rota.
- **Compliance and risk:**

- Ensure adherence to appropriate compliance standards and policies.
- Conduct periodic third-party audits: Ensure vendors comply with Zero Trust policies (e.g., FedRAMP, NIST).
- Raise appropriate risk and review risk mitigations and exceptions regularly.
- **Monitoring and incident response:**
 - Ensure well-defined policies and procedures exist for incident response for handling security breaches, backup, and recovery.
 - Require teams to build observability (logs, metrics, traces) into all systems.
 - Review incident response plans. Conduct tabletop exercises to test breach containment and recovery processes.
 - Conduct regular drills and simulations.
 - Enforce minimum standards and monitor service level agreements for log completeness, retention, and response times.
- **Security culture, training, and awareness:**
 - Plan regular security training for teams.
 - Identify security champion(s) and raise awareness, foster a culture of security awareness.
 - Stay informed about emerging threats and best practices.

Conclusion

Continuous protective monitoring, which involves real-time monitoring of networks, users, and devices, is crucial for Zero Trust security. The Colonial Pipeline case study illustrates the consequences of a ransomware attack stemming from a dormant VPN account lacking MFA, resulting in substantial economic losses and operational disruptions. The rest of the chapter covered monitoring strategy, infrastructure, incident response, team management, reporting, and metrics. In the next chapter, we will discuss container security and provide an example of implementing the key pillars of the Zero Trust architecture.

Points to remember

- Continuous monitoring is crucial for identifying emerging threats and informing the development of effective security measures.
- A continuous monitoring system must always be on, real-time, continuous, holistic, accessible, offer rapid response, generate awareness, and encourage collaboration.
- The monitoring strategy defines the approach and objectives for continuously observing systems to detect threats and ensure security.
- Monitoring infrastructure tools and technologies deployed to collect, aggregate, and analyze security data across the organization, as well as SIEM, is the foundational technology. It should typically be integrated with all other systems and generate security logs and events.
- Analysis and reporting transform raw security data into actionable insights, communicating findings through dashboards and reports for executive leaders, teams, and compliance purposes.
- Compliance and audit ensure adherence to regulatory requirements and validate security controls through systematic reviews and documentation.
- Security operations include day-to-day security activities, threat detection, and response to maintain organizational resilience, either via a security operations center or MDR.
- Incident response outlines processes for identifying, containing, and resolving security incidents to minimize impact.
- Team management, organizing, training, and supporting security personnel and teams to maximize operational effectiveness, including 24/7 support.
- Security awareness and learning are necessary to educate staff on security best practices and emerging threats to foster a security-conscious culture.
- Backup and recovery are crucial for safeguarding data and restoring operations quickly in the event of disruptions or attacks.
- Forensic analysis investigates security incidents by collecting and examining digital evidence to determine root causes and attribution.

- Process automation uses technology to streamline repetitive security tasks, improving efficiency and response times.
- Emerging technologies such as artificial intelligence, IoT, cloud-based integrated security platforms, and SOAR will continue to have an impact on continuous monitoring and response.

References

1. https://iaeme.com/MasterAdmin/Journal_uploads/IJRCAIT/VOLUME_7_ISSUE_2/IJRCAIT_07_02_110.pdf
2. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-137.pdf>
3. <https://www.exclusive-networks.com/ie/wp-content/uploads/sites/19/2021/07/The-Essential-Guide-to-SIEM.pdf>
4. <https://learn.microsoft.com/en-us/azure/sentinel/get-visibility>
5. <https://csrc.nist.gov/pubs/sp/800/61/r3/final>
6. <https://www.elastic.co/security/siem>
7. <https://learn.microsoft.com/en-us/azure/sentinel/overview?tabs=azure-portal>

CHAPTER 9

Container Security

Introduction

Currently, cloud is considered table stakes, and very few tend to operate without cloud. All new businesses launching today operate on the cloud and use different flavors of services offered by cloud providers. Containers are yet another technology that emerged as a logical evolution of virtualization technologies, creating a lightweight alternative. Containers have become a fundamental building block for modern enterprise applications.

Unlike virtual servers, containers are short-lived and rapidly created and destroyed. They have a smaller footprint and are used with the application of modern microservices architecture. There can be many containers, sometimes scaling to thousands or hundreds of thousands, along with increased communication. This dynamic and distributed nature introduces new security challenges. Traditional perimeter-based security models are insufficient, as containers often communicate across ephemeral networks and can be orchestrated at a massive scale. The same pillars of Zero Trust that we covered so far can be applied to container-based environments.

By combining containerization with Zero Trust, organizations can achieve robust, scalable security that aligns with the demands of modern cloud-native applications. This ensures that only authorized actions are permitted and minimizes the impact of potential breaches.

Structure

The following topics are covered in the chapter:

- Overview of containers
- Security of containers
- Use case introduction
- Applying the Zero Trust approach

Objectives

Containers have revolutionized the way modern applications are built, shipped, and deployed, offering agility and scalability that traditional computing models have struggled to provide. This chapter begins with a brief history and overview of containers, which should clarify why they are so popular. We will then discuss a practical use case to guide our application of security principles throughout the chapter. The Zero Trust model is at the heart of this discussion, which assumes no implicit trust and demands strict verification across all system layers. We examine how Zero Trust can be effectively applied to container security by focusing on six key pillars discussed in this book. Together, these principles form a robust, proactive security posture for containerized environments in today's cloud-native world.

Note that containers are a complex topic, and we will not cover the full details of how the technology works or its usage. The chapter provides an overview of key concepts, focusing on core controls and elements of Zero Trust security to secure the software you will deploy on top.

Overview of containers

Let us begin by understanding how infrastructure and virtualization technologies evolved, leading to the development of containers, followed by an explanation of what containers are, before diving into how to secure them.

Brief history

The history of containers is closely tied to the evolution of virtualization and the need for systems to distribute tasks across multiple units for improved scalability. Traditional virtualization, which originated in the 1950s and 1960s, utilized **virtual machines (VMs)** to emulate entire physical computers, each running its own operating system and applications on top of a hypervisor. This approach provided strong isolation but was resource-intensive, as every virtual machine included a full guest OS. This further led to the emergence of containers. [Figure 9.1](#) provides a summary of the history of containers^[1, 2, 3]:

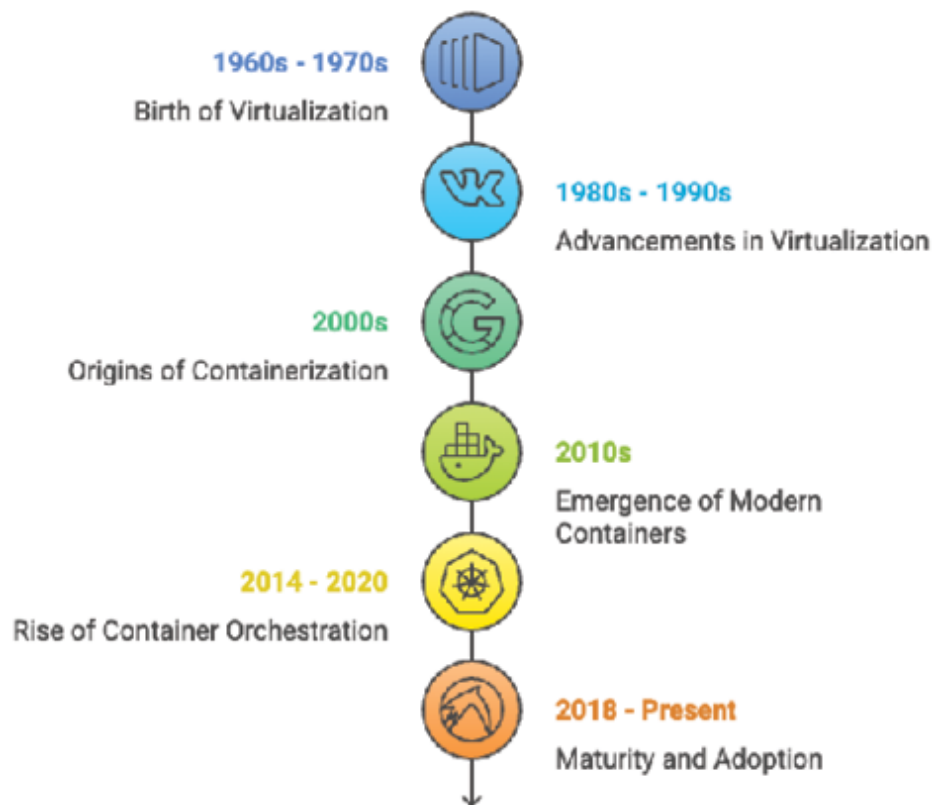


Figure 9.1: Timeline for the evolution of containerization

- **1960s—1970s: The birth of virtualization**
 - **1960s**: The concept of virtualization emerges with the development of time-sharing systems, allowing multiple users to share computing resources.
 - **1972**: IBM introduced the CP-40 mainframe, a pioneering virtual

machine system allowing multiple operating systems to run concurrently on a single mainframe.

- **1979:** The *chroot* system call was introduced to help segregate file access by process.
- **1980s—1990s: Advancements in virtualization**
 - **1980:** IBM introduced the first commercial virtual machine operating system, VM/370, allowing multiple operating system instances to run on hardware.
 - **1998:** VMware was founded and presented Workstation 1.0, allowing a user to run multiple operating systems like virtual machines on a single personal computer.
- **2000s: Origins of containerization**
 - **2006:** Google introduces *process containers* internally, using lightweight process isolation techniques for application deployment.
 - **2008: Linux containers (LXC)** are released using cgroups and Linux namespaces as the first complete implementation of a Linux container manager. They provide a user space interface for the Linux kernel containment features, allowing multiple isolated Linux systems to run on a single host.
- **2010s: Emergence of modern containers**
 - **2013: Let me contain that for you (LMCTFY)** kicked off as an open-source version of Google's container stack.
 - **March 2013:** Docker is launched. It simplifies the process of creating, deploying, and managing containers and popularizes the use of containers in software development and DevOps practices.
- **2014—2020: The rise of container orchestration**
 - **2014:** Kubernetes was introduced by *Google*, providing a powerful platform for automating the deployment, scaling, and management of containerized applications.
 - **2015:** The **Open Container Initiative (OCI)** establishes common standards for container formats and runtimes, promoting interoperability across different container platforms.

- **2016:** The Cloud Native Computing Foundation adopted Kubernetes (2016), and VMware, AWS, Azure, and Docker announced their support in 2017.
- **2018—Present: Maturity and adoption**
 - **2018:** Major cloud providers, AWS, Azure, and Google Cloud, enhance container services, making it easier for organizations to adopt containerization.
 - **2020:** The COVID-19 pandemic accelerates digital transformation, increasing reliance on cloud and containerized applications. 2022 was a significant year for cloud-native technologies, with Kubernetes as a preferred platform for digital transformation and cloud-native workloads.

Containers and container platforms

Containers are a modern approach to software packaging and deployment, designed to address the challenges of portability, consistency, and efficiency in application delivery. Unlike traditional software deployments on VMs, containers encapsulate an application and all its dependencies into a single, lightweight, and portable unit that can run consistently across diverse environments, from a developer's laptop to large-scale cloud infrastructures.

The official definition of a container (from Docker) is a standard unit of software that packages code and all its dependencies so that the application runs quickly and reliably from one computing platform to another. [Figure 9.2](#) shows the various layers involved. At the bottom is the infrastructure, which could be your physical servers or virtual machines. This is followed by the host operating system running on the infrastructure—the container engine (such as Docker, containerd). The container engine manages the lifecycle of a container, provides networking for containers, handles its resources (processing, memory, storage), and supports running multiple containers on top. Every container is typically based on a container image that is pre-built and available in a container registry; the container engine usually fetches a container from a registry to create an instance called the container:

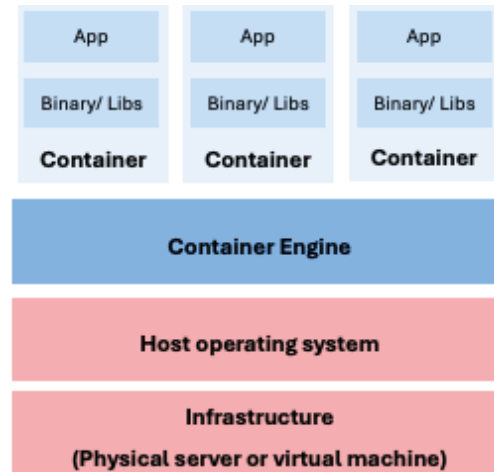


Figure 9.2: Containers

A container platform allows you to manage containerized applications and includes orchestration, monitoring, governance, security, and automation of the containers. They typically come in various flavors. Currently, the most common implementation is the platform as a service offerings from cloud providers, such as AWS Elastic Kubernetes Service, Azure Kubernetes Service, and GCP Kubernetes Engine. In addition, cloud providers also offer serverless options for running containers, such as AWS Fargate, Azure Container Apps, and GCP Cloud Run. [Figure 9.3](#) shows a container platform cluster running on multiple virtual machines. The application containers are spread across this cluster, and the container platform manages their lifecycle, ensuring scheduling, reduced downtime, high availability, networking, security, and most other infrastructure-level considerations.

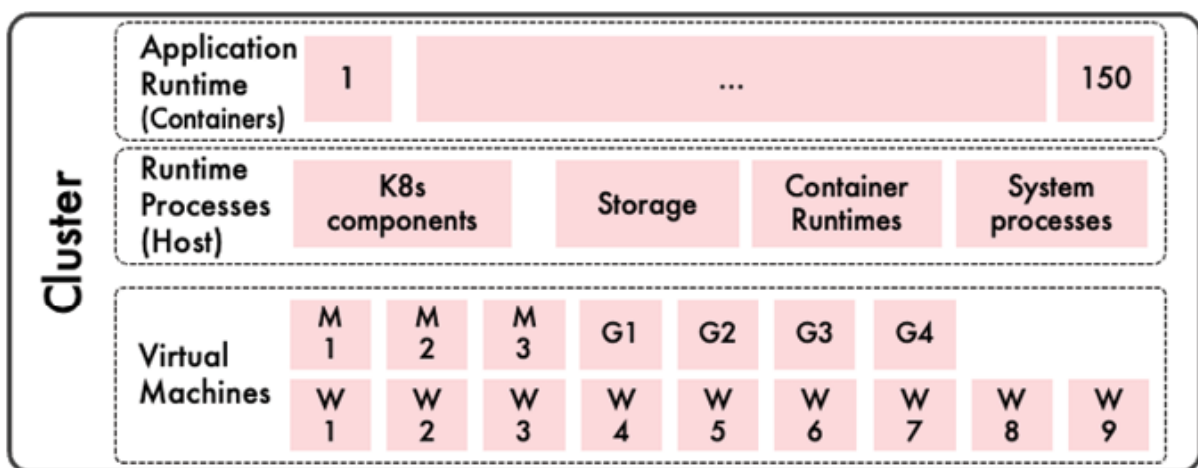


Figure 9.3: Container platform components

Table 9.1 summarizes the key terms used in the context of containers and container platforms:

Term	Definition
Container	A lightweight, isolated executable package containing an application and its dependencies.
Container image	A read-only template used to create containers; contains app code, libraries, and dependencies.
Container runtime	Software that runs and manages containers on a host OS (e.g., Docker, containerd).
Orchestration	Automated management of container deployment, scaling, and networking (e.g., Kubernetes).
Host OS	The underlying operating system on which containers run and share the kernel.
Registry	A repository for storing and distributing container images (e.g., Docker Hub).
Microservices	An architectural style where applications are composed of small, independent services, often deployed in containers.
Cluster	A group of nodes (machines) managed together, running containerized applications via a control plane.
Node	A physical or virtual machine in a cluster that runs containers and is managed by the control plane.
Pod	The smallest deployable unit in Kubernetes; a group of one or more containers sharing storage/network.
Container	A lightweight, portable executable image with application code and dependencies, running inside pods.
Deployment	A resource that manages the desired state, scaling, and updates of pods and containers.
ReplicaSet	Ensures a specified number of pod replicas are running at any given time.
StatefulSet	Manages deployment and scaling of stateful pods, ensuring unique identities and persistent storage.
Service	An abstraction that defines a logical set of pods and a policy for accessing them (network endpoint).
Ingress	Manages external access to services, typically HTTP, providing load balancing and routing.
Namespace	Provides logical partitioning of cluster resources for different users or projects.
Volume	Storage resource attached to pods, persisting data beyond container lifecycle.

Table 9.1: Container terminologies

Security of containers

Along with all the benefits, containers and container platforms bring along some complexities due to the unique way this work is done. The following is a list of some of these:

- **Shared kernel:** Since containers share the host kernel, they can be affected by kernel vulnerabilities and exploits. Kernel isolation with Kata Containers, gVisor enhances Zero Trust runtime by decoupling execution environment from the host kernel.
- **Multi-tenancy:** By definition, containers are multi-tenanted, and a single host can run multiple containers, increasing the attack surface and ability to compromise due to lateral movement.
- **Dynamic nature:** Since the runtime maintains the container lifecycle and recreates it in case a container crashes or is redeployed, it is harder to attack, but equally, malicious behavior is more challenging to track.
- **Networking complexity:** Due to virtualized overlay networks and the density of applications running in environments, monitoring, controlling, or detecting suspicious traffic is much more complicated.
- **Supply chain risks:** Container registries are the primary source of the images used to run containers, and due to the proliferation of public registries, proving provenance and integrity can be more challenging. It is also much easier for malicious code and backdoors to be released and installed without proper controls.
- **Misconfigurations:** Most configurations are managed via command line and configuration as code, and without proper checks and balances, this can lead to misconfigurations, some of which could have a broader impact across multiple applications across a cluster.

Table 9.2 summarizes some common threats to containers and container platforms:

Attack	Details
Breakout attacks	Escaping container isolation to access the host (e.g., via kernel exploits).
Supply chain risks	Compromised or outdated base images or malicious components in public registries.
Misconfigured settings	Excessive privileges (e.g., root access), exposed ports, or insecure

	networks.
Runtime compromise	Malicious code injection or lateral movement through unsegmented networks.
API server exposure	Unauthenticated access to the cluster's control plane.
RBAC misconfigurations	Overly permissive roles enabling lateral movement.
Default settings	Unencrypted secrets, root-run containers, or public dashboards.

Table 9.2: Key threats and attacks on container platforms

Use case introduction

We will use a typical digital payment processing application, which is quite common within financial services. The solution's scope is to allow users to log in, initiate a payment using a payment gateway we support, perform fraud checks to detect anomalies, and send a notification based on the outcome. For simplicity, this use case does not include a front-end, which could be run as a container or a static single-page application delivered from an object storage such as AWS Simple Storage Service. [Figure 9.4](#) shows a high-level view of the key components in this application:

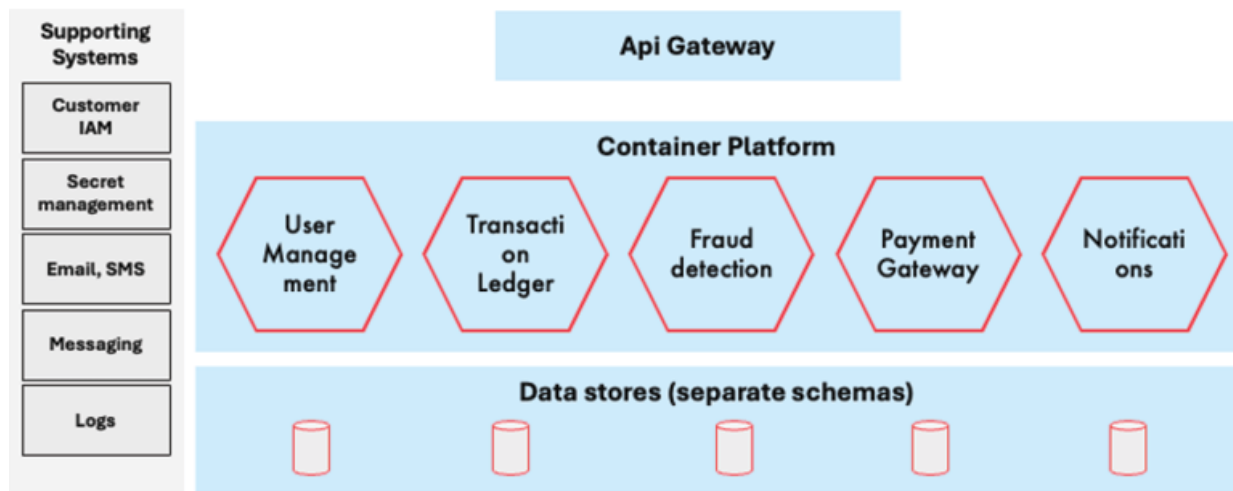


Figure 9.4: Use case architecture

The core microservices or components and their responsibilities are described in [Table 9.3](#). In reality, each of these domains (say, user management or payment gateway) is much more complex and can be comprised of multiple microservices:

Microservice	Details
User management	Manages customer identities, authentication, and authorization
Payment transaction	Maintains immutable transaction history
Fraud detection	Uses real-time analytics and machine learning to flag suspicious activity
Payment gateway	Handles card/bank authentication and payment routing
Notifications	Sends alerts and transaction confirmations via email/SMS

Table 9.3: *Digital payment processing application microservice components*

The technical stack required to deploy these microservices can be deployed on the following capabilities in [Table 9.4](#) across various cloud providers. Note that the runtime here focuses on services that support containers; there are many. More options are available from the cloud providers, but the focus here is on containers:

Capability	AWS	Azure	GCP
Identity	AWS Cognito	Azure AD B2C	Identity Platform
Compute runtime	Elastic Kubernetes Service or App Runner	Azure Kubernetes Service or Container Apps	Google Kubernetes Engine/Cloud Run
Databases	Amazon Aurora	Azure Postgres	CloudSQL
Object store	AWS S3	Azure Blob Storage	GCP Object Storage
API gateway	AWS API Gateway	Azure APIM	GCP Apigee
Integration	AWS Simple Queue Service, EventBridge	Azure EventGrid	GCP Pub/Sub
Notification	Amazon SES	Azure communication	Firebase cloud messaging
Logs	CloudWatch Logs	Azure Monitor	Cloud Logging
Secrets	Secrets manager	Key Vault	Secret manager
Source control	GitHub, GitLab	GitHub, GitLab	GitHub, GitLab
Build and deploy	GitHub, GitLab	GitHub, GitLab	GitHub, GitLab

Table 9.4: *Technical stack across cloud providers*

In hybrid environments, where organizations have a combination of on-premise infrastructure, services such as AWS EKS Anywhere and Azure Arch offer consistent control plane that allow extending security capabilities offered by cloud providers.

Applying the Zero Trust approach

Now that we have seen the overall stack, let us discuss how to apply the various pillars of Zero Trust we have learned so far. We will take each of the six pillars and address some of the key areas of the security application. This is not meant to be exhaustive, and readers are encouraged to reference individual topics for a comprehensive understanding of capabilities within each pillar. However, this chapter will provide an approach to applying these concepts in an actual use case.

Secure your code

Let us start by revisiting the core concepts of securing your code. *Figure 9.5* shows the key steps for the same:

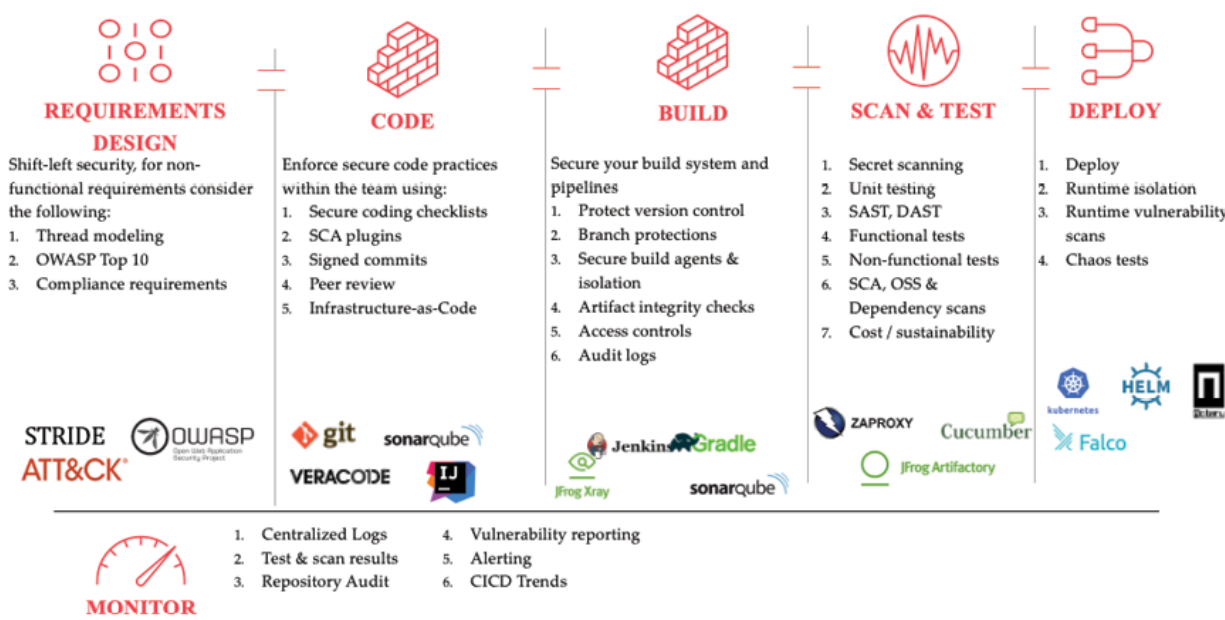


Figure 9.5: Core concepts of secure your code

Table 9.5 outlines the key steps in the process, along with the procedures and tools for implementing them:

Step	Implementation details

Step	Implementation details
Secure requirements and design	Use threat modeling and Open Web Application Security Project (OWASP) top 10 vulnerabilities to define key requirements. When it comes to compliance with the payment gateway integration service, we will need special consideration. We will plan to integrate a payment gateway and will not capture any card information on our platform.
Secure code	The first setup is to maintain code in a version control system; we will choose GitHub with security controls (GitHub advanced security) and branch protections enabled to protect any deployment branches. Terraform should be used for infrastructure as code and pipeline automation will be using GitHub Actions.
Secure build	- Secure build agents by using our own private runners based on customer secure images. - Package management uses tools such as Gradle to build your application library and only allows imports of libraries from secure sources. - Access controls integrate all build tools with the enterprise identity solution and implement appropriate access controls. - Secret management: All secrets should be managed in secret manager. - Minimal base images and reducing the attack surface. - Managing OS packages and dependencies. - Follow best practices for Dockerfile and image hardening using CIS Benchmarks.
Secure scan and test	- Secrets scanning is part of GitHub's advanced security. - Select the unit testing framework of choice based on language. - SCA, OSS, and dependency scanning: JFrog Xray - Static code analysis: Sonar, Snyk - Dynamic application security: Burp Suite, OWASP ZAP - Container security: GCPArtifact scanning - Artifact integrity scans use checksums. - Use secure image registries and trusted sources. - Functional and performance test tools of your choice. - Generate and integrate SBOMs and continuously monitor dependencies.
Secure deploy	Environment provisioning: Use Terraform to configure infrastructure (Cloud SQL, Cloud Run), use tools such as Checkov for secure configurations (we will discuss a bit more in the subsection <i>vulnerability management</i>). Testing, monitoring, and rollback procedures.
Monitoring and access	- Integrate all logs into the central logging and monitoring solution. - Implement appropriate access controls for users of the configuration, build system, and application runtimes.

Table 9.5: Secure code build and deploy steps and implementation

The following is a sample GitHub continuous integration workflow pipeline that includes some of the key steps to demonstrate the key controls. Cross-reference the latest documentation to build a working example:

```
on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
env:
  IMAGE_NAME: <microservice-name>
  SERVICE_NAME: <microservice-name>
jobs:
  build-and-test:
    name: Build and Unit Test
    ... build process based on language      ...
  sonar-scan:
    name: Static Code Analysis (SonarCloud)
    runs-on: ubuntu-latest
    needs: build-and-test
  ...
  xray-scan:
    name: SCA & Dependency Scan (JFrog Xray)
  ...
  build-container:
    name: Build Docker Image
    ... docker build and push to registry...
  artifact-attestation:
    name: Generate artifact attestation for integrity...
  zap-dast:
    name: ZAP scan
    steps:
      - name: ZAP Scan
        uses: zaproxy/action-full-scan@v0.12.0
        with:
```

```
        target: 'http://url-to-service' # Replace
with actual url
        ...other options
... deploy steps ...
  functional-tests:
    name: Run Functional Tests
... functional tests and upload results ...
```

Secure your identities

The process for securing user identities accessing applications depends on the types of identities. We will cover the various techniques and implementation details here, but first let us understand three types of identities to consider, as shown *Figure 9.6*:

- **Internal user** or colleague identities would be used to manage the infrastructure and systems and the application lifecycle (software development, build, and deployment). This would rely on Microsoft Entra or Google Identity Platform. Typically, the identity platform for internal users is an enterprise-wide solution that remains unchanged with the application; most internal applications will integrate with the central identity solution.
- The **external user**, the end-user or customer, registers and makes payments on our platform. We will integrate platforms such as Amazon Cognito, Azure AD B2C, ForgeRock, Okta, and Firebase Authentication for this. Solutions for this should also typically be enterprise-wide, and it is best to integrate with the same platform across all applications. This way, end users get a consistent experience when they move from one application to another. However, the chosen platform will have to be integrated into your application.
- **System or service identities** that applications access or invoke other applications or systems. For instance, the payment transaction microservice is called the payment gateway service, which makes the transaction mTLS via a service mesh.

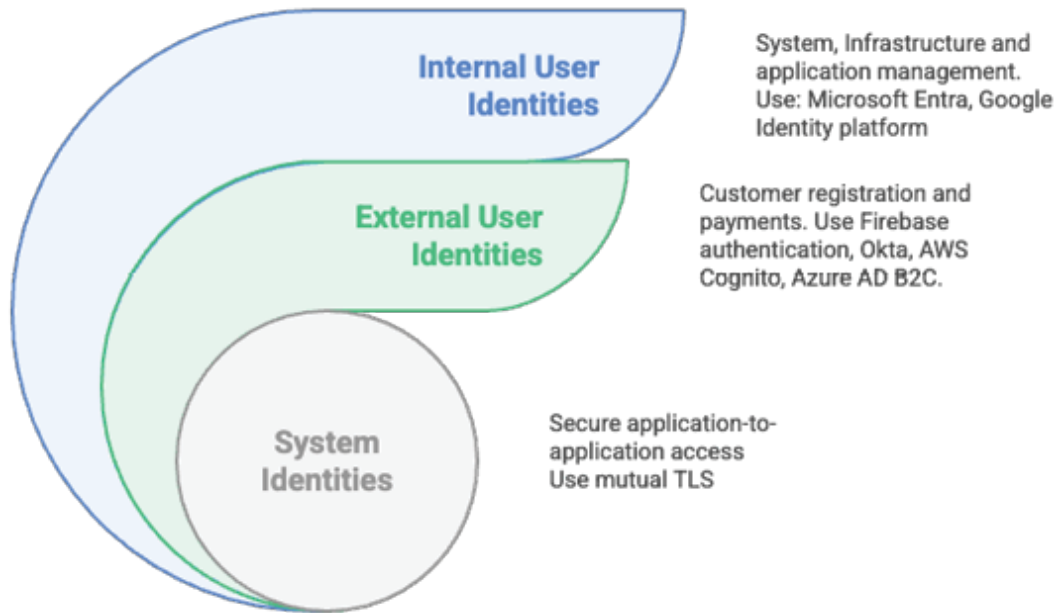


Figure 9.6: Types of identities and how to secure them

An overview of the steps to secure identities, along with corresponding implementation details, is covered in [Table 9.6](#):

Step	Implementation details
Strong authentication	For stronger authentication, it is best not to build the authentication yourselves but to integrate an existing identity platform that supports key requirements such as password policies, multi-factor authentication, support for well-known protocols OAuth 2.0, SAML2, etc. Use the solutions mentioned in the previous section for the three different types of identities (internal, external, and system).
Access controls	These are also typically existing policies such as least privilege access, role- and attribute-based access controls that should already be in place. If the application requires an administration interface accessible to enterprise or company users, implement appropriate authorization and role-based access controls.
Securing privileged access	Mainly for internal identities and it requires integration of a privileged access management system. Secure remote access using secure workstations when accessing production data.
Incident response and recovery	For internal users, implement break-glass procedures and a fallback for privileged access for critical operations with full auditing.
Federated identities	To support federated identities such as Google, Microsoft, Facebook for customer identities, use standards-based identity federation enabling secure logins via Google and other OIDC providers.

Identity governance	For internal users, automating identity lifecycle and enforcing regular access validation through Joiner-Mover-Leaver processes and ongoing reviews.
Securing APIs and systems	Secure service-to-service communication using scoped tokens and mutual TLS verification.
Policy design	Enforce multi-factor authentication, stronger passwords, idle session timeouts, identity lifecycle, least privilege access, role-based access controls, and segregation of duties.
Continuous monitoring	Comprehensive logging and audit trails within your application for all critical activities (successful and failed logins, password resets), integrate with SIEM, automated alerts, access patterns, anomaly detection.

Table 9.6: *Secure your identities steps and implementation*

Let us cover an example that shows how to secure customer identities (external):

1. Generate the Firebase authentication keys and integrate Firebase authentication into your front-end login screen using the software development kit.
2. Next, we would ensure that all our microservices only allow authorized requests. To that end, we would build an authentication filter to decode and verify the token.
3. Secure your microservice endpoints using the authentication filter.

The following code demonstrates how the Firebase authentication filter can be added to specific API-protected paths:

```
public class SecurityConfig {
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity
http) throws Exception {
        http
            .csrf(csrf -> csrf.disable())
            .authorizeHttpRequests(auth -> auth

.requestMatchers("/api/public/**").permitAll()

.requestMatchers("/api/protected/**").authenticated()
            )
            .addFilterBefore(new
```

```

FirebaseAuthenticationFilter(),

org.springframework.security.web.authentication.Username
PasswordAuthenticationFilter.class);
    return http.build();
}
}

```

Secure your network

The first step is to design the network and its segmentation, based on everything we have learned. We will use the following approach to create our network. *Figure 9.7* shows a simplified high-level view of these controls. It does not cover all network devices and appliances in depth, but it covers the main components and controls:

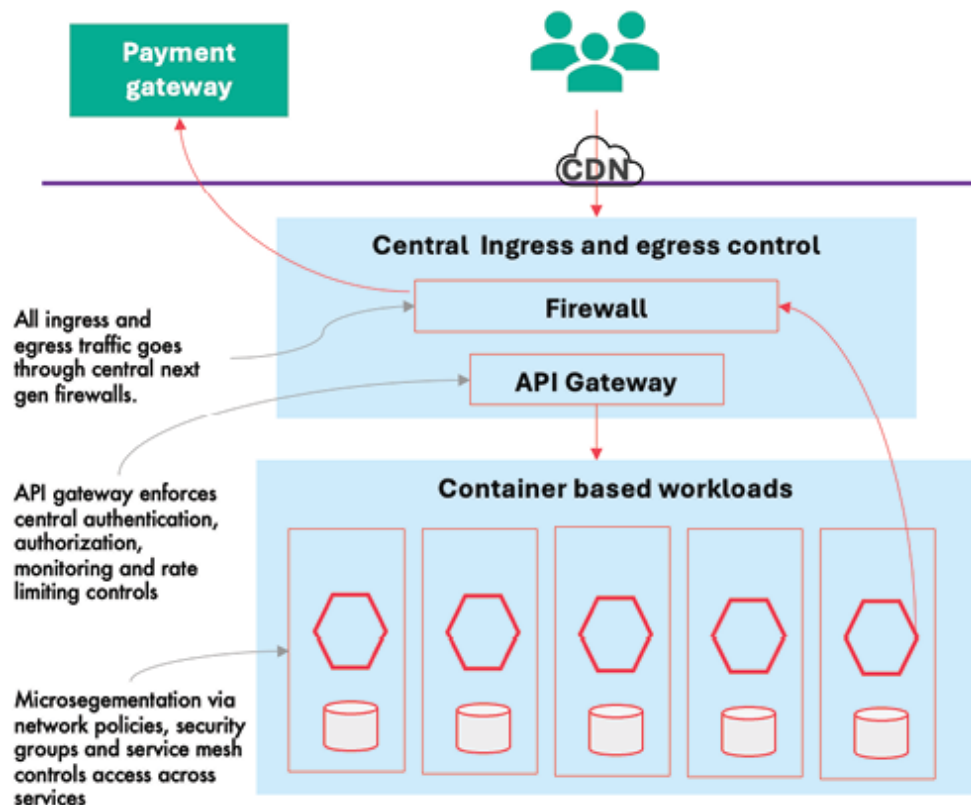


Figure 9.7: Secure your network flow

Table 9.7 covers the various steps and implementation details of this flow:

Step	Implementation details

Central ingress egress	• Organizations typically set up a central hub in cloud native architecture as the gateway for all incoming and outgoing traffic. • A content delivery network such as Cloudflare can be used to serve, filter, protect, and optimize requests before forwarding them to your network. • A next-generation firewall then manages all ingress (inbound) and egress (outbound) traffic. These appliances also offer intrusion detection, prevention, and data loss prevention control. • All requests are then forwarded to an API gateway or web application firewalls (WAF) to inspect and control traffic and enforce strict ingress and egress rules, in addition to protection against OWASP top 10 Vulnerabilities.
Workloads and micro segmentation	• If deploying to Kubernetes (assuming all services are deployed on a single cluster per environment—considering a small footprint), isolate namespaces for each microservice, then define network policies that only allow communication within a namespace and selectively allow communication across services. In addition, a service mesh such as Istio can be deployed to enforce authentication, authorization, and mutual TLS. • If using serverless capabilities such as GCP Cloud Run, leverage native network controls and service mesh capabilities offered by the provider (such as GCP Cloud service mesh).
Communication	• Encrypt all traffic between containers and external services using TLS (HTTPS). Kubernetes Network Policies. • For east-west traffic, use a service mesh like Istio to apply automatic mutual TLS and enforce secure service-to-service communication.
Endpoint protection	Separate from the application architecture, organizations also leverage VPN or Zero Trust Network Access (ZTNA) solutions like Zscaler to ensure secure remote access and enforce device trust using mobile device management (MDM) for endpoint compliance.
Network operations	Monitor and secure container networks with access control lists, VPC firewalls, and IDS/IPS. Use tools like NetFlow or VPC flow logs for traffic analysis, and forward logs to a central SIEM platform for threat detection, correlation, and automated response.

Table 9.7: Secure your network steps and implementation

Secure your data

To secure the data, there are several high-level considerations on securing your data listed as follows:

- Data residency is the geographical decision of where the data will be

stored, and it is decided based on compliance regulations. For instance, GDPR requires that personal data for **European Union (EU)** citizens be stored and processed within the EU. This means you will use your cloud provider's zones in those areas if you operate in the EU.

- Domain-specific requirements then drive data security requirements. As a payment application, we will have to capture confidential customer information, classed as **personally identifiable information (PII)**, and this application will most likely not store any restricted information. [Table 9.8](#) covers the minimum classification level for each domain service, details on what data is needed, and the implications of storing the data:

Domain service	Minimum classification	Details and implications
User management	Confidential	Examples: Users name, contact information, identifiers, preferences, acceptance status, and dates of consents, terms of use, identity documents. Passwords would be managed by external identity provider. Encrypt any sensitive information, such as date of birth or any identity information or documents, using symmetric encryption and secure key management systems (AWS KMS, GCP Cloud Key Management).
Transaction ledger	Confidential	This domain would store the time, amount, currency, and payment information. Payment gateway will be used to manage the actual card information or bank details reducing scope for PCI DSS compliance. Typically, tokenization will be used extensively, masking can be used if required in production-scale data, and necessary testing and development.
Fraud	Confidential	This service should not store any information directly but is expected to receive information about the transaction and decide if it is fraudulent based leveraging machine learning models. Since transaction data is transient, use encryption for interservice communication and decrypt the data only when required.

Domain service	Minimum classification	Details and implications
Payment gateway	Confidential	Just like fraud, this service will pass on data to the external payment gateway and is not expected to store much information, but it is used by the transaction ledger. This service should also not receive or store any card information, but receive a tokenized response from the payment gateway. Use tokenization and encryption to protect data.
Notifications	Confidential	This service will typically send email and SMS notifications and will probably have access to basic PII such as name and contact information.

Table 9.8: Data classification across domain services

- Secret management keeps secrets such as passwords, credentials, and API keys out of container images and code, using tools such as GCP and AWS secret manager or Vault. Secret rotation and proper access controls are also essential across environments.
- Data at rest should be encrypted using **customer-managed encryption keys (CMEK)**. Object versioning and database locks should be set up for compliance or immutability. Lastly, field-level encryption should be applied for sensitive data (date of birth, personal identity documents).
- Data in transit must be encrypted end-to-end from the customer to the datastores, using protocols per NIST guidance, such as TLS 1.2+ for all network traffic.
- Enable or set up versioning of objects and data, automated backups, and point-in-time database recovery.
- To complete the lifecycle, set up lifecycle policies to automatically archive data when it is not in active use and delete the data when not required or for compliance with the right to forget as part of GDPR compliance.
- Lastly, maintain audit trails of all changes, set up and continuously test your backup and recovery procedures.

Vulnerability management

Table 9.9 covers the different areas within vulnerability management and details the implementations in each area:

Area	Implementation
Configuration management	- Secure cloud configuration and harden host operating system and kernel using best practices and CIS Benchmarks. Apply container-specific kernel hardening (AppArmor, SELinux, seccomp) for Kubernetes nodes. - Secure your Kubernetes and container compliance by adopting CIS Benchmarks for Docker and Kubernetes - Build your infrastructure to be immutable.
Vulnerability scanning	- Integrate SCA, including container scans, OSS, SAST, DAST into your CI CD toolchain. - Integrate SBOM generation and attestation using tools like Syft within your pipelines and further integrate with artifact scanning tools. - Image signing and verification using cosign, notary.
Monitoring	- Set up cloud security posture monitoring using services such as GCP Security Command Center, AWS Security Hub and Azure Defender. - Monitoring and auditing of Kubernetes and cloud infrastructure events. - Implement runtime threat detection using services like AWS GuardDuty, Inspector, GCP Security Center and Azure Defender and for Kubernetes integrate Falco and Open Policy Agent (OPA). - Use CNAPP such as Wiz, Aqua.

Table 9.9: Vulnerability management

Continuous monitoring

Many of the areas within continuous monitoring are standard organizational technologies and processes. [Table 9.10](#) covers the specific areas applicable in an application context and the corresponding implementation details using native cloud monitoring, centralized log shippers, audit trails, and SIEM integration to capture every layer, finally feeding data into dashboards and automated alerting:

Area	Implementation
Monitoring infrastructure	- For serverless offerings, use the native cloud monitoring and cloud logging offered by cloud providers to track access patterns. - When running Kubernetes clusters, use log shippers (e.g., Filebeat, Fluent Bit) to forward logs to a centralized location. - Enable audit trails for all user and service actions. This would mean integrating a mature logging framework and logging all critical applications related to payment transactions, fraud, etc. - Enable cloud audit logs for all services provided by the provider.

	• Ensure monitoring covers all layers: containers, hosts, orchestrators, and network. • Adopt standards such as OpenTelemetry for observability to standardize how traces, metrics, and logs are produced and captured.
Security operations	• Feed logs into a SIEM (e.g., Splunk, Sentinel) for normalization, storage, and correlation. • Centralize security event management, automate alerting, and integrate with case management systems. • Use AI driven correlation, anomaly detection and automated remediation offered by various tools in the monitoring toolchain.
Dashboards and alerting	• Create dashboards to visualize real-time threats, system health, and risk posture. Service level objectives and indicators are a good way to track these. Some examples include repeated login failures, payment failures, and fraudulent transactions beyond thresholds. • Set up alerting upon breaches to predefined indicators. • Schedule regular reporting for compliance and executive teams.
Incident response	• Prepare incident response playbooks specific to your application use cases, such as DDoS attack, service availability, anomalous logins, and payments or fraudulent transactions beyond threshold. • Use tools for evidence collection, container quarantine, and notification. • Automate response with SOAR and implement playbooks to automatically respond (e.g., isolate host, disable account), integrate with ticketing systems (e.g., Jira, ServiceNow) for orchestration.
Backup and recovery	Regularly back up container configurations, secrets, and persistent data. Test disaster recovery procedures and ensure backups are stored securely and can be restored quickly. Use automated backup tools and validate through routine drills.

Table 9.10: Continuous monitoring for containers

Conclusion

Containers are a modern software packaging and deployment approach that encapsulates applications and dependencies into lightweight, portable units. This chapter discusses container security through a digital payment processing application use case. We covered a generic technical stack across cloud providers for hosting this application using containers. We then applied the Zero Trust approach and the six pillars we have learned in this book so far. Containers are a scalable and portable solution for modern

applications, necessitating a shift to Zero Trust security models due to their dynamic nature. Adopting Zero Trust for containers enables continuous verification across the software lifecycle and protects your application workloads.

In the next chapter, we will discuss the emergence of **artificial intelligence (AI)** and how we can prepare for it.

Points to remember

- Cloud computing is essential for businesses, and most companies rely on cloud services. Containers offer scalability, portability, and agility, enabling the development of modern applications. Still, the dynamic and distributed nature of containers necessitates a shift from traditional security models to Zero Trust approaches. Containers evolved from virtualization to address scalability needs, with traditional virtualization using virtual machines and containers offering a more resource-efficient approach.
- A container is a standard unit of software that packages code and dependencies for consistent execution across platforms. Container platforms manage containerized applications, including orchestration, monitoring, governance, security, and automation.
- **Security challenges:** Containers face security risks due to shared kernels, multi-tenancy, and dynamic nature. Monitoring and detecting suspicious traffic in containerized environments is challenging due to the complexity of virtualized networks and high application density.
- The rest of the chapter showed how the six pillars of securing your code, identities, network, data, managing vulnerabilities, and continuous monitoring can be applied to any container-based application. It used the example of a digital payment processing application that allows users to log in, initiate payments, perform fraud checks, and send notifications.

References

1. <https://inventivehq.com/history-of-virtualization/>
2. <https://www.bdrsuite.com/blog/virtualization-trends-series-part-1-a-brief-history-of-virtualization/>
3. <https://www.aquasec.com/blog/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016/>

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



CHAPTER 10

Security in the Age of Artificial Intelligence

Introduction

The idea of a machine or system performing human-like intelligence, learning, and reasoning to solve problems and tasks dates back to the 1950s. Turing's paper from 1954 discusses the imitation game as a way to assess the intelligence of a machine. From this early thinking to now, where large language models have been released and are sparking an era of productivity and innovation. Today, on one side, machine learning algorithms can help identify anomalies and respond quickly to attacks. On the other hand, **artificial intelligence (AI)** systems continually evolve their code to evade detection and create more sophisticated social engineering attacks that can deceive even security-conscious users. As AI becomes more mainstream and accessible, threat actors no longer require extensive technical expertise to launch sophisticated attacks, effectively lowering the barrier to entry for cybercrime while simultaneously straining existing security measures and prompting organizations to continually evolve their defensive strategies. At the same time, there is also a need for guiding principles to ensure the responsible development of artificial intelligence, promoting fairness, transparency, and accountability.

Structure

The following topics are covered in the chapter:

- Brief history of AI
- Emerging threats due to AI
- AI as a defense tool
- AI-specific threats and mitigations
- Regulatory and ethical considerations
- Checklist for building secure AI systems

Objectives

As with most topics, we will start with a brief history and background on the emergence of AI and **machine learning (ML)**. We will cover some of the recent advances, specifically in the field of generative AI and large language models. We will then talk about how AI is leading to the emergence of more sophisticated attacks and reducing the entry barrier. Next, we will examine how it enhances threat detection and response, and some of the specific threats and mitigations with AI systems. Finally, we will discuss ethical and accountability issues, as well as the need for a responsible AI framework and its adoption.

Brief history of AI

Artificial intelligence as a concept may have existed for much longer, but its history as we know it began in the 1940s and 1950s. This is when pioneers like *Warren McCulloch* and *Walter Pitts* developed the first mathematical model of a neural network, and *Alan Turing* introduced the concept of machine intelligence with his famous *Imitation Game*. The field formally took shape in 1956, when the term AI was coined at the *Dartmouth Conference*, setting the stage for decades of innovation. The 1970s and 1980s were characterized by high costs, limited scalability, and unmet expectations, resulting in the so-called *AI Winter*, a period marked by reduced funding and waning interest.^[1]

Later in the 1990s, AI experienced a revival as researchers shifted due to advancements in computational power and the emergence of vast new data sources. A landmark moment came in 1997, when IBM's Deep Blue defeated chess champion *Garry Kasparov*. The 2000s and 2010s witnessed breakthroughs in deep learning and neural networks, enabling AI to excel in tasks such as image recognition, language processing, and speech recognition. Virtual assistants like *Alexa*, *Google Assistant*, and *Siri* came out and became household names.

In the 2020s, the most significant revolutionary shift was the public release of ChatGPT by OpenAI in November 2022. The release introduced the world to **large language models (LLMs)** and marked the beginning of the era of **generative AI (GenAI)**. This GenAI ecosystem has expanded significantly with various modules such as GPT, Llama, Gemini, Claude, Mistral, and others trained on vast amounts of data, these generative AI models, that can now create human-like text, convert text to images, generate code, and even create music and videos; these models have seen widespread adoption across industries, finance, healthcare to the creative arts.

AI comprises multiple areas. *Figure 10.1* illustrates these relationships and shows the connections between them. *Table 10.1* describes each of these. AI is the generic term used broadly to represent any intelligent system.

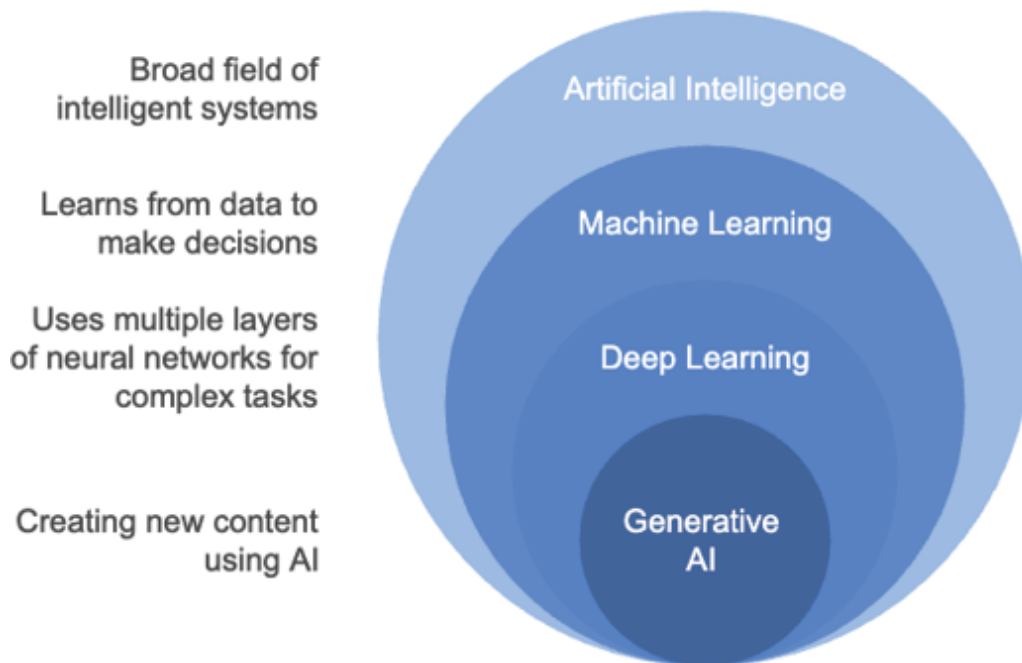


Figure 10.1: AI streams

Area	Definition	Examples
AI	The broad field of computer science focuses on creating systems that can perform tasks requiring human-like intelligence, such as reasoning, learning, and problem-solving.	Robotics, speech recognition, expert systems and everything from the following categories.
Machine learning (ML)	A subset of AI that uses algorithms to analyze data, learn from it, and make predictions or decisions without explicit programming.	Fraud detection, recommendation engines, medical diagnosis using decision trees, and linear regression.
Deep learning (DL)	A subset of ML that uses artificial neural networks inspired by the structure of the human brain to extract features from structured and unstructured data for complex tasks.	Image and facial recognition, natural language processing.
Generative AI	A further subset of ML focused on generating new content—such as text, images, music, or code—that mimics the data it was trained on.	Chatbots, image generation, and synthetic data creation.

Table 10.1: AI streams

Machine learning can also be categorized into three different types:

- **Supervised learning**, which requires labeled datasets typically provided by humans, trains the models to learn relationships that enable them to

predict inputs they have not seen before. Examples include spam detection and image recognition.

- **Unsupervised learning**, which uses datasets that are not labeled, and the model tries to identify clusters and anomalies without any external guidance. Examples include customer segmentation and anomaly detection.
- **Reinforcement learning** is used for training robots, landing spacecraft, and other applications where the agent learns a decision-making strategy or policy through interaction with a real environment and receives rewards or penalties for its actions. Also, **reinforcement learning with human feedback (RLHF)** combined with **retrieval augmented generation (RAG)** architectures dominate LLM fine-tuning. RLHF allows for the alignment of model behavior to safety, helpfulness and tone.

Emerging threats due to AI

Although the concept of AI has been around for many decades, recent advancements have lowered the barrier for entry into this space, enabling threat actors to leverage AI and ML to generate sophisticated attacks. ML algorithms can help attackers identify patterns in network traffic and user behavior, enabling more targeted and stealthy attacks. Natural language processing capabilities allow the creation of more persuasive social engineering attacks that can deceive even security-conscious users. Additionally, AI-powered tools can generate polymorphic malware that continuously changes its code to evade detection. The democratization of AI tools means that threat actors no longer require extensive technical expertise to launch sophisticated attacks, effectively lowering the barrier to entry for cybercrime while simultaneously straining existing security measures and necessitating that organizations continually evolve their defensive strategies. This section will cover the various emerging threats based on the different types of AI we introduced so far.

Deepfakes and synthetic media

In 2024, Arup, a UK-based engineering firm, made a transfer of millions of

dollars following a video call with senior management, which was a deepfake created using artificial intelligence. Deepfakes are realistic fake videos, audio, or images that are being used for fraud, impersonation, and disinformation; they can manipulate public opinion, damage reputations, or enable financial crime.^[2] EU AI Act Article 52 sets out requirements around transparency for providers and users of AI systems, including labeling of deepfakes.

AI-driven phishing and social engineering

One thing that generative AI excels at is crafting highly personalized content. This is used to create convincing phishing emails or messages at scale, bypassing traditional spam filters and tricking users more effectively. Deep impersonation techniques are also used to create fake identities that appear legitimate and pass through verification processes, allowing fraudsters to create accounts, loan applications, etc. Large languages have also enabled attackers to clone voices, fake online personas, and build emotional relationships with victims. There has also been a surge in smishing scams where attackers send text messages claiming delivery issues. An employee who clicked such a link and entered credentials on a fake site led to a breach that resulted in a \$10 million lawsuit. In 2025, cybercriminals also exploited Microsoft Teams for sophisticated social engineering by first sending a suspicious email, then using Teams to impersonate IT staff, leveraging AI to mimic communication styles and gather personal details. Employees are then convinced to grant remote desktop access, enabling the deployment of ransomware and unauthorized lateral movement within company networks by this fake IT support.^[3]

Autonomous AI agents and unchecked access

The increased adoption of artificial intelligence has led to a proliferation of AI agents capable of acting with minimal human oversight, introducing new attack surfaces. If not properly governed, they can access sensitive data, make unauthorized changes, or be manipulated by attackers. A 2025 SailPoint research study found that 80% of organizations reported AI agents acting in unexpected and risky ways, such as accessing unauthorized resources or sharing sensitive data, often due to a lack of oversight and

governance. The same research also highlighted that more than half (53%) of AI agents are accessing sensitive information. *Figure 10.2* shows some of the unintended actions that such AI agents have performed.^[4]

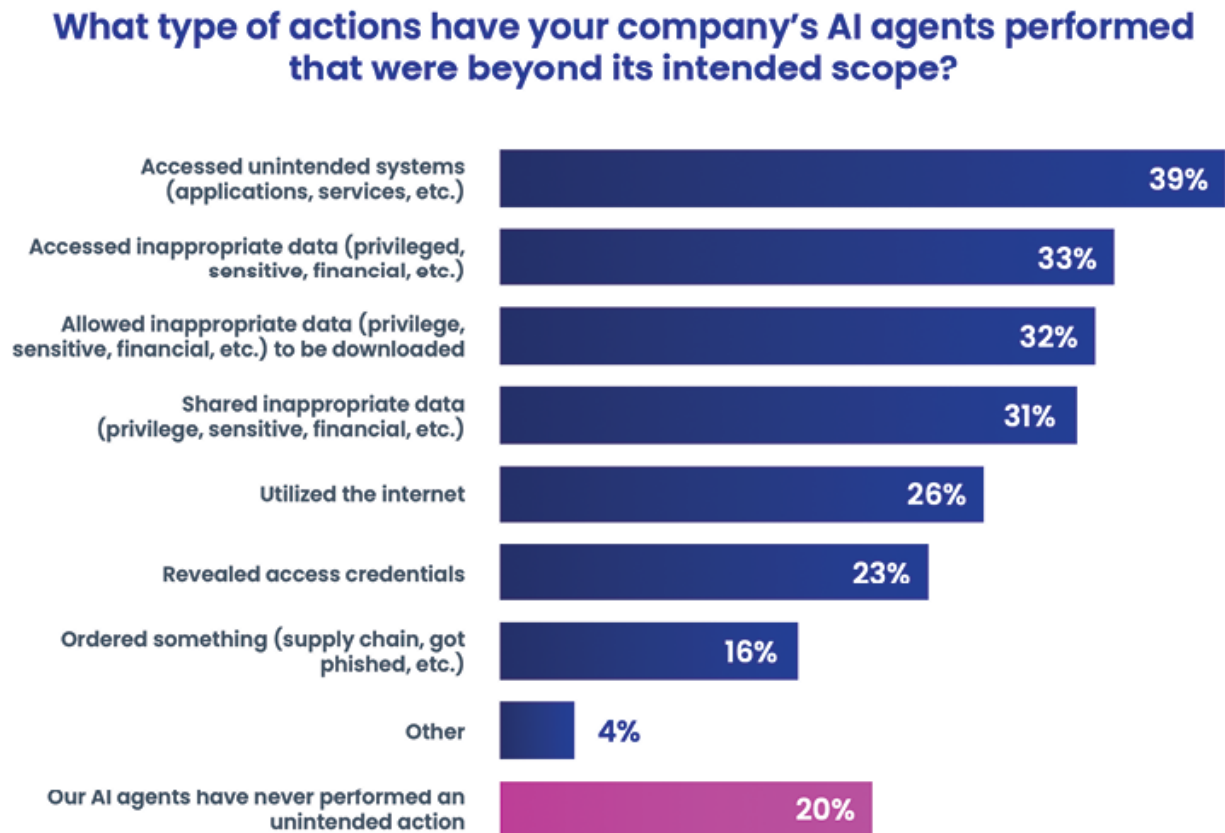


Figure 10.2: Actions performed by company AI agents^[4]

AI agents need to use governance practices such as sandboxing and human-in-the-loop for key interactions to reduce the risk of exposure. Sandboxing is limiting an agent's permissions, isolating their execution environments, and enforcing strict policies, reducing blast radius if something goes wrong. On the other hand, human-in-the-loop places a checkpoint on consequential actions—approvals for data access, financial transactions, system changes, or external communications—to control mistakes or misuse before impact.

AI-enhanced malware and automated attacks

Cybercriminals utilize AI to automate and personalize cyberattacks, including malware that adapts to evade detection and AI-powered password-guessing techniques. AI-supported malware can identify system

vulnerabilities and adjust its behavior, making it harder to detect and stop. Sophisticated malware also uses encryption, encoding, etc., to hide and disguise the true intent of the malicious code, known as content obfuscation.
[5]

Zero-day and one-day attacks

Zero-day and one-day vulnerabilities are either unknown to the vendor or for which a patch has been recently made available. AI accelerates the discovery of such vulnerabilities and exploits them to launch attacks more quickly.^[6]

Identity and access attacks

Previously, we discussed that AI-generated deepfakes and synthetic identities are increasingly used to bypass identity verification systems and perpetrate fraud. There is a surge in deepfake attacks targeting identity verification across sectors.

In addition, MFA fatigue, also known as **push bombing**, was recently exploited by cybercriminals who bombarded employees at a multinational energy company with a barrage of MFA push notifications at unusual hours, ultimately causing them to approve the request. This led to attackers gaining access to critical infrastructure systems, disrupting operations for 72 hours, and causing an estimated \$50 million in operational losses.

Economic and political manipulation

Deepfakes and AI-generated disinformation have been used to sway elections and destabilize governments through disinformation campaigns, as seen in Malaysia and elsewhere. AI can also be weaponized to manipulate financial markets, spread misinformation, or destabilize political systems.

Complexity and cost of protection

The final, most significant risk emerging and affecting most organizations is the increased complexity of systems as AI becomes an integral part of the ecosystem. Although the barrier to using the basics of AI has been reduced, adopting it at an enterprise scale remains expensive to implement and maintain, as it requires deploying and supporting advanced AI security systems and safeguards. This, in turn, necessitates significant investments in

hardware, software, and skilled personnel.

With this understanding of the threats posed by AI and how attackers are exploiting this capability, let us explore how AI is helping organizations defend themselves against these emerging threats.

AI as a defense tool

As the use of AI has enabled faster and more sophisticated attacks, organizations must also adopt AI to enhance their team's defenses more quickly, or they will be caught off guard. AI enables teams to detect threats more quickly, shift from reactive to proactive protection, and automate incident response, allowing them to focus on addressing complex problems.

Figure 10.3 summarizes the various aspects of AI usage as a defense tool; we will dive deeper into this later:

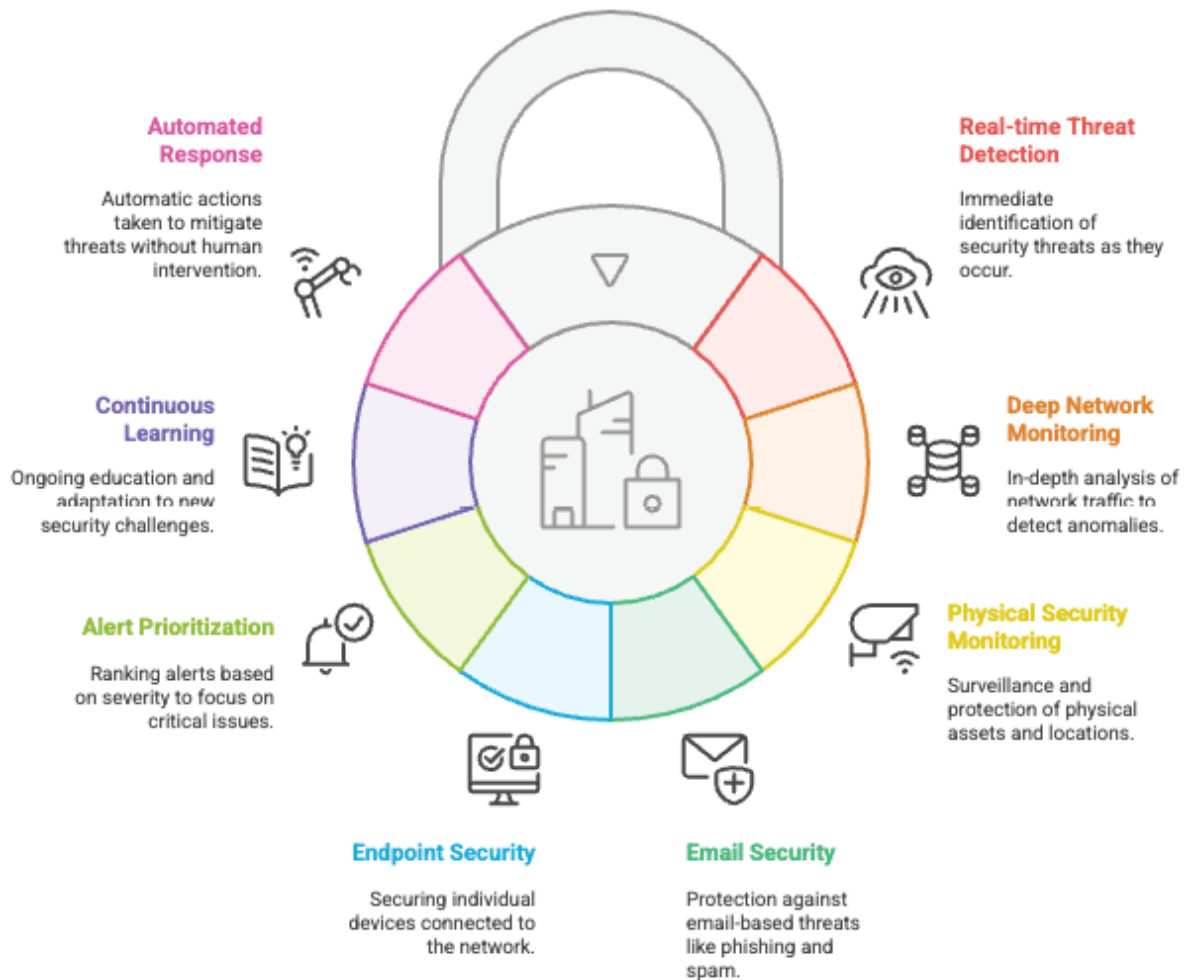


Figure 10.3: AI as a defense tool

Another emerging area is the AI4Sec tools, such as Microsoft Copilot for Security, Google Sec-AI workbench, geared towards protecting organizations using generative AI. These tools attempt to combine existing security capabilities in the provider ecosystem, for instance, Copilot for Security allows the use of natural language across Defender, Entra, and threat intelligence capabilities that Microsoft already offers. The benefit of these tools is the power to combine speed and scale and allowing humans to respond faster.

Real-time threat detection

AI can analyze vast amounts of data (network traffic, system logs, and user behavior) in real time, identifying threats and anomalies much faster and more accurately than human analysts. Correlation of data from multiple

sources, prioritization of alerts, and provision of actionable insights become easier, improving the efficiency of security operations centers. This enhances threat detection, enabling early detection of attacks and suspicious activities. The following are some key areas where AI is helping with threat detection:

- **Threat hunting:** It leverages ML for pattern recognition (e.g., detecting credential stuffing) and predictive analytics to preempt attacks. Machine learning models can also anticipate potential attacks by analyzing historical data and emerging patterns, helping organizations prepare for and prevent breaches.
- **User and Entity Behavior Analytics (UEBA):** It utilizes AI to monitor user activity, detecting insider threats or compromised accounts by flagging deviations from established behaviors.
- **Advanced malware and phishing detection:** AI-powered tools can detect and block new and evolving malware, as well as sophisticated phishing attempts, by analyzing code behavior and email patterns.
- **Threat intelligence integration:** AI aggregates and analyzes data from global sources to identify emerging threats and predict attack patterns, empowering proactive defense.

Deep network monitoring

Network traffic analysis incorporates AI to scrutinize packet flows across networks for signs of malicious activity. By monitoring real-time traffic, it detects abnormalities that may suggest a network breach. Anomaly detection systems identify deviations from established patterns within network traffic or system processes. These systems utilize AI to develop normal activity baselines and identify irregularities, such as unusual data flows or access requests, which may indicate the presence of malware, infiltration, or data exfiltration attempts unnoticed by traditional methods. They provide critical and timely threat recognition and are vital in defending against AI-based attacks. By dynamically updating their understanding of network behaviors, anomaly detection systems effectively neutralize novel threats, offering an adaptable defense layer against AI-generated malware. This proactive approach minimizes risk exposure, safeguarding critical infrastructure.

Physical security monitoring

Physical security refers to the measures adopted to protect people, property, and information from physical threats and unauthorized access to sensitive facilities, offices, and devices. Traditional physical security involved controlling access, utilizing physical barriers, and conducting manual surveillance of facilities. AI has changed this in many ways, mentioned as follows:

- **Facial recognition and access control:** AI-driven facial recognition enhances access control, reducing risks associated with lost or stolen credentials and improving the tracking of individuals in secure areas.
- **Intelligent surveillance:** AI enhances traditional surveillance with real-time video analysis, detecting suspicious activities, unauthorized access, and unusual behavior patterns.
- **Image and video analysis:** AI-driven models analyze surveillance footage for unauthorized access, suspicious behavior, or physical security breaches.
- **Predictive threat assessment:** AI analyzes diverse data (historical patterns, weather, social media) to predict, prepare, and adapt its defenses.
- **Automated response:** Incident response, escalation of high-priority incidents, and dispatch of security personnel can be automated, reducing response times and improving efficiency.
- **Autonomous security robots:** AI-powered robots can patrol and monitor environments, detect intrusions, and act as visible deterrents.

Phishing detection

Email phishing is still the most common target for many attacks. Unlike conventional rule-based approaches, AI utilizes **natural language processing (NLP)** and behavioral analysis to examine content, sender behavior, and contextual cues, thereby detecting and blocking phishing emails, including sophisticated, AI-generated attacks. Advanced AI tools can also analyze URLs, attachments, and embedded images for malicious intent and automatically quarantine them. AI-based detections can also learn and adapt quickly to emerging attack strategies, thereby increasing their effectiveness.

Endpoint security

Endpoint security has become an essential part of an organization's security posture with the emergence of remote work and the **Internet of Things (IoT)**. Every device connected to a network, whether in the office or remote, represents a potential entry point for attackers. The risk of data breaches, malware infections, and insider threats increases dramatically.

AI is revolutionizing endpoint security by enabling real-time detection and response to threats directly on devices such as laptops, smartphones, and servers. Modern solutions, such as **endpoint detection and response (EDR)** and **extended detection and response (XDR)**, utilize AI-driven behavioral analytics to monitor user and system activity, quickly identifying anomalies that may signal malware or other attacks. These systems use machine learning to distinguish between normal and suspicious behaviors, enabling them to predict and prevent attacks before they can cause harm and perform adaptive baselining by continuously learning what is typical behavior. For example, Cylance's AI-powered endpoint security platform uses advanced algorithms to proactively detect and block threats, often before traditional antivirus tools would even recognize them. GenAI is also used to summarize and provide enriched content to assist with root cause identification and recommendation of next steps. By integrating attack signal intelligence and continuously learning from new threat patterns, AI-powered endpoint security provides organizations with a robust, adaptive defense against the rapidly evolving tactics of cybercriminals.

Alert prioritization

With the growth and sprawl of connected devices and infrastructure, there is a heightened need to reduce noise and streamline incident response. AI systems filter out false positives and automatically prioritize high-risk incidents, enabling security teams to focus their attention on the most critical threats and alleviating the problem of alert fatigue. AI also automates the process of gathering and enriching relevant data for each alert, providing valuable context that accelerates investigations and decision-making. This targeted and efficient approach enhances the effectiveness of security operations, ensures that genuine threats are addressed promptly, and minimizes potential damage to the organization.

Automated and adaptive response

Once alerts are prioritized, it is essential to act on them promptly; this is where incident response plays a crucial role. Automated incident response systems integrate AI for rapid threat mitigation and system recovery. These systems autonomously assess, prioritize, and respond to incidents, executing pre-configured actions to contain and remediate threats. By leveraging AI, they enhance response times, crucial in managing AI-driven malware through:

- **Automated response and mitigation:** AI can automate routine security tasks, such as log analysis, vulnerability scanning, and incident response, freeing human experts to focus on complex threats and reducing response times. AI systems can also isolate affected devices, block malicious traffic, or quarantine suspicious emails in real time, minimizing damage and reducing response times.
- **Dynamic incident response:** AI-driven **security operations centers (SOCs)** utilize automation to triage alerts, investigate incidents, and adjust response workflows in response to evolving threats.
- **Autonomous security operations:** **Security orchestration, automation, and response (SOAR)** platforms utilize AI to dynamically prioritize and respond to incidents, enabling faster and more effective containment.
- **Enrichment and faster incident response:** AI rapidly synthesizes data from multiple sensors and sources to guide security teams in real time during incidents.

Continuous learning and awareness

Lastly, improving security awareness is an integral part of an organization's continuous learning culture and improvement of security posture, as many attacks start with human compromise. A successful awareness program should be embedded into regular workflows, providing employees with ongoing opportunities to learn, adapt, and reinforce good security behaviors and leverage AI where possible. The aim is to ensure that security becomes second nature across all levels of the organization. When employees can identify and report threats early, they effectively act as an extension of the security team.

AI can help create realistic simulations and adaptive training techniques. Platforms like *Keepnet* and *KnowBe4* enable organizations to deliver a wide variety of simulated attacks, including AI-driven phishing emails, smishing (SMS-based attacks), vishing (voice phishing using synthetic voices), and pretexting scenarios. These simulations not only mimic real-world threats but also assess individual employee responses, helping identify those who may be more vulnerable. For example, MFA fatigue simulations can test how employees respond to multiple authentication requests, highlighting areas where further training is needed.

Gamification can also be used to encourage further engagement, utilizing tools such as leaderboards and team competitions, to foster a positive security culture. Recognizing and rewarding employees or departments who consistently perform well in simulations reinforces good behavior and helps normalize vigilance.

The effectiveness of these efforts can be measured through metrics such as click-through rates, reporting time, and training completion. These should be reviewed regularly and used to adjust the program in response to evolving threats.

Lastly, these insights can be integrated into broader security operations, such as incident response plans and insider threat programs. Ultimately, continuous learning supported by AI transforms the organization's people into a proactive line of defense, aligning people, processes, and technology to reduce risk.

AI-specific threats and mitigations

AI systems differ fundamentally from conventional software in several key ways: they rely on large datasets, employ complex algorithms, and undergo continuous learning. This leads to some unique vulnerabilities that can go undetected. They can be attacked through data (e.g., data poisoning), their logic (e.g., adversarial), or their outputs (e.g., model inversion and extraction). For instance, consider an AI model that makes decisions related to who should be issued a loan or a mortgage; an attacker can manipulate the data used for training such a model, thereby corrupting its behavior and causing it to make incorrect decisions. Since AI systems frequently process

large volumes of sensitive data, thereby increasing the risk of accidental leaks or intentional data exfiltration, particularly when integrated into widely used platforms. In 2023, *Samsung* employees leaked confidential company information by using ChatGPT to review internal code and documents, prompting the company to ban the use of generative AI tools.^[7] Slack’s AI was shown to leak data from private channels via prompt injection attacks in 2024.^[8]

As the use of AI increases in critical domains such as healthcare, financial services, security, and infrastructure, successful attacks can result in privacy breaches, economic loss, reputational damage, regulatory non-compliance, or even physical harm. It is thus vital to understand these specific threats and how they can be mitigated. *Table 10.2* presents some common threats to AI models, their meanings, and potential mitigations:

Threat	Definition	Mitigation
Data poisoning	Attackers manipulate training data to corrupt AI models, causing them to make incorrect or malicious decisions.	Secure training data, regular audits, and strict access controls
Prompt injection	Specific to generative AI, adversaries exploit vulnerabilities by injecting malicious prompts or hijacking model outputs, leading to data leakage or unintended actions.	Input validation, regular audits
Model hijacking	Unauthorized control of the model is gained to alter its behavior or perform unintended tasks.	Secure model training, access controls, monitoring, and validation.
Model inversion	Attackers reconstruct or infer information and models decision-making by analyzing their outputs, compromising privacy and intellectual property.	Encrypt or anonymize data, restrict access, retrain, monitor model outputs, and perform regular audits.
Data leakage	Information that would not be available during prediction time is used during training, leading to unreliable models and inaccurate predictions.	Understanding data, careful feature engineering, regular audits, and access controls
Adversarial attacks	Deliberate attempts to manipulate inputs (called adversarial inputs) to deceive AI-based detection systems, allowing threats to bypass security controls undetected.	Adversarial training by exposure to adversarial examples, preprocessing inputs, access controls, and human oversight.

Threat	Definition	Mitigation
Data exfiltration via embeddings	Sensitive data is extracted through embedding vectors, vector search leaks or model outputs encoding proprietary data.	Isolate embedding flows, encrypting at rest and transit, protect vector data access through access controls, segment models and application contexts
Prompt-leaks and cross-context attacks	Prompts are used by attackers to share context to leak data across isolated contexts, such as across agents or tools.	Strict boundaries, monitoring of agent interactions, and sanitization of prompts and context data

Table 10.2: AI model threats and mitigations overview

The *OWASP Machine Learning Security top 10* initiative aims to deliver an overview of the top 10 security issues in machine learning systems. This is currently in draft release, but it still serves as a good starting point for all attack considerations, much like the OWASP top 10 for web applications.^[9] Adoption of such checklists and recommendations for AI Systems also helps provide a foundation for meeting risk frameworks such as NIST AI RMF (specifically, managing and governing functions of the framework). The top 10 security risks as of this writing are listed in *Table 10.3*:

Security risk	Details
Input manipulation attack	Umbrella term for adversarial attacks where inputs are altered to mislead the model. Preventions: Adversarial training, input validation, and robust models.
Data poisoning attack	Training data is altered to cause the model to behave in an undesirable way. Preventions: Data validation and verification, secure data storage, data separation (training from production), access controls, monitoring, auditing, anomaly detection on training data.
Model inversion attack	Attacker reverse-engineers the model to extract information from it. Preventions: Access control, input validation, model transparency, monitoring, and regular model retraining.
Membership inference attack	Attackers manipulate the model's training data to cause it to behave in a way that exposes sensitive information. Preventions: Model training on randomized or shuffled data, model obfuscation, regularization, reducing training data, removing redundancy, testing and monitoring.
Model theft	Attackers gain access to the model's parameters. Preventions: Encryption, access controls, backups, model obfuscation, watermarking, legal protections, monitoring, and auditing.

Security risk	Details
AI supply chain attacks	A broad category that includes how actors can target the supply chain of ML models, it spans data management platforms, model management software, model hubs, and other types of software used by ML engineers. Preventions: Verify package integrity, use of up-to-date packages, install packages from secure sources, ML infrastructure security using all principles of Zero Trust.
Transfer learning attack	Attackers train a model on one task and then fine-tune it on another task to cause it to behave in an undesirable way. Preventions: Regular monitoring and updates to the training dataset, secure and trusted dataset usage, model isolation, differential privacy, and regular audits.
Model skewing	Manipulation of the distribution of the training data to cause the model to behave in an undesirable way. Preventions: Access control, authenticity of feedback data, data validation and cleaning, anomaly detection, monitoring, and regular training.
Output integrity	An attacker aims to modify or manipulate the output of a machine learning model to change its behavior or cause harm to the system it is used in. Preventions: Cryptography to verify authenticity of results, secure communication, input validation, tamper-evident logs, regular updates, monitoring and audit.
Model poisoning	Manipulation of the model's parameters to cause it to behave in an undesirable way. Preventions: Regularization, robust model design, cryptography to secure parameters and weights

Table 10.3: OWASP Machine Learning Security Top 10

Since the release of ChatGPT, numerous LLMs have been introduced, and they are now being integrated into various aspects of customer interactions and internal operations. With this broad adoption, developers and security professionals are discovering new vulnerabilities and ways to counter them. The OWASP top 10 for large language model applications was introduced in 2023 and has recently been updated with a new Top 10 in 2025^[10]. [Table 10.4](#) summarizes the key vulnerabilities and their details:

Vulnerability	Definition
Prompt injection	User prompts alter LLM's behavior or output in unintended ways. Mitigations: Constrain model behavior, validate output formats, input and output filtering, access controls, approvals for high-risk actions, and external content

Vulnerability	Definition
Sensitive information disclosure	LLM models embedded in applications expose proprietary algorithms, information, or sensitive data such as personal identifiable information (PII), financial details, and health records. Mitigations: Data sanitization, input validation, access controls, federated learning and privacy, user education and transparency, secure system configuration, homomorphic encryption and tokenization
Supply chain	Supply chain includes anything that affects the integrity of training data, models, and deployment. The rise of open-source LLMs and the use of public platforms such as <i>Hugging Face</i> introduced supply chain risks. Mitigations: Careful selection of data sources and suppliers, up-to-date inventories and updates, integrity checks of models and libraries, auditing and monitoring, anomaly detection on data, and encrypting models at the AI edge.
Data and model poisoning	Vulnerabilities, backdoors, or biases are introduced when pre-training, fine-tuning, or embedding data, leading to the manipulation of security, performance, or ethical behavior. Mitigations: Track data origin and transformation, output validation, vendor validation, sandboxing to limit model exposure, data version control, red team testing, monitoring, integrate RAG, and grounding to reduce hallucinations (typically caused by probabilistic generation and weak grounding).
Improper output handling	Insufficient validation, sanitization, and handling of the outputs generated by large language models before they are passed downstream to other components and systems, leading to privilege escalation, remote code execution, etc. Mitigations: Adopt a Zero Trust approach on outputs, encode model output, context-aware output, parameterized queries, strict content security policies (SCP), robust logging, and monitoring.
Excessive agency	Enables damaging actions to be performed in response to unexpected, ambiguous and manipulated outputs from an LLM, typically in agentic systems caused by excessive functionality, permissions or autonomy to the system. Mitigations: Minimize extensions, functionality, avoid open-ended extensions, execute actions in the user's context, introduce human approvals, and sanitize LLM inputs and outputs.

Vulnerability	Definition
System prompt leakage	Instructions and system prompts used to steer the behavior of the model can contain sensitive information, allowing attackers to determine guardrails and restrictions and potentially compromise systems. Mitigations: Separate sensitive data from system prompts, use an independent system to inspect the output, and avoid system prompts for behavior control.
Vector and embedding weaknesses	RAG used to enhance performance and context uses vector embeddings. Weaknesses in how vectors and embeddings are generated, stored, and retrieved can be exploited to inject harmful content, manipulate outputs, and access sensitive information. Mitigations: Access control, data validation, verified sources of data, data review and classification, monitoring, and logging.
Misinformation	LLMs produce false or misleading information that appears credible. For instance, hallucinations can lead to answers that sound correct; other examples include biases in training data and incomplete information. Mitigations: Use of RAG, fine-tuning, human oversight, automatic validations of output, secure coding, user interface design that integrates content filters, labeling of AI-generated content informing users of accuracy, training, and education.
Unbounded consumption	LLMs generate outputs that allow excessive and uncontrolled inferences, leading to DoS, economic loss, model heft, and service degradations. Mitigations: Input validation, rate limiting, resource allocation monitoring and management, timeouts and throttling, sandboxing, logging, monitoring and anomaly detection, watermarking, graceful degradation, and access controls.

Table 10.4: OWASP GenAI and LLM top 10^[10]

As an example, let us look at the data and model poisoning, described in [Figure 10.4](#). Attackers who have observed and identified targets, understanding their data collection and model training process, inject malicious data that is biased or incorrect. The poisoned data is used to train or fine-tune the machine learning model, in turn changing its behavior. Such models typically perform well on clean test or evaluation data, making it difficult to detect (for targeted backdoor attacks). Once deployed, the model can lead to misclassification, reduced accuracy, and activation of backdoors. In addition, poison data from data sources can also be used to avoid detection.

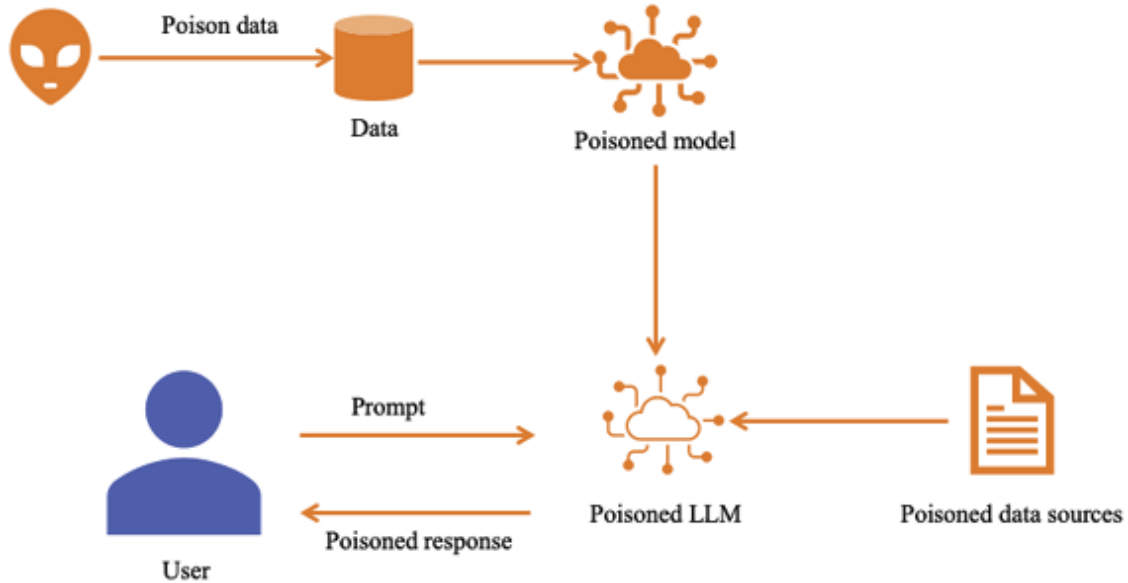


Figure 10.4: Data and model poisoning attack

The initiatives and frameworks we have covered so far will guide you in building a more secure model creation and deployment workflow. As a user of these tools, specifically GenAI and LLM-based offerings, it is crucial to understand the security implications of using these tools and technologies as AI becomes increasingly embedded within organizations. Here are some key steps you can follow to become more aware, ensuring you do not compromise your security as well as your organization's security when using generative AI and in prompt engineering:

- **Understand data classification:**
 - Familiarize yourself with the classification of the data you are working with. Treat and handle it according to its sensitivity level.
 - Understand data handling policies of the AI provider, including how your data is processed, stored, and shared.
- **Vigilance and awareness:**
 - Always be vigilant about the information you input or copy and paste into GenAI tools. Unintentionally, you may be disclosing details you should not.
 - Critically evaluate AI-generated content before acting on it, as GenAI tools can produce convincing but inaccurate or misleading information.

- **Be cautious of phishing and social engineering scams:**
 - Be alert to AI-generated phishing attempts as these tools can craft highly realistic emails, messages, or even voice calls that mimic trusted contacts.
 - Do not click on links or download attachments from unsolicited or unexpected AI-generated communications.
- **Secure access:**
 - Ensure that access to GenAI tools is secure.
 - Use strong authentication methods (and MFA) and restrict access to authorized personnel only.
 - Limit tool access to only what is necessary; avoid granting broad permissions to AI applications, especially those that integrate with email, file storage, or messaging platforms.
- **Misinformation and bias:**
 - AI can hallucinate, and outputs can sometimes include misinformation or biases.
 - Cross-check meaningful outputs with trusted sources, especially for business-critical or regulatory-sensitive tasks.
- **Monitor for data leakage and prompt injection:**
 - Be aware of prompt injection risks as maliciously crafted prompts can trick AI models into revealing sensitive information or performing unintended actions.
 - Regularly review and audit AI tool activity for signs of unauthorized access or data exfiltration.
 - Implement **data loss prevention (DLP)** strategies to monitor, detect, and prevent data breaches or unauthorized data transfers.
- **Adopt a Zero Trust mindset:**
 - Never assume AI tools are inherently secure. Always verify their outputs, monitor their integrations, and restrict their access according to the principle of least privilege.
 - Regularly review AI tool permissions, integrations, and outputs.

Regulatory and ethical considerations

In 2019, the US financial regulator began investigating allegations that *Apple's* credit card offered different credit limits for men and women.^[11] This was behind some complaints that the algorithm used set limits biased against women. Such discrimination is rightfully prohibited by *New York* law and is also prohibited by laws across various jurisdictions. A similar investigation against *UnitedHealth Group*, where the algorithm favored white patients over black patients, was also underway. Such biases can creep up due to various reasons, say, from a biased data set in which women pose a greater financial risk than men.^[12]

Around the same time, Huawei, a Chinese technology company, announced that an AI algorithm was able to complete an unfinished composition by *Franz Schubert* from 182 years ago. This also raised some interesting questions on copyrights and whether the algorithm could be recognized as an author with the same rights as an artist. This sparked the need for new frameworks to prevent the exploitation of human beings' creative work.^[13]

These incidents highlight how bias in AI can deepen disparities, lead to reputational damage, and even create liabilities to companies. The following are some of the main concerns with AI systems and considerations for the need for ethics and responsible usage of AI:

- **Data:** AI systems trained on biased data can perpetuate or amplify discrimination, particularly in areas such as law enforcement, hiring, or lending.
- **Fairness and non-discrimination:** As we saw in the preceding examples, AI systems should avoid bias and treat individuals equitably.
- **Transparency:** AI systems can act as a black box, making decisions without providing a clear explanation of how each decision was determined. Although there have been improvements in the explainability of AI systems' decisions, there is still more work to be done.
- **Accountability:** AI-driven security decisions can be opaque, raising questions about accountability when errors occur (e.g., false positives in behavior detection or wrongful denial of access).

- **Privacy and data protection:** The use of AI in surveillance and behavior analytics can lead to significant privacy issues, as large volumes of sensitive data are processed and stored.
- **Safety and security:** Systems require more robust monitoring of failures and cyber threats, as well as enhanced incident response capabilities.
- **Sustainability:** AI systems can be resource-intensive, potentially impacting an organization's ability to achieve its sustainability targets.

There are multiple frameworks and standards now available to organizations for AI risk management and building ethical and responsible AI systems.

Table 10.5 summarizes some of the key frameworks and considerations:

Standard/Framework	Details
NIST AI Risk Management Framework (RMF) ^[13]	Voluntary framework to manage risks to individuals, organizations, and society from AI. Focuses on trustworthiness, governance, and responsible AI.
EU AI Act ^[14]	Comprehensive, risk-based regulation for AI in the EU. Categorizes AI systems by risk level and imposes strict requirements for high-risk systems.
ISO/IEC AI Standards ^[15]	International standards for quality, reliability, transparency, and accountability in AI systems.
Google Secure AI Framework	Enhances AI system security and resilience against cyber threats.
Microsoft responsible AI Framework	Principle-driven approach designed to ensure the ethical development, deployment, and governance of AI technologies. It is rooted in six core principles: fairness, reliability and safety, privacy and security, inclusiveness, transparency, and accountability
OECD AI principles	Global principles: human rights, transparency, accountability, sustainability, and risk management.
US Executive Order on AI	Sets national standards for safe, secure, and trustworthy AI development and civil rights protection.
Industry-specific regulations	GDPR, CCPA, and California AI regulations ensure AI systems adhere to data minimization and user consent rules, address data privacy, bias, discrimination and transparency in specific sectors
Ethical AI principles	Fairness, transparency, accountability, privacy, safety, human-centered design.

MITRE sensible regulatory framework	Focuses on cybersecurity and vulnerability management in AI, defining responsibilities and usage.
UNESCO Recommendation on Ethics of AI	Aims to ensure that the development and deployment of AI technologies are grounded in human rights, human dignity, and the common good, providing a comprehensive ethical framework for governments, organizations, and individuals involved with AI

Table 10.5: AI risk management frameworks and considerations

Choosing the AI risk management framework and its implementation depends on your organization's industry, regulatory environment, and risk profile. Most organizations will benefit from adopting a combination of international standards (e.g., NIST AI RMF, ISO/IEC) and complying with local regulations (e.g., EU AI Act, GDPR), while also embedding ethical principles into their AI governance structures. This ensures the responsible, trustworthy, and legally compliant deployment of AI.

Checklist for building secure AI systems

Building secure AI systems requires a comprehensive, multi-layered approach that integrates technical, organizational, and operational safeguards throughout the AI lifecycle. Following is a practical checklist, aligned with leading industry frameworks and current best practices:

- **Never trust, always verify:**
 - **Strict authentication:** Enforce strong authentication for all AI models, datasets, APIs, and users, including MFA and **role-based access controls (RBAC)**.
 - **Zero Trust principles:** Assume no implicit trust between components. Verify every access attempt, regardless of origin.
- **Integrate AI into identity and access management (IAM):**
 - **MFA and RBAC:** Require MFA and granular, role-based permissions for all AI system access.
 - **Audit trails:** Maintain comprehensive logs of all access and changes for auditing and compliance.
- **Tools and frameworks for AI security:**

- **Adopt security frameworks:** Utilize established AI security frameworks (e.g., Google's Secure AI Framework, OWASP AI Security Guide) to guide implementation.
- **Automated tools:** Deploy automated tools for vulnerability scanning, monitoring, and policy enforcement across the AI stack.
- **Secure AI development lifecycle (secure SDLC):**
 - **Threat modeling:** Incorporate AI-specific threat modeling at every stage of development.
 - **Code reviews and testing:** Conduct regular code reviews, vulnerability scanning, and penetration testing for AI pipelines and supporting infrastructure.
 - **Supply chain security:** Assess and secure all third-party components, libraries, and pre-trained models used in development.
- **Data-centric protections:**
 - **End-to-end encryption:** Encrypt AI training data, model artifacts, and outputs both at rest and in transit.
 - **DLP:** Deploy DLP solutions to monitor and block unauthorized data transfers from AI systems.
 - **Anonymization:** Apply data anonymization and masking to reduce the risk of sensitive data exposure further.
- **Micro-segmentation and isolation:**
 - **Workload segmentation:** Isolate AI workloads and data pipelines using network segmentation and containerization to limit lateral movement in the event of a breach.
 - **Access controls:** Restrict communication between AI components to only what is necessary for operation.
- **Adopt explainable AI (XAI):**
 - **Transparency:** Ensure AI decisions are explainable and interpretable to facilitate auditing, regulatory compliance, and user trust.
 - **Documentation:** Provide clear documentation of model logic, features, and decision criteria.

- **Assess AI dependencies and data flows:**
 - **Dependency mapping:** Document and regularly update all data flows between AI models, APIs, users, and external systems.
 - **Third-party risk assessment:** Evaluate security practices of all external partners and service providers involved in the AI ecosystem.
- **Model auditing and verification:**
 - **Model audits:** Perform independent audits to assess model robustness, fairness, and compliance with security standards.
 - **Version control:** Maintain detailed records of model versions, training data, and changes for traceability and rollback if needed.
 - **Adversarial testing:** Test models against adversarial inputs and data poisoning attempts to ensure resilience.
- **Continuous monitoring and anomaly detection:**
 - **Real-time monitoring:** Utilize AI/ML-driven monitoring to detect anomalies, model drift, adversarial attacks, and unauthorized access in real-time.
 - **Centralized logging:** Aggregate logs from all AI components for rapid incident detection and response.
- **Privacy-preserving machine learning (PPML) and federated learning:**
 - **Data minimization:** Limit data collection and retention to what is strictly necessary for model performance.
 - **PPML techniques:** Implement privacy-preserving techniques, such as differential privacy, secure multi-party computation, and federated learning, to protect sensitive data during both training and inference.
- **Train teams and address skill gaps:**
 - **Security training:** Provide ongoing training for developers, data scientists, and operators on AI security, privacy, and Zero Trust principles.
 - **Incident response drills:** Regularly conduct drills and tabletop exercises focused on AI-specific incidents.

Conclusion

As AI becomes embedded in organizations and people frequently use GenAI tools, it is vital to understand the security implications of using these tools and technologies. This chapter will help you become more aware, so you do not compromise your security, as well as that of your organization. We covered why security matters in AI, the threats, and vulnerabilities in the overall security model. We also covered how AI can be used as a defense tool, including some of the threats and mitigations, as well as regulatory and ethical considerations. When building secure systems, you will likely face the challenge of balancing complexity with security and other functional and non-functional requirements. In the next chapter, we will explore how to apply a risk-based approach to strike the optimal balance.

Points to remember

- **AI's duality in cybersecurity:** AI's accessibility is lowering the barrier to entry for cybercrime, allowing threat actors with less technical expertise to launch sophisticated attacks. But AI can be used for both offensive and defensive purposes in cybersecurity.
- AI is a broad field that focuses on creating systems capable of performing tasks that require human-like intelligence, such as reasoning, learning, and problem-solving.
 - ML is a subset of AI that uses algorithms to learn from data and make predictions, while DL is a subset of ML that uses artificial neural networks for complex tasks.
 - **Supervised learning:** Uses labeled datasets to train models for prediction tasks like spam detection and image recognition.
 - **Unsupervised learning:** Utilizes unlabeled datasets to identify patterns and anomalies, making it useful for customer segmentation and anomaly detection.
 - **Reinforcement learning:** Trains agents to make decisions through interaction with an environment, receiving rewards or penalties, and is commonly used in robotics and spacecraft applications.

- Cybercriminals are using AI to create personalized phishing emails, impersonate IT staff, and automate malware attacks.
 - AI-driven disinformation, including deepfakes, is also being used to manipulate elections and destabilize governments.
 - AI accelerates the discovery and exploitation of vulnerabilities, enabling faster and more effective attacks.
 - Implementing AI at scale is also complex and expensive, requiring significant investments in security systems and personnel.
 - AI agents can access sensitive data, make unauthorised changes, or be manipulated by attackers if not properly governed.
 - Lastly, cybercriminals are using AI to automate and personalise cyberattacks, making malware more challenging to detect and stop.
- AI also enhances security operations by analyzing data in real-time, detecting threats, and automating incident response. It improves efficiency, prioritizes alerts, and provides actionable insights, enabling proactive threat mitigation and faster incident response. AI can monitor network traffic for anomalies, analyze data to predict and prepare defenses/AI-driven SOCs and SOAR platforms, and dynamically adjust response workflows based on evolving threats. AI also enables faster incident response by automating routine tasks, synthesizing data from multiple sources, and providing real-time guidance to security teams.
- Above all, though, security awareness is essential for continuous learning and improving security posture, as many attacks begin with human compromise. AI-powered training simulation platforms utilize AI to create realistic simulations of phishing, smishing, vishing, and pretexting attacks, assessing individual responses and identifying vulnerabilities.
- AI systems are also vulnerable to attacks through data, logic, or outputs due to their reliance on large datasets, complex algorithms, and continuous learning.
 - Successful attacks on AI systems in critical domains can lead to privacy breaches, economic loss, reputational damage, regulatory non-compliance, or even physical harm.

- Mitigation strategies include addressing data poisoning, adversarial attacks, model inversion, and extraction, as well as implementing robust data security measures.
- Prompt injection and sensitive information disclosure are key vulnerabilities in large language models.
- **AI bias and discrimination:** AI systems trained on biased data can perpetuate discrimination, leading to legal and ethical concerns. Ethical considerations in AI address concerns about bias, fairness, and potential harm caused by AI systems, including the exploitation of human creativity and copyright infringement. Organizations need to adopt frameworks and standards to manage AI risks, ensuring ethical and responsible development and deployment.

References

1. <https://courses.cs.washington.edu/courses/csep590/06au/projects/history-ai.pdf>
2. <https://www.weforum.org/stories/2025/02/deepfake-ai-cybercrime-arup/>
3. <https://www.techtarget.com/searchsecurity/tip/Generative-AI-is-making-phishing-attacks-more-dangerous>
4. <https://keepnetlabs.com/blog/6-shocking-advanced-phishing-attacks>
5. <https://www.sailpoint.com/identity-library/ai-agents-attack-surface>
6. <https://perception-point.io/guides/ai-security/ai-malware-types-real-life-examples-defensive-measures>
7. <https://www.businesstoday.in/technology/news/story/samsung-employees-accidentally-leaked-company-secrets-via-chatgpt-heres-what-happened-376375-2023-04-06>
8. <https://www.itpro.com/security/hackers-could-dupe-slacks-ai-features-to-expose-private-channel-messages>
9. <https://owasp.org/www-project-machine-learning-security-top-10/>
10. <https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>

11. <https://www.bbc.co.uk/news/business-50365609>
12. <https://www.unesco.org/en/artificial-intelligence/recommendation-ethics/cases>
13. <https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf>
14. <https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng>
15. <https://www.iso.org/standard/42001>

CHAPTER 11

Dealing with Complexity

Introduction

Throughout the book, we have covered how you can apply security across various aspects of an organization. However, as you begin to apply security to your products and solutions, you will likely end up using many different products and services, which must be integrated into your ecosystem, leading to increased complexity. *Gartner* predicted back in 2019^[1] that 99% of cloud security failures would be the customer's fault by 2025. Although more recent research highlights the importance of SaaS and generative AI, misconfigurations and identity and access controls remain a significant area of concern. This implies security breaches are not just caused by hackers outsmarting technology, but by simple mistakes in configuring complex systems. This highlights the need for approaches that strike a balance between robust security and simplicity and usability.

Imagine your operations teams working late into the night, surrounded by dashboards, alerts, and a dizzying array of security tools. The irony? The more tools they add to protect their digital assets, the more complex and potentially vulnerable their environment becomes. Studies show that the average organization now deploys an average of 76 security tools, and this proliferation has led to a 20% increase in operational complexity over three years (up from 64)^[2]. Each tool presents its management challenges, update

cycles, and learning curves, resulting in a patchwork of defenses where gaps and blind spots can easily emerge.

The *ENISA 2024 Threat Landscape*^[3] states that organizations face manifold challenges due to their complex estate, and the volume of vulnerabilities across their estate, combined with the processes and difficulty of timely patching, allows attackers to exfiltrate data more easily. This complexity is a real risk; complexity can be the silent killer of cybersecurity, making it harder to detect, prevent, and respond to threats in time.

Structure

The following topics are covered in the chapter:

- Security vs. other requirements
- Striking the right balance

Objectives

This chapter will discuss some of the sources of complexity. You will learn how to strike the right balance between security and other non-functional requirements and discover techniques to make sound judgment calls when security and usability seem to clash. It will guide you through using a risk-based approach to navigate the inevitable complexity that comes with securing modern products and solutions.

Security vs. other requirements

Although security is a key non-functional requirement, a sound system will also encompass various other functional and non-functional requirements, such as availability, performance, and usability. While security aims to protect data, control access, and ensure compliance, other **non-functional requirements (NFRs)** focus on qualities such as speed, scalability, ease of use, and reliability. These requirements define how well a system performs its functions and shape the user experience, system stability, and operational

efficiency.

Most organizations follow a structured process to gather, analyze, and evaluate these requirements throughout the development lifecycle, ensuring they are addressed before a product is released to users. However, if these requirements are not adequately implemented or prioritized, significant issues may not be identified until after launch. Such post-release problems may include poor user experience, system instability, security vulnerabilities, or difficulties scaling the solution, ultimately impacting customer satisfaction, operational efficiency, and the overall success of the product. *Table 11.1* summarizes the key non-functional requirements and some of the security implications that could arise from these:

Type	Description	Security implications
Availability	The degree to which a system is operational and accessible (available) to the end users when required.	High-availability designs (e.g., redundant pathways) typically expand attack surfaces (due to additional infrastructure) and complicate access monitoring.
Scalability	The capability to handle increasing numbers of users, devices, or workloads without degradation.	Rapid scaling can outpace security policy enforcement and lead to extended attack surface, inconsistent access controls or unsecured resources.
Performance	The system's ability to respond quickly and efficiently under expected workloads.	Strict latency requirements may force trade-offs in security checks (e.g., reduced encryption depth or skipped authentication steps), creating vulnerabilities.
Compatibility/ interoperability	The ability to integrate with existing systems, platforms, or protocols.	Legacy system integration often requires supporting dated and insecure protocols (e.g., outdated authentication), weakening Zero Trust principles.
Usability	The ease with which users can interact with the system to achieve their goals.	Complex security controls (e.g., multi-factor authentication, micro-segmentation, endpoint protection) may create a hindrance to users by slowing down access processes.
Maintainability	The ease with which the system can be updated, fixed, or improved.	Frequent updates to security policies risk misconfigurations or delayed patches if maintenance is cumbersome.
Migration/integration	The ease of transitioning from old systems or	Transitioning from legacy systems may require temporary security relaxations,

	integrating with new ones.	exposing hybrid environments to attacks.
Reliability	The system's ability to function correctly and consistently over time.	Fail-safe mechanisms (e.g., automatic failover) can bypass security gateways during failures, allowing unauthorized access.
Support	The availability of vendor or technical assistance for the system.	Inadequate vendor support delays vulnerability remediation, extending exposure windows.

Table 11.1: Non-functional requirements

Several organizations have suffered attacks because of limitations in their security environments, often due to overlooked risks or trade-offs made to meet other non-functional requirements. The following are some examples illustrating how compromises in areas such as usability, performance, or availability can lead to security vulnerabilities:

- Target, where attackers gained access via HVAC vendor credentials, was primarily caused by a lack of network segmentation within the vendor's access to internal systems. Insufficient segmentation can be due to various reasons, such as performance, compatibility, or legacy systems.
- Equifax, where attackers exploited an unpatched Apache Struts vulnerability. Despite being aware of the vulnerabilities, organizations often delay patching and updates due to concerns about availability and uptime, or a lack of automation in their release processes.
- Uber was targeted by an attacker who sent multiple **multi-factor authentication (MFA)** requests to an employee until they were approved. Such scenarios can typically be averted through rate-limiting or context-based MFA challenges. This illustrates how attackers exploit weak links and highlights the importance of usability considerations.

Striking the right balance

Product design and software development involve numerous trade-offs. Within the context of an organization, there are always resource, cost, timeline, and environmental constraints. This requires a level of pragmatism when dealing with various types of requirements, including those related to

security. In current times, it is not acceptable to take years to deliver a full-fledged product with all bells and whistles; it might be too late by then, the opportunity may not exist, a competing product may be out in the market with sufficient market share, or expectations may have changed due to changing regulations. Product development thus requires taking a mindset of building a viable product based on known needs and anticipated priority risks. If not, the product becomes an academic exercise with no end in sight. But how does one do this? How do you evaluate your requirements and decide what is an absolute necessity, especially when it comes to security?

Human-centric approach

In its publication, the **National Institute of Standards and Technology (NIST)**^[4] recommends adopting a human-centric approach to security, which is a crucial aspect of establishing and maintaining a robust security culture. The paper highlights six human element pitfalls listed as follows in cybersecurity:

- **Assuming users are naive:** In reality, people make mistakes and suffer from security fatigue, as they are not cybersecurity experts, suffer from an optimism bias, or think security is someone else's responsibility.
- **Not tailoring communications to the audience:** Experts rely on technical jargon when it is challenging to translate highly technical information into a language that can engage people, most likely due to a failure to understand the intended audience.
- **Unintentionally creating insider threats due to poor usability:** When users are pushed to their limits by time pressures and distractions, unusable security can increase their burden, leading to the adoption of practices that compromise security, such as writing passwords on sticky notes due to overly complex policies.
- **Having too much security:** Security experts can lean towards using the most secure solution or configurations, taking a one-size-fits-all approach. Overly stringent measures can mean users consider them as impediments.
- **Depending on punitive measures or negative messaging to get users**

to comply: Disabling user accounts for not completing security training, posting offender names, or firing someone for falling for phishing emails, fails to consider the root causes and motivations behind the actions.

- **Not considering user-centric measures or effectiveness:** Most common security metrics gathered include updates deployed and risk assessment scores, but tend to neglect user behaviors and attitudes. However, this can be a crucial way to gather insights into the level of understanding of users or employees regarding security measures.

Usability is defined as the extent to which people can use systems, products, and services with effectiveness, efficiency, and satisfaction to accomplish their goals in a specified context of use. It is made of three key components: effectiveness, efficiency, and satisfaction^[3], described as follows:

- **Effectiveness** is whether people can successfully achieve their goals, for example, whether an employee can report a phishing email to cybersecurity staff.
- **Efficiency** refers to the resources (e.g., time, cognitive) used to achieve those goals, for instance, how long it takes an employee to authenticate to an application successfully.
- **Satisfaction** is the intersection between a user's physical, cognitive, and emotional responses when using a system, product, or service, and how well the user's needs and expectations are met.

Refer to the following table:

Type	Description
Motivate and empower, build relationships, practice empathy	Foster a culture of security by making users feel valued and supported. Build trust, listen actively, and treat users as partners rather than obstacles.
Do not rely on fear or punishment alone	Avoid scare tactics or punitive measures. Fear can lead to resistance or avoidance and ultimately to an inability to identify the root cause. Instead, encourage positive behavior through rewards, recognition, and support.
Be context-aware and act as a translator	Communicate security messages in a way that resonates with different roles and business contexts. Translate technical risks into business impact, and avoid using technical and security jargon.

Type	Description
Use multiple formats and media	Reinforce learning through varied channels—videos, infographics, interactive simulations, newsletters, and live demos—to match diverse learning preferences.
Conduct usability testing	Test security features and policies (e.g., MFA, secure access portals) for ease of use. Friction leads to workarounds, so usability must be prioritized. Avoid large rollouts of sweeping changes to security infrastructure and use learnings from usability tests or pilots to refine.
Offload burden with tools to support users	Reduce cognitive load by providing user-friendly tools (e.g. password managers, phishing filters) that automate or simplify security practices without sacrificing protection.
Apply a risk-based approach with a framework	Use formal risk frameworks (e.g., NIST RMF 2.0, ISO 27005, ISO 42001 for AI-related risk) to prioritize controls based on impact and likelihood, rather than applying a one-size-fits-all policy.
Understand the support capability of users	Assess the digital literacy and capacity of users before rolling out controls. Tailor training and support to their skill levels and roles.
Gather user-centric data and metrics to drive Improvements	Collect feedback, behavior data, and incident patterns to continuously adapt and improve both training and security measures with users in mind.

Table 11.2: Measures to adopt for human-centric security

Risk-based approach

We have covered risk-based prioritization in [Chapter 7, Manage Vulnerabilities](#). A risk-based approach focuses on identifying and addressing the most significant threats and vulnerabilities first, rather than applying blanket controls across all systems. Security can become complex when organizations attempt to protect every asset equally, often leading to redundant tools, conflicting policies, and a degraded user experience. This complexity is further compounded when there is pressure to do it all without aligning with actual risk levels. The key lies in applying the most stringent security controls to high-risk assets or data, while balancing other non-functional requirements, such as performance, scalability, and usability. This approach ensures that security is practical, sustainable, and tailored to the business's real-world needs. Here is how one can apply a risk-based approach to implementing security:

- **Risk assessment:** Utilize structured frameworks that enable teams to

evaluate threats, assess their impact, and prioritize mitigations, such as STRIDE (for threat modeling), CVSS, and MITRE ATT&CK (for risk prioritization), and FAIR (for quantifying risk in financial terms), which provide repeatable methods to guide security conversations.

- **Prioritization:** Risk matrices and prioritization techniques should then be used to further aid in identifying high-risk areas by visually mapping likelihood against impact.
- **Aligning risk decisions with business goals:** This ensures that security efforts protect what matters most while enabling innovation and agility.
- **Risk tolerance and acceptance principles:** Recognizing that not all risks can or should be eliminated. What is considered **acceptable risk** varies across contexts and organizations; for instance, a startup might tolerate more operational risk than a regulated enterprise.

Risk assessment

Starting with applying STRIDE, we would first identify the key threats listed as follows:

- Credential theft | Spoofing
- Malware introduction | Tampering, elevation of privilege
- Data exfiltration | Information disclosure, elevation of privilege
- Session-based access | Information disclosure, repudiation
- Compliance/data loss | Information disclosure, tampering

Then, we would apply **factor analysis of information risk (FAIR)** to prioritize. [Table 11.3](#) summarizes the likelihood and impact of the risks identified in the preceding section:

Risk	Likelihood	Impact	Derivation (FAIR lens)	Priority
Unauthorized access via credential theft	High	Critical	High frequency (password reuse, phishing) × High asset value (critical systems)	High
Compromised devices introducing malware	High	High	Devices unmanaged or bring-your-own; malware can escalate privileges, disrupt ops	High

Data exfiltration across segmented zones	Medium	Critical	An attack may need more effort, but the impact is a business-wide data breach	Med-High
Unauthorized access during active sessions	Medium	High	Harder to detect/prevent without monitoring; user context is harder to maintain	Medium
Covert data loss, compliance breach via tooling	High	Critical	Compliance fines, stealthy behavior, and tools may bypass controls	High

Table 11.3: Risk prioritization example

Risk prioritization

Risk matrices are an effective way to represent risks based on likelihood and impact visually. They help organizations categorize vulnerabilities into low, medium, high, or critical risk levels, as shown in [Table 11.4](#), and prioritize them accordingly. The following matrix is typically used in risk-based prioritization, which combines likelihood vs. impact:

		Impact				
		Negligible	Minor	Moderate	Significant	Severe
Likelihood	Very likely	Medium	High	High	Very high	Very high
	Likely	Medium	Medium	High	Very high	Very high
	Possible	Low	Medium	Medium	High	Very high
	Unlikely	Low	Low	Medium	High	High
	Rare	Low	Low	Low	Medium	High

Table 11.4: Sample risk matrix

Another approach is to use a manageability matrix such as the one in [Table 11.5](#). At the lower left of the chart, risks are less manageable and have a higher impact and imminence. They are therefore more severe and need to be mitigated sooner. Meanwhile, in the upper right of the chart, risks are more manageable, with less impact and a longer timeline for mitigation, shown as follows:

Multiple mitigations	1	2	3	4	4
Likely	1	2	3	3	4
Possible	1	2	2	3	3
Some mitigations	1	1	2	2	2
No mitigations	0	0	0	0	0
Mitigations required vs. risk occurrence	Past	Imminent	Near term	Distant	Long term

Table 11.5: Sample manageability matrix

Business goals

Beyond balancing likelihood and impact to manage security complexity while maintaining business agility, each risk should also be assessed with business goals and NFRs, such as performance and usability. Here are a few examples of how this addresses some of the preceding risks and how a balanced approach could work:

- **Unauthorized access via credential theft** is a high-priority risk as it scores high on both likelihood and impact. While enforcing MFA everywhere may cause some user fatigue or login friction, this is an acceptable trade-off to prevent catastrophic breaches and aligns well with business risk tolerance. However, when introducing it, it is best to roll out gradually and tweak the experience and technology based on feedback, and educate users as you go along.
- **Data exfiltration across segmented network zones**, while less likely, could have a high impact. Micro-segmentation and **role-based access controls (RBAC)** reduce lateral movement but can disrupt legitimate workflows. Thus, the threat modeling exercise could help refine which roles require which access, preventing over-segmentation. Applying just enough segmentation supports business operations while maintaining a strong security posture.
- **For unauthorized data access during sessions**, where the impact is high but the likelihood is moderate, session monitoring is a lightweight, non-intrusive control. The goal here is not to prevent access upfront, but to detect misuse in real-time, striking a balance between usability

and oversight.

These are some examples that need to be evaluated in the context of your organization and ecosystem before implementing the mitigations and solutions.

Architecture optimization

As we covered in the introduction, implementation of Zero Trust requires sound architecture principles supported by technologies needed to implement specific security requirements (across many areas such as network firewalls, secret management, security scanning tools, Key management, and security posture management). Here are some of the reasons implementing new capabilities and buying security tools adds complexity:

- **Overlapping functionality:** Buying multiple tools for the same purpose (e.g., two endpoint detection tools or multiple identity providers) results in duplicate alerts, competing agents, user confusion, and excessive costs.
- **Lack of integration:** Tools that do not work well together often require custom integration or manual processes (e.g., SIEMs that do not pull logs from specific sources), which increases the workload for teams.
- **Fragmented policies:** Using different firewalls, **Cloud Access Security Broker (CASB)**, or DLP tools across environments results in inconsistent policy enforcement and audit complexity.
- **Multiple UIs and training requirements:** Each new product has its own user interface, dashboard, configuration model, and alert systems—teams must learn and manage various interfaces.
- **Hidden operational costs:** Some tools require dedicated infrastructure or frequent updates, driving up the total cost of ownership and operational effort.
- **Alert fatigue:** Excessive tool-generated alerts can overwhelm teams, causing them to overlook or miss critical events.
- **Vendor lock-in:** Buying into a vendor suite that does not support open standards may limit future flexibility and complicate migrations.
- **License and access management overhead:** Managing users,

entitlements, renewals, and license tiers across multiple security products adds an administrative burden.

- **Increased attack surface:** Every new product and tool introduces new vulnerabilities (e.g., misconfigurations, third-party integrations).
- **Cost overruns and longer time to value:** Complex solutions often result in overlapping licenses, consulting fees, and training expenses, leading to increased costs and prolonged time-to-value.

Balancing security and complexity will improve operational efficiency, user experience, security effectiveness, incident response, and cost optimization as follows:

- **Operational efficiency:** Simpler environments are easier to manage, automate, and scale, leading to faster business workflows.
- **User experience:** Less intrusive controls reduce friction for users, leading to a faster and more effective experience, which improves adoption and productivity.
- **Security effectiveness:** Fewer, better-integrated tools reduce attack surfaces and blind spots.
- **Incident response:** Unified logging and alerts enhance response times and minimize errors, thereby reducing the need for manual monitoring and constant tuning.
- **Cost optimization:** Avoid unnecessary spending and focus on ROI from fewer but more impactful solutions.

Table 11.6 presents a list of key questions to ask when simplifying the architecture during the implementation of a new solution, capability, or product:

Area of concern	Questions to ask
Tool or capability assessment	• Is this a unique or unmet requirement or risk? • Is this duplicating an existing capability or tool?
Integration	• Does the proposed solution integrate with existing platforms? • Does it support open APIs and automation?
Consolidation	• Can the solution replace multiple capabilities or streamline workflows? • Can we use platform-based versus point solutions?
Usability	• Is the product easy for the team to configure and use? • Does it add complexity, dashboards, consoles?

Alerts	• Can we reduce noise by prioritizing alerts? • Are there any false positives or duplicate alerts that the solution can lead to?
Vendor risk	• Are we overly dependent on a single vendor? • What is the vendor roadmap for integration and support? Is there a risk of the vendor
Operational overhead	• What maintenance is required? • Does the solution require dedicated staff or training?
Policy	• Can the solution enforce policies across a hybrid environment (multiple cloud providers, on-premises)? • Is it compliant with the standards and regulations applicable?
Scalability	• Will the solution scale with a growing business? • Is the solution flexible to a changing organizational landscape and requirements?

Table 11.6: Simplifying architecture concerns

Phased implementation

Deploy Zero Trust incrementally to isolate and resolve NFR conflicts (e.g., pilot groups for usability testing). Let us consider two hypothetical, contrasting cases to understand the value of a phased rollout in security implementation:

- A mid-size service provider successfully secures remote access by starting with cloud-based MFA and centralized identity, achieving rapid deployment, strong user adoption, and an 85% reduction in phishing risk. The organization also phases the rollout across the user community, leading to reduced training requirements and the ability to learn from feedback to refine the user journeys and approach.
- A large financial firm attempts to implement military-grade Zero Trust protection by deploying DLP, encryption, and micro segmentation simultaneously. The lack of a phased rollout results in the need for extensive user education, user frustration, and a threefold increase in operational overhead. The program also takes significantly longer to deliver business outcomes, leading to people using unsafe workarounds.

These examples highlight that gradually introducing security controls not only improves adoption and effectiveness but also prevents the unintended

consequences of over-engineering.

Automation

Lastly, automation is your best friend to reduce complexity. It plays a crucial role in simplifying modern security environments by eliminating manual effort, minimizing configuration errors, and ensuring the consistent enforcement of controls. Rather than relying on human intervention to apply policies or respond to threats, organizations can use orchestration and automation tools such as Terraform, GitOps, Kubernetes-native security operators, policy as code (using Open Policy Agent, Sentinel) and AI-driven SecOps to deploy and enforce security policies across distributed systems automatically. Many companies nowadays use infrastructure as code to apply network segmentation rules, reducing deployment time from weeks to hours while ensuring consistency across environments. Although this approach can introduce configuration drift or supply chain risks, these can be countered through security scanning and SBOM verification. They can also utilize automated identity governance to provision and deprovision user access based on role changes, thereby reducing insider risk without incurring operational delays. By automating repetitive tasks and utilizing orchestration to apply policies at scale, organizations can enforce robust security controls without compromising performance or overburdening teams.

Conclusion

The increasing complexity of security ecosystems, with an average of 76 security tools per organization, leads to operational challenges and potential vulnerabilities. This chapter examined the trade-off between security and other non-functional requirements, including availability, performance, and usability, while highlighting the impact of complexity on cybersecurity.

Product design and software development require a pragmatic approach, balancing security requirements with resource, cost, timeline, and environmental constraints. A human-centric approach to security, as recommended by NIST, emphasizes usability, communication, and user-centric measures to foster a robust security culture. A risk-based approach,

utilizing frameworks, helps prioritize controls based on their impact and likelihood, ensuring a balanced and effective security strategy. Zero Trust implementation is a journey that should be implemented with incremental deployments and constant user feedback, improving adoption and effectiveness while preventing over-engineering. Automation simplifies modern security environments by eliminating manual effort and minimizing configuration errors. In the next and final chapter, we will cover various policies and frameworks, exploring why they exist and covering some of the key regulations you will encounter.

Points to remember

- **Security complexity:** Security is a key non-functional requirement, focusing on data protection, access control, and compliance. However, excessive security tools and integrations can lead to increased complexity, potentially compromising security.
- **Security and other requirements:** Neglecting non-functional requirements, such as availability, performance, and usability, can lead to post-release issues that compromise system performance and user experience, including poor user experience, system instability, and security vulnerabilities. For example, strict latency requirements can lead to reduced security measures, creating vulnerabilities, and complex security controls can hinder user experience and slow down access processes.
- **Security trade-offs:** Balancing security needs with resource, cost, timeline, and usability constraints in product design and development, and focusing on building a viable product by addressing known needs and anticipated priority risks, rather than aiming for a perfect solution.
- **Human-centric security approach:** NIST recommends a human-centric approach to security, emphasising the importance of understanding user behaviour and needs.
- **Common pitfalls in cybersecurity:** Assuming users are naive, not tailoring communications, creating insider threats due to poor usability, overemphasizing security, relying on punitive measures, and failing to

consider user-centric measures:

- **Importance of user-centric metrics:** Gathering insights into user behaviours and attitudes is crucial for understanding the effectiveness of security measures.
- **Security culture:** Foster a culture of security by making users feel valued and supported, avoiding fear-based tactics, and communicating security messages in a relatable way.
- **User-centric security measures:** Provide user-friendly tools, tailor training to skill levels, and gather user feedback to improve security measures.
- **Risk-based security approach:** Prioritize security controls based on impact and likelihood using frameworks such as NIST RMF or ISO 27005.
- **User support and training:** Assess user digital literacy and capacity, then provide training and support tailored to their needs:
 - **Risk assessment method:** Risk matrices visually represent risks based on likelihood and impact, helping organisations prioritise vulnerabilities.
 - **Risk categorization:** Risks are categorised into low, medium, high, or critical levels based on their impact and likelihood.
 - **Risk prioritisation:** Organisations prioritise risks based on their impact and likelihood, focusing on those with the highest potential damage.
 - **Risk severity and mitigation:** Risks with higher impact, imminence, and less manageability require immediate mitigation.
- **Architecture optimization:** Implementing multiple security tools can lead to overlapping functionality, a lack of integration, fragmented policies, and increased operational costs. Simpler environments improve operational efficiency, user experience, security effectiveness, incident response, and cost optimisation. Key considerations for simplification include evaluating the necessity of tools, assessing integration capabilities, and considering consolidation opportunities when simplifying the architecture.

- A phased rollout of security features enhances the adoption and effectiveness of security controls while minimizing unintended consequences.
- Automation simplifies security environments by reducing manual effort, minimizing errors, and ensuring consistent enforcement of control.

References

1. <https://www.gartner.com/smarterwithgartner/is-the-cloud-secure>
2. <https://panaseer.com/resources/reports/2022-security-leaders-peer-report>
3. https://securitydelta.nl/media/com_hsd/report/690/document/ENIS-A-Threat-Landscape-2024.pdf
4. <https://www.nist.gov/publications/users-are-not-stupid-six-cyber-security-pitfalls-overturned>

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



CHAPTER 12

Policies, Frameworks and Compliance

Introduction

The concept of regulation dates back to ancient civilizations, such as those in Mesopotamia, where rules governed trade and business practices. The *Industrial Revolution* gave rise to new challenges, including unsafe working conditions and environmental impacts resulting from rapid industrialization, which in turn led to the introduction of new regulations. The earliest standards for evaluating the security of computer systems, the **Trusted Computer System Evaluation Criteria (TCSEC)**, also known as the *Orange Book*, were established in 1983.^[1] The standard categorized security levels and influenced subsequent security standards. The first quality management standard (ISO 9000 family) from the **International Organization for Standardization (ISO)** was published in 1988^[2], and TCSEC was later superseded by Common Criteria (ISO/IEC 15408). The first modern financial regulator can be traced to the Dutch authorities, who enacted the first recorded financial regulation—a ban on short selling in 1610. The Bank of England was founded in 1694 as the government's banker and debt manager, playing a crucial role in financing the war. It then gradually assumed central banking functions. The **Office of the Comptroller of the Currency (OCC)** in the United States is also cited as

the first dedicated financial regulator established by the National Banking Act of 1863.

There are numerous regulatory bodies and regulations worldwide, all of which aim to protect the interests of consumers and the public, safeguard data, drive consistency and accountability, and maintain trust and fairness. Policies and frameworks supplement regulations, helping to transform complex regulatory and security requirements into actionable, standardized practices that protect organizations, ensure legal compliance, and foster trust. Their origins lie in the need to protect the public and manage risk, evolving from ancient codes to the sophisticated digital standards of today.

Structure

The following topics are covered in the chapter:

- Compliance overview
- Compliance and regulatory organizations
- Compliance regulations
- Standards and frameworks
- Emerging trends and technologies

Objectives

When working in regulated industries, you will encounter policies, frameworks, and compliance in the context of organizational governance, risk management, and legal obligations. This chapter will introduce you to why these exist. It emphasizes the role of compliance in ensuring that organizations meet legal requirements, industry standards, and ethical expectations, preventing costly violations and reputational damage. Additionally, it outlines various global and industry-specific regulatory requirements that organizations must navigate, including data protection laws (e.g., GDPR), financial regulations (e.g., AML), and environmental standards.

Note that this chapter includes various checklists of regulatory bodies,

regulations, standards, and frameworks. However, they are not meant to be exhaustive; one should research the regulatory landscape and standards applicable to them based on the industry and services they offer.

Compliance overview

Regulatory compliance encompasses numerous agencies, bodies, and standards. It is thus crucial to understand the various terminologies involved. *Table 12.1* outlines the different terms involved, along with brief definitions of each:

Term	Definition
Regulation	Regulation refers to a rule, requirement, or control, often established by government or an independent organization, that limits, steers, or otherwise controls social or economic behavior. Examples: Health Insurance Portability and Accountability Act (HIPAA), General Data Protection Regulation (GDPR), UK Money Laundering Regulations (MLR).
Regulator	A regulator is a person or organization, typically appointed or authorized by government, whose job is to control an activity or process to ensure it operates as intended and in accordance with laws, standards, and regulations. Examples: The US Securities and Exchange Commission (SEC), Financial Conduct Authority (FCA) in the UK.
Policy	A policy is a set of coherent decisions, guidelines, or a statement of intent adopted by an organization or government to guide present and future choices and achieve specific long-term goals. They typically provide directions and a clear set of guidelines. Examples: Data privacy policy, environmental policy, or a company's code of conduct.
Framework	A framework is a conceptual or structural foundation that provides support, organization, or guidance for building something more specific. Security frameworks provide standards, policies, procedures, and best practices to enhance an organization's security posture. Examples: NIST Cybersecurity Framework (CSF), Service Organization Control 2 (SOC2)
Standard	A standard is a documented set of rules, guidelines, or characteristics established by consensus and approved by a recognized body. Typically developed by industry experts and standards organizations, they ensure that activities, products, or services are performed or produced consistently, safely, and

	efficiently. They are typically voluntary, unless incorporated into law or contracts.
	Examples: Payment Card Industry Data Security Standard (PCI DSS), ISO 27001: Information security management systems

Table 12.1: Regulatory compliance terminology

Regulations are established and enforced by governments or regulatory authorities to ensure safety, fairness, and compliance with the law; they are legally binding on organizations engaged in trade. Organizations must comply with these regulations, and failure to do so can result in legal penalties, loss of market access, or other sanctions.

Standards and frameworks are generally voluntary and are not legally binding unless they are specifically referenced or incorporated into regulations or legislation. However, adherence to these assures customers that an organization follows industry-recognized practices, maintains a structured approach to managing risk, and is committed to transparency, accountability, and continuous improvement in areas such as security, privacy, and compliance.

Policies are internal rules or guidelines that organizations adopt to guide their own operations and compliance efforts. They are binding within the organization but do not have legal force outside of it unless required by regulation, and are usually inspired by industry standards and frameworks.

Compliance and regulatory organizations

Regulators ensure that the system is transparent and fair, protecting the interests of consumers from those of the institutions in their trade and domain. For instance, financial regulators oversee various areas like banking, securities, insurance, and privacy. They set the standards and rules, and monitor and enforce these standards, promoting stability and integrity. As institutions become more global, regulatory frameworks are becoming increasingly complex, and the need for specialized oversight has grown. Most jurisdictions typically have multiple regulators, including a central bank, an authority to oversee banks and insurers, as well as those responsible for securities and commodities markets, and those responsible for data privacy and protection.

Tables 12.2, 12.3, and 12.4 list some of the key regulatory organizations globally, their geographic jurisdictions, and the scope of their activities:

Org	Geo	Full name	Scope
Federal Reserve	US	Federal reserve	It is the central bank and it oversees monetary policy, supervises financial institutions, and maintains financial stability.
SEC	US	Securities and exchange commission	Regulates the entire securities industry, including securities exchanges, brokerage firms, investment advisors, and public companies
OCC	US	Office of the comptroller of the currency	Charters, regulates, and supervises all national banks, federal savings associations, and federal branches and agencies of foreign banks
FinCEN	US	Financial crimes enforcement network	Anti-money laundering (AML) and counter-terrorism financing (CTF)
FINRA	US	Financial industry regulatory authority	Self-regulatory body—Regulation of brokerage firms and registered brokers, to protect investors and ensure the integrity and fairness of the U.S. securities markets
FTC	US	Federal trade commission	Protects consumers and promotes competition
CFTC	US	Commodity futures trading commission (CFTC)	Regulates the derivatives markets, including futures and options
CFPB	US	Consumer financial protection bureau	Protects consumers in the financial sector, including issues related to lending, payments, and financial products.
NCUA	US	National credit union administration	Regulates and supervises federal credit unions
FDA	US	Food and drug administration	The federal agency responsible for protecting and promoting public health by overseeing various products, including food, drugs, medical devices, and cosmetics.

Table 12.2: List of some key regulators across the US

Org	Geo	Full name	Scope
BoE	UK	Bank of England (*Nationalised 1946)	Sets the monetary policy and financial stability and regulates certain financial market infrastructures.

Org	Geo	Full name	Scope
PRA	UK	Prudential Regulation Authority	Prudential regulation and supervision to promote the safety and soundness of banks, building societies, credit unions, insurers, and major investment firms.
FCA	UK	Financial Conduct Authority	Regulating financial markets, including the oversight of brokerage firms and financial advisors.
ICO	UK	Information Commissioner's Office	Oversees data protection and privacy regulations, ensuring compliance with the UK GDPR.
PSR	UK	Payment System Regulators	Ensures payment systems serve everybody as well as they can by promoting competition, serving customers, ensuring transparency, and resiliency.
MHRA	UK	Medicines and Healthcare products Regulatory Agency	Regulates medicines, medical devices, and blood components for transfusions
ECB	EU	European Central Bank	Monetary policy in the Eurozone is supervised by significant banks under the single supervisory mechanism (SSM) .
EBA	EU	European Banking Authority	Ensures effective and consistent banking regulation and supervision across the EU.
ESMA	EU	European Securities and Markets Authority	Enhances investor protection and promotes stable and orderly financial markets across the European Union (EU) .
EIOPA	EU	European Insurance and Occupational Pensions Authority	Oversees insurance and pension sectors, ensuring consumer protection and financial stability.
EDPB	EU	European Data Protection Board	Ensures consistent application of data protection laws across the EU, particularly the GDPR.
EMA	EU	European Medicines Agency	Protects and promotes human and animal health by evaluating and monitoring medicines within the EU.

Table 12.3: List of some key regulators across the EU and UK

Org	Geo	Full name	Scope
FATF	All	Financial Action Task Force	Global watchdog for money laundering and terrorist financing that sets international standards to prevent these illegal activities. It is not a law enforcement

			agency but develops policies and guidelines.
MAS	SG	Monetary Authority of Singapore	Financial regulator that oversees banking, capital markets, insurance, and payment systems.
TGA	AU	Therapeutic Goods Administration	Evaluates, assesses, and monitors products that are defined therapeutic goods (medicines, medical devices, and biologicals)

Table 12.4: List of some key global regulators and other jurisdictions

Real-world relevance

Here are some concrete steps that any business or enterprise should follow to understand the regulators:

1. Understand the regulatory landscape, which requires you to identify the regulators for the domain you operate. If you are a bank, look for the organizations that regulate banks in your jurisdiction. If you are a food or medicine business, look for the relevant food and drug regulators.
2. Have a clear understanding of your customer base and their geographic location. Depending on the nature of your business, how you interact with your customers, and the specific domain (e.g., financial, data, health, pharmaceutical), you may be subject to regulations governing the customers you serve.

Compliance regulations

Regulations are obligations imposed by government laws, statutes, and regulatory agencies. These are legally binding on organizations, and any non-compliance can result in penalties, fines, or loss of license to operate. Different types of regulations can apply to an organization, its employees, and systems or products. The following are some of these types:

- **Data protection and privacy** govern the collection, storage, and sharing of personal data. Examples include the GDPR and the California Consumer Privacy Act.
- **Employment and labor compliance** refers to laws that establish standards for employee rights, fair labor practices, compensation, and anti-discrimination.

- **Consumer protection** laws protect consumers from unfair trading practices and ensure they receive accurate product information and service.
- **Financial and tax compliance** rules for financial reporting, tax filings, anti-money laundering, and accounting standards. Examples include annual reports and statements filing, as well as AML compliance.
- **Health and safety regulations** encompass the standards for workplace safety, food safety, and the communication of hazards and emergencies.
- **Environmental regulations** cover the discharge of pollutants into air and waterways, and the management of hazardous waste.
- **Industry-specific regulations** are tailored to institutions operating in specific industries, such as financial services, healthcare, food and beverage, construction, and others.
- **Corporate or internal compliance** policies and procedures to ensure ethical conduct, manage risk, and meet legal and organizational standards. This includes codes of conduct, anti-bribery and corruption policies, confidentiality policies, and conflict of interest policies.

Let us discuss some of these and their implications for cybersecurity and Zero Trust.

Data protection

Data protection and privacy of individuals apply to all businesses and organisations (entities) operating in most jurisdictions. The scope includes data processing, subject rights, impact assessments, rights to access, rectification, erasure, processing, portability, and object. All of these have significant implications for products and propositions in the appropriate jurisdiction.

The GDPR is a comprehensive regulation that ensures the protection of personal data, based on strict principles of consent and privacy.

Other jurisdictions have released their versions: **California Consumer Privacy Act (CCPA)**, **Personal Information Protection and Electronic Documents Act (PIPEDA)** in Canada, UK Data Protection Act, and Australia Privacy Act.

Lastly, HIPAA best practices are influential, though not binding, in

jurisdictions aside from the US, and it is based on the protection of people's health information. Best practices in this area still apply to most countries.

Consumer duty and protection

Consumer duty and protection are regulations and principles to protect consumers during the sales and purchase of products. These can also extend to providing consumers with the support they need, when they need it, clear communications, products, and services that meet their needs, and offer fair value.

These regulations require organizations to prioritize customer interests, safeguard their data, provide transparent communication, and ensure fair value. To achieve this, enhanced data protection, continuous monitoring and authentication, clear communication, and audit trails are required, in addition to limiting the impact of cyber incidents, fraud, or service interruptions.

Some examples include *Consumer Duty* (UK), *Consumer Rights Directive* (EU), and **United Nations Guidelines for Consumer Protection (UNGCP)**.

Corporate governance and accounting

There are various regulations globally that aim to ensure effective, ethical, and sustainable management, balancing the interests of stakeholders, shareholders, management, employees, customers, suppliers, financiers, etc. Increasingly, these regulations have integrated security requirements due to growing risks of cyber threats, financial fraud, data breaches, internal corporate fraud, and deceptive accounting practices. Some of these requirements include board accountability for security, risk management, internal controls, incident response, and reporting.

The increased focus on cyber threats and security measures requires boards to oversee effective risk management, mitigate risks impacting financial stability and stakeholder value, implement adequate controls to prevent, detect, and respond to fraud, errors, and unauthorized activities, as well as test incident response plans.

Examples include the UK Corporate Governance Code, the Sarbanes–Oxley

Act, the Dodd-Frank Act, the SEC, and the EU Accounting Directive.

Anti-money laundering

Jurisdiction-specific regulations to prevent and detect money laundering and terrorist financing activities, and assist government agencies. These regulations apply to a wide range of financial institutions and other businesses, covering multiple aspects of financial transactions and operations, including customer identification, transaction monitoring, record-keeping, and reporting suspicious activities.

Some of the key Zero Trust implications include verifying customer identities during onboarding and transactions, monitoring transactions, detecting suspicious activities, maintaining record-keeping and audit trails, and implementing data protection measures.

Examples include the **Bank Secrecy Act (BSA)** in the US, the Money Laundering Regulations in the UK, the **Anti-Money Laundering Directive (AMLD)** for Europe, and others.

Financial

In addition to data protection and anti-money laundering, various other financial regulations apply to financial firms, including banks, payment institutions, investment firms, insurance companies, and crypto-asset service providers. Let us look at some of them:

- The **Payment Services Directive (PSD2)** (upcoming PSD3 in draft and expected soon as of writing this book) is an EU law that regulates payment services to promote innovation.
- **Open Banking** is also primarily driven by PSD2 mandates, which secure data sharing via APIs and prioritize customer protection through robust security protocols and regulatory oversight.
- The **Digital Operational Resilience Act (DORA)**, enforceable since January 2025, in the EU aims to achieve a high standard of digital operational resilience.
- **Client Assets Sourcebook (CASS)** is a set of regulations for how financial firms must handle and safeguard client money and custody assets.

- In addition, financial regulators such as the **SEC** in the US and the **FCA** in the UK have their rules and principles for safe, fair, and transparent financial practices.

Medical and pharma

Regulations in the medical and pharmaceutical sectors also encompass various security considerations, including the protection of patient data, the cybersecurity of medical devices, and the integrity of the drug supply chain. HIPAA is already covered under data protection. In addition, the following are some other regulations in this space:

- The **EU Medical Device Regulation (MDR)** includes requirements for medical devices sold in Europe, such as registration, labeling, and post-market surveillance and vigilance.
- The **Food and Drug Administration (FDA)** also provides guidance on cybersecurity in medical devices, emphasizing a risk-based approach and the shared responsibility for cybersecurity.
- The **International Health Regulations (IHR)** are a framework established by the **World Health Organization (WHO)** that applies to member states, encompassing the security of public health information, informed consent, and non-discrimination, among other principles.
- The **Drug Supply Chain Security Act (DSCSA)** requires the implementation of an electronic, interoperable system to track and trace certain prescription drugs as they are distributed throughout the United States.
- **Good Manufacturing Practices (GMP)** and **Good Distribution Practices (GDP)** focus on meeting high-quality standards, ensuring the intended use, and fulfilling the requirements of product specifications. It has provisions for the physical security of manufacturing facilities and the control of materials to prevent contamination, cross-contamination, and tampering.

Real-world relevance

Understanding applicable regulations is key to identifying all the requirements that a product or solution must comply with. Even within an

organization, each product or application may need to comply with different regulations depending on the system's scope. If you were building a product from, say, a European nation and providing services to customers based in the US, you may need to comply with multiple US laws. Here are some examples of regulations you might have to comply with based on the domain you operate in:

- **Financial services:** Certain U.S. regulators may require foreign firms to register or comply with U.S. rules if they offer services to U.S. clients, even from abroad. Cross-border financial activity often triggers dual regulation.
- **Health:** Companies exporting health products to the US must comply with US FDA regulations, customs, and labeling requirements, regardless of physical presence.
- **Data privacy and protection:** If you process personal data of U.S. residents, you may need to comply with U.S. data privacy laws, such as the CCPA.
- **Commerce and consumer rights:** Selling to US consumers may require compliance with US consumer protection, tax, and intellectual property laws.

Once you understand the regulations in scope, the next step is to identify key security requirements. Let us look at data protection regulations as an example.

The GDPR within the EU and its variants across other jurisdictions outline the rights of individuals, placing specific requirements on the data management lifecycle. These regulations outline the legal standards for how organizations must manage and safeguard personal information, serving as an essential part of comprehensive data lifecycle management and data governance strategies.

Data residency is a critical part of most data protection rules (such as GDPR). GDPR requires data to be stored within the EU, and most other jurisdictions require the storage of data about consumers in their respective areas. This implies that the product you are building should host and process data in compliance with the applicable data protection rules.

Beyond hosting the data, the GDPR also outlines specific rights for

individuals, which are summarized in [Table 12.5](#). Meeting these requirements requires efficient and standardized data storage, management, retrieval, erasure, and request mechanisms. Each of these rights needs to be implemented by any product that stores customer data.

Right	Details
Informed	Links to transparency require clear and concise information about data being collected, usage, storage duration, and sharing practices provided to the user.
Access	Individuals can request copies of their personal data via Data Subject Access Request (DSAR) .
Rectification	Allow correction of inaccurate or incomplete information.
Erasure	Individuals can request organizations to stop processing their personal data and delete it.
Restrict processing	Request data to be stored but not processed (typically for a limited time).
Data portability	Allow subjects to obtain their data (provided to the controller under consent) in a <i>structured, commonly used and machine-readable format</i> . This is also a broader topic covered by evolving data portability frameworks, the EU Data Governance Act (2022) and the Digital Markets Act (2023) in the EU, which aim to enable data sharing, digital competition, and data portability.
Object	This is an absolute right that an individual has to object to the processing of their data.
Automated Decision-making	Right not to be subject to any automated decision-making that may have legal or significant consequences. In addition, the AI Act (EU 2024) also introduces a risk-based framework to regulate AI Systems and emphasizes human-in-the-loop oversight.

Table 12.5: GDPR individual rights

Standards and frameworks

Security standards and frameworks are a set of guidelines and standards established by a recognized body or industry experts. Standards outline specific requirements and best practices for safeguarding sensitive data, mitigating vulnerabilities, and ensuring the confidentiality, integrity, and availability of information systems. Security standards are not regulations but can be enforced by independent bodies or contractual terms. For

instance, the **Payment Card Industry Data Security Standard (PCI DSS)** is implemented by major card brands, such as Visa and Mastercard, and enforced through contracts. Frameworks, on the other hand, provide the structure for implementing and maintaining robust security programs. Frameworks serve as a reference architecture, providing a flexible, structured approach to managing cybersecurity risks through practices, tools, and processes. Examples include the **NIST Cybersecurity Framework (CSF)**, the **Center for Internet Security (CIS) Controls**, and the **Cloud Security Alliance (CSA)**. [Table 12.6](#) summarizes some of the popular standards and frameworks, along with their details:

Name	Definition	Details and scope
CIS	Center for Internet Security	Recommends practices to safeguard organisations against the most prevalent cyber threats. Multiple secure benchmarks across operating systems, cloud providers, software, DevSecOps tools, and network appliances.
ISO 27001	International Organization for Standardization 27001 standard	Helps organisations protect critical information, implement, maintain, and improve information security systems. Focuses on evaluating risks to an organisation's data protection system and information it needs to protect across all areas of the organisation, including people, process, and technology.
ISO 42001	International Organization for Standardization 42001 standard	It is the world's first AI management standard, which provides a structured, auditable, and holistic approach for organizations to manage AI. It addresses unique challenges of AI systems, including ethical considerations and transparency.
PCI DSS	Payment Card Industry Data Security Standard	Protect how cardholder data or sensitive information is accepted, processed, stored, or transmitted by businesses accepting card payments. Policies, procedures, management, network architecture, software design, and other critical protective measures.
SOC2	Service Organization Control 2	Ensure service providers securely manage data to protect the privacy and interests of their clients. Software offered to clients, including the data, infrastructure, and people and processes supporting it. Security, availability, processing integrity, confidentiality, and privacy.

Name	Definition	Details and scope
CSA CCM	Cloud Controls Matrix from Cloud Security Alliance	It is a framework of security controls and guidelines that helps organisations assess security risks with cloud computing and implement measures across the entire cloud supply chain from application development to run/ops. It is mapped to well-known frameworks such as CIS, ISO 27001, NIST, and PCI DSS. CSA Star Certification assures independent third-party assessments.
NIST CSF	U.S. National Institute of Standards and Technology Cybersecurity Framework	Best practices (voluntary) framework of security best practices for organizations to improve cybersecurity risk management. Exhaustive framework to protect organisations' cybersecurity posture using core functions—Govern (included in version 2.0), Identify, Protect, Detect, Respond, Recover.
NIST AI RMF	U.S. National Institute of Standards AI Risk	A risk management framework designed to equip organizations and individuals with approaches to increase trust worthiness of AI systems and help responsible design, development and deployment, and use of AI systems over time. It is intended as a practical, adaptable and operationalizable by organizations as AI technologies continue to develop.
Cyber essentials	UK government-backed scheme for cybersecurity	It is a framework that helps organisations protect themselves against online threats. Cloud services that host organisations' data or services, including IaaS, PaaS, and SaaS. Other examples: ENISA guidelines (EU), CMMS (US DoD), CSA Cyber Essentials (Singapore)

Table 12.6: List of key standards and frameworks

Security standards and frameworks are essential tools for organizations to systematically manage cybersecurity risks, comply with regulations, and safeguard their information assets. An organization should evaluate the well-known security standards and frameworks and adopt those that best align with its specific needs, industry, and risk profile. This may involve integrating elements from multiple frameworks or customizing them to their needs and context.

Emerging trends and technologies

Navigating the regulatory landscape, adopting standards and frameworks, creating and managing policies, and keeping them up to date can be tedious and resource intensive. There are now various tools and automated compliance management platforms that make it easier to track and organize compliance obligations and policies worldwide.

Table 12.7 covers some of these tools and technologies, which provide end-to-end management of regulatory, standards, and framework compliance. They help integrate cloud and information technology systems, support major compliance frameworks and regulations, automate evidence collection, dashboards, and reporting, as well as workflow and automation, and vendor management.

Tool	Details
Drata	• 20+ security/privacy frameworks, including SOC 2, ISO 27001, GDPR, PCI DSS, HIPAA, and more. • Provides good automation, deep integrations, and multi-framework support • Automation: Centralizes controls, continuous compliance, offers real-time dashboards, and supports multinational orgs. • Integrations: Long list of extensive deep integrations speeds security review and integrates with developer tools for compliance as code.
AuditBoard	• 30 preloaded frameworks, multiple versions of NIST, SOC2, PCI DSS • Out-of-the-box integrations providing seamless data exchange, testing automation, evidence collection, and continuous monitoring. • AI embedded workflows, enhanced data accuracy, and unified governance, risk, and compliance. • Integrations: 200+ out-of-the-box integrations, support for API based integrations.
Centraleyes	• Focused on governance, risk, and compliance, offers tens of pre-populated integrated risk and compliance frameworks, including NIST, ISO, SOC, PCI DSS • Centralized risk and compliance management, automated workflows and data collection, real-time dashboards and reporting.
Sprinto	• 15+ security/privacy frameworks, including SOC 2, ISO 27001, GDPR, PCI DSS, HIPAA, and more. • Provides good automation, deep integrations, and multi-framework support • Automation: Centralizes controls, magic mapping of checks and controls, offers real-time dashboards, and supports multinational orgs. • Integration: Hundreds of integrations out-of-the-box, along with API based custom integrations.

Table 12.7: Automated compliance tools and technologies

Conclusion

Integrating regulatory compliance into your business practices begins with developing a comprehensive understanding of the regulatory landscape, including identifying key regulators in your industry and the specific regulations that govern your trade and activities. Once this foundational knowledge is established, it is essential to determine the standards and frameworks relevant to your sector. For instance, ISO 27001 is generally suited for organizations in the information technology sector. At the same time, SOC 2 is especially applicable to **software as a service (SaaS)** providers. Next, define the core requirements that arise from these regulations, standards, and frameworks, tailoring them to your business's specific use cases, such as a payment platform that aligns with PCI DSS and adheres to robust data protection obligations.

To ensure these requirements are consistently met, thoughtfully embed compliance considerations into your software development lifecycle and use compliance checkpoints throughout the development process, from initial development through deployment. Adopt modern tools and technologies that leverage compliance-as-code to provide a comprehensive view of regulations and frameworks, and to measure and monitor compliance continuously. Finally, actively engage with industry professionals and subject matter experts to stay informed about evolving regulations and adopt best practices, thereby maintaining effective and up-to-date compliance throughout your organization.

Points to remember

- Regulation has a long history, but the earliest computer security standards, TCSEC, were established in 1983, and the first ISO standard was published in 1988. The first modern financial regulator, the Dutch authorities, enacted regulations in 1610, and the Bank of England, founded in 1694, played a crucial role in financial regulation.

- Compliance ensures organisations meet legal requirements, industry standards, and ethical expectations, preventing violations and reputational damage. Organisations must navigate various global and industry-specific regulations, including data protection, financial regulations, and environmental standards.
- Regulations come in various types, including data protection, employment and labour, consumer protection, financial and tax, health and safety, environmental, industry-specific, and corporate compliance.
- **Security standards and frameworks:** Guidelines and standards for safeguarding data, mitigating vulnerabilities, and ensuring information system security. Examples: ISO 27001 provides a framework for organizations to protect sensitive information, PCI DSS focuses on the protection of cardholder data, SOC2 ensures secure data management by service providers, ISO 42001 and NIST AI RMF (2023) for management of AI systems and associated risks, and NIST CSF 2.0 offers a voluntary framework for organizations to improve cybersecurity risk management.
- Various tools and automated platforms help manage compliance obligations, policies, and regulatory requirements.

References

1. <https://oteemo.com/blog/history-of-compliance/>
2. <https://www.iso.org/about>
3. https://www.regulation.org.uk/history-1-to_1974.html
4. <https://www.federalreservehistory.org/essays/federal-reserve-history>
5. <https://www.bankofengland.co.uk/about/history>
6. <https://www.nist.gov/cyberframework>
7. <https://artificialintelligenceact.eu/>
8. <https://gdpr-info.eu/>
9. <https://www.pcisecuritystandards.org/standards/>
10. <https://www.hhs.gov/hipaa/>

11. <https://www.iso.org/standard/27001>
12. <https://www.iso.org/standard/42001>

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

<https://discord.bpbonline.com>



Index

A

- ABAC, illustrating [93](#)
- Access Controls [90](#)
- Access Controls, solutions
 - Data Governance [192](#)
 - Secure Data Lifecycle (SDLC) [188](#)
- Access Controls, terms
 - Attribute-Based Access Control (ABAC) [92](#)
 - Least Privilege Access (LPA) [90](#)
 - Role-Based Access Control (RBAC) [91](#)
 - Segregation of Duties (SoD) [94](#)
- Access Identity [144](#)
- Access Identity, integrating [144](#), [145](#)
- Access Keys [110](#)
- Access Keys, configuring [111](#)
- Access Policy Design [116](#)
- Access Policy Design, elements [116](#), [117](#)
- Access Reviews [104](#)
- Access Tokens [111](#), [112](#)
- Access Tokens, configuring [112](#)
- AI-Assisted Scanning [234](#)
- AI-Based Prioritization [234](#)
- AI, configuring [296](#)
- AI/ML [120](#)
- AI Systems, checklists [314-316](#)
- AI, threats
 - AI-Driven/Social Engineering [298](#)
 - Automated Attacks [300](#)
 - Autonomous AI Agents [299](#)
 - Cost Complexity [300](#)
 - Deepfakes/Synthetic Media [298](#)
 - Economic/Political Manipulation [300](#)
 - Identity/Access Attacks [300](#)
 - Zero-Day/One-Day Attacks [300](#)
- AI, tools
 - Adaptive Response [304](#)
 - Alert Prioritization [304](#)
 - Deep Network Monitoring [302](#)
 - Endpoint Security [303](#)
 - Phishing Detection [303](#)

- Physical Security [303](#)
- Security Awareness [305](#)
- API Gateways [153](#)
- API Gateways, capabilities [153](#), [154](#)
- API Gateways, ensuring [154](#)
- API/Microservices [109](#)
- Artificial Intelligence (AI) [296](#)
- Attacks [10](#)
- Attacks, lessons
 - Access Identities [27](#)
 - Continuous Monitoring [29](#)
 - Data Security [28](#)
 - Network Segregation [28](#)
 - Vulnerability Management [29](#)
- Attacks, types
 - DDoS [12](#)
 - Insider Threats [13](#)
 - Malware [11](#)
 - MITM [13](#)
 - Phishing [10](#)
 - Ransomware [12](#)
 - Supply Chain [13](#)
- Attribute-Based Access Control (ABAC) [92](#)
- Automation [263](#)
- Automation/Orchestration [234](#)
- Automation, roles [263](#)
- AWS Architecture, services
 - Access Keys [110](#)
 - Access Tokens [111](#)
 - API/Microservices [109](#)
 - Mutual Authentication [113](#)
 - OAuth 2.0 [115](#)

B

Behavioral Biometric [120](#)

C

Capital One [208](#)

Capital One, configuring [209](#)

Capital One, terms [209](#), [210](#)

CIA Traid [35](#)

CIA Traid, pillars

- Availability [36](#)
- Confidentiality [35](#)
- Integrity [35](#)

CI/CD [73](#)

CI/CD, metrics [73](#)

- CI/CD, ways [73](#)
- Cloud-Based Access Control [119](#)
- CNAPPs [235](#)
- Colonial Pipelines [242](#)
- Colonial Pipelines, configuring [242](#), [243](#)
- Colonial Pipelines, terms [243](#), [244](#)
- Compliance Requirements [61](#)
- Containers [274](#)
- Containers, configuring [276](#), [277](#)
- Containers, evolutions [274-276](#)
- Containers, lists [279](#)
- Continuous Assessment [235](#)
- Continuous Monitoring [118](#), [199](#)
- Continuous Monitoring, areas [249](#)
- Continuous Monitoring, challenges [266](#)
- Continuous Monitoring, checklists [267](#)
- Continuous Monitoring, components [118](#), [119](#)
- Continuous Monitoring, ensuring [245](#), [246](#)
- Continuous Monitoring, strategies
 - Analysis Reporting [254](#)
 - Audit Compliance [255](#)
 - Automation [263](#)
 - Incident Reponse [256](#)
 - Robust Security Posture [262](#)
 - Security Awareness [260](#)
 - Security Operation Center (SOC) [255](#)
 - SIEM [251](#)
 - Team Management [260](#)
- Continuous Monitoring, technologies
 - Cloud-Based/Integrated Security [267](#)
 - Edge/IoT [267](#)
 - SOAR [267](#)
- Cryptography [173](#)
- Cryptography, approach
 - Digital Certificates [185](#)
 - FIPS [174](#)
 - Key Management [176](#)
 - Public Key Infrastructure (PKI) [186](#)
 - Tokenization [180](#)
- Cryptography, aspects
 - Algorithm Complexity [173](#)
 - Key Management [173](#)
- CSPM [229](#)
- CSPM, configuring [229](#), [230](#)
- CVSS [57](#)
- CWPPs [231](#)
- CWPPs, configuring [231](#)
- Cyberthreats [3](#)

- Cyberthreats, initializing [6](#)
- Cyberthreats, sector
 - Finance [4](#)
 - Healthcare [4](#)
 - Root Cause [5](#)
 - Transport/Logistics [4](#)

D

- Database Encryption [196](#)
- Data Classification [169](#)
- Data Classification, configuring [170](#)
- Data Governance [192](#)
- Data Governance, capabilities [192](#), [193](#)
- Data Governance, steps [193](#)
- Data Lifecycle Management (DLM) [189](#)
- Data Masking [182](#)
- Data Masking, configuring [182](#)
- Data Privacy [171](#)
- Data Protection [171](#)
- Data Protection, pitfalls [172](#)
- Data Security Requirements [169](#)
- Data Security Requirements, techniques
 - Data Classification [169](#)
 - Data Privacy [171](#)
 - Data Protection [171](#)
 - Regulatory Compliance [170](#)
- Datastore Security [194](#)
- Datastore Security, checklists [201](#)
- Datastore Security, technologies
 - AI/ML Data Protection [200](#)
 - Blockchain/Data Integrity [200](#)
 - Confidential Computing [200](#)
 - Quantum Computing [199](#)
- Datastore Security, terms
 - Continuous Monitoring [199](#)
 - Database Encryption [196](#)
 - Field-Level Encryption [197](#)
- Decentralized Identifiers (DID) [120](#)
- Device Trust Assessment [145](#)
- Digital Certificates [185](#)
- DLM, stages [189](#)
- Domain Name System (DNS) [152](#)

E

- Security Standards/Frameworks [343](#)
- Security Standards/Frameworks, ensuring [344](#), [345](#)
- Security Standards/Frameworks, flows [346](#)

Edge Computing [121](#)
Emerge Technologies [14](#)
Emerge Technologies, configuring [14](#)
Equifax [210](#)
Equifax, configuring [211](#)
Equifax, terms [211](#)

F

Federated Identity Management (FIM) [100](#)
Field-Level Encryption [197](#)
Field-Level Encryption, benefits [197](#)
Field-Level Encryption, integrating [197-199](#)
FIM, components
 Access Reviews [104](#)
 Compliance [105](#)
 Identity Governance Administration (IGA) [106](#)
 Identity Governance/Compliance [103](#)
 Identity Lifecycle Management [103](#)
 Single Sign-On (SSO) [100](#)
 Standards/Protocols [101](#)
FIPS [174](#)
FIPS, configuring [175](#), [176](#)
Firewalls [140](#)
Firewalls, configuring [141](#)
Firewalls, types [141](#)

G

Guidelines/Checklists [77](#)
Guidelines/Checklists, roles
 Developers Engineers [77](#)
 For Leaders [79](#), [80](#)
 Infrastructure Engineers [78](#)

H

Honeypots [232](#)
Honeypots, benefits [232](#)
Honeypots, considerations [232](#), [233](#)

I

IaC, tools [66](#)
Identity Governance Administration (IGA) [106](#)
Identity Governance/Compliance [103](#)
Identity Lifecycle Management (ILM) [103](#)
IGA, configuring [106](#), [107](#)
IGA, pitfalls [107](#), [108](#)
Incident Reponse [256](#)

- Incident Reponse, activities [257](#)
- Incident Reponse, functions [256](#)
- Incident Response [99](#)
- Incident Response, illustrating [99](#)
- Incident Response, pitfalls [99](#), [100](#)
- Incident Response, scenario
 - Access Procedures [99](#)
 - Disaster Recovery [99](#)
 - Incident Detection [99](#)
- Infrastructure as Code (IaC) [66](#)

K

- Key Management [176](#)
- Key Management, configuring [178](#), [179](#)
- Key Management, stages [177](#)

L

- Least Privilege Access (LPA) [90](#)
- LPA, illustrating [91](#)

M

- MFA, configuring [88](#), [89](#)
- MFA, solutions [89](#), [90](#)
- Micro-Segmentation [139](#)
- Micro-Segmentation, configuring [140](#)
- MITRE ATT&CK [56](#), [57](#)
- MOVEit [50](#)
- MOVEit, case study
 - Adopt Secure Coding [51](#)
 - Continuous Monitoring [52](#)
 - Incident Response [52](#)
 - Proactive Patch Management [52](#)
 - Secure Build [52](#)
 - Third-Party Risk Management [52](#)
- MOVEit, configuring [51](#)
- Multi-Factor Authentication (MFA) [88](#)
- Mutual Authentication [113](#)
- Mutual Authentication, configuring [113](#), [114](#)
- Mutual Authentication, ensuring [114](#)
- Mutual Transport Layer Security (mTLS) [184](#)

N

- Network Operations Center (NOC) [156](#)
- Network Policies [155](#)
- Network Security, case study
 - NotPetya [130](#)

- Sony [134](#)
- Target [134](#)
- TJX Companies [134](#)
- Network Security, checklists
 - Developer Engineers [160](#)
 - For Leaders [162](#)
 - Infrastructure Engineers [161](#)
- Network Security, ensuring [134-136](#)
- Network Security, technologies
 - AI/ML [158](#)
 - Edge Computing [159](#)
 - Remote Browser Isolation (RBI) [158](#)
- Network Segmentation/Isolation [137](#)
- Network Segmentation/Isolation, configuring [138](#), [139](#)
- NFRs, approach
 - Architecture Optimization [327](#)
 - Automation [329](#)
 - Human-Centric Approach [322](#)
 - Phased Implementation [329](#)
 - Risk-Based [324](#)
- NFRs, configuring [320-322](#)
- NOC, parts [157](#)
- Non-Functional Requirements (NFRs) [320](#)

O

- OAuth 2.0 [115](#)
- OWASP [58](#)
- OWASP, vulnerabilities [58](#)

P

- PAM, features [97](#), [98](#)
- Passwordless Authentication [119](#)
- PKI, elements [186](#)
- PKI, solutions [188](#)
- Plugins [74](#)
- Plugins, pitfalls
 - Desired Outcomes [76](#)
 - End-to-End Process [76](#)
 - Minimum Tools [76](#)
 - Prevent Proliferation [76](#)
- Plugins, tools [121](#)
- Prioritize Access [222](#)
- Prioritize Access, terms
 - Asset Criticality [222](#)
 - Business Impact Analysis [224](#)
 - Common Vulnerability Scoring System (CVSS) [223](#)
- Privileged Access [97](#)

- Privileged Access, attacks
 - Privileged Access Management (PAM) [97](#)
 - Secure Remote Access [98](#)
 - Short-Lived Credentials [98](#)
- Privileged Access Management (PAM) [97](#)
- Public Key Infrastructure (PKI) [186](#)

R

- RBAC, illustrating [91](#), [92](#)
- Regulators Across [121](#)
- Regulatory Compliance [170](#), [334](#)
- Regulatory Compliance, configuring [335](#)
- Regulatory Compliance, integrating [342](#)
- Regulatory Compliance, terms
 - Anti-Money Laundering [340](#)
 - Consumer Duty [340](#)
 - Data Protection [339](#), [341](#)
 - Governance/Accounting [340](#)
 - Medical/Pharmaceutical [341](#)
- Regulatory/Ethical Considerations [312](#)
- Regulatory Organizations [336](#)
- Regulatory Organizations, activities [336](#)
- Regulatory Organizations, steps [338](#)
- Robust Security Posture [262](#)
- Role-Based Access Control (RBAC) [91](#)

S

- SBOMs [233](#)
- Scan Test [70](#)
- Scan Test, steps [70](#)
- SDLC, configuring [189](#)
- SDLC, solutions [192](#)
- SDLC, stages [190](#)
- Secure Archival/Detention [193](#)
- Secure Archival/Detention, configuring [193](#)
- Secure Archival/Detention, solutions [194](#)
- Secure Build [67](#)
- Secure Build, pitfalls
 - Access Controls [70](#)
 - Audit Trails [69](#)
 - Automation [69](#)
 - Build Agent Isolation [69](#)
 - Secrets [69](#)
- Secure Build, steps [68](#)
- Secure Coding [62](#)
- Secure Coding, checklists [65](#)
- Secure Coding, integrating [63](#), [64](#)

- Secure Communication [146](#)
- Secure Communication, ensuring [146-148](#)
- Secure Communication, layers
 - Application [150](#)
 - Network Layer [149](#)
- Secure Communication, protocols
 - API Gateways [153](#)
 - Domain Name System (DNS) [152](#)
 - Network Policies [155](#)
 - Secure Routing [151](#)
 - Service Mesh [155](#)
 - Web Application Firewalls (WAFs) [152](#)
- Secure Data, case study
 - H&M [169](#)
 - LastPass [168](#)
 - Marriott [166](#)
 - Yahoo Breach [168](#)
- Secure Data Lifecycle (SDLC) [188](#)
- Secure Deploy [71](#)
- Secure Deploy, pitfalls
 - Environment Segmentation [72](#)
 - Fewer Ephemeral [72](#)
 - GitOps [73](#)
 - Immutable Infrastructure [72](#)
 - Parity Environments [72](#)
- Secure Deploy, steps [71](#)
- Secure Network Design [137](#)
- Secure Network Design, approach
 - Access Identity [144](#)
 - Device Trust Assessment [145](#)
 - Firewalls [140](#)
 - Micro-Segmentation [139](#)
 - Network Segmentation/Isolation [137](#)
 - Virtual Private Networks (VPNs) [142](#)
 - Zero Trust Network Access (ZTNA) [143](#)
- Secure Remote Access [98](#)
- Secure Routing [151](#)
- Security Attack [2](#)
- Security Attack, configuring [2, 3](#)
- Security Awareness [260](#)
- Security Awareness, terms
 - Backup/Recovery [261](#)
 - Backups [261](#)
 - Response Procedures [262](#)
- Segregation of Duties (SoD) [94](#)
- Service Mesh [155](#)
- Short-Lived Credentials [98](#)
- SoD, benefits [94](#)

- SoD, challenges [95](#)
- SoD, integrating [95](#)
- SoD, pitfalls [96](#), [97](#)
- Software Development [53](#)
- Software Development LifeCycle (SDLC) [53](#)
- Software Development, solutions
 - Business Impact [62](#)
 - Continuous Improvement [62](#)
 - Documentation [62](#)
 - Key Collaboration [62](#)
 - Prioritization [62](#)
 - Scope Boundary [62](#)
- Software Development, techniques
 - Compliance Requirements [61](#)
 - OWASP [58](#)
 - Threat Modeling [54](#)
- SolarWinds [46](#)
- SolarWinds, configuring [46](#), [47](#)
- SolarWinds, ensuring [47-49](#)
- SolarWinds, lessons
 - Centralized Logging [50](#)
 - Robust Code [49](#)
 - SBOM [49](#)
- Standards/Protocols, illustrating [103](#)
- Standards/Protocols, pitfalls [102](#)
- STRIDE [54](#)
- STRIDE, configuring [55](#), [56](#)
- Strong Authentication [86](#)
- Strong Authentication, configuring [86](#), [87](#)

T

- Team Management [260](#)
- Threat Actors [8](#)
- Threat Actors, types
 - Cybercriminals [8](#)
 - Hackivist [8](#)
 - Insider Threats [9](#)
 - Nation-State Sponsored [9](#)
- Threat Modeling [54](#)
- Threat Modeling, approaches
 - CVSS [57](#)
 - MITRE ATT&Ck [56](#)
 - STRIDE [54](#)
- Threats Mitigations [305](#), [306](#)
- Threats Mitigations, ensuring [307-310](#)
- Tokenization [180](#)
- Tokenization, architecture [181](#)
- Tokenization, configuring [181](#)

- Tokenization, terms
 - Data Masking [182](#)
 - Mutual Transport Layer Security (mTLS) [184](#)
 - Transport Layer Security (TLS) [183](#)
- Traditional Network [22](#)
- Traditional Network, advantages
 - Easy Traffic Management [23](#)
 - Protection [23](#)
 - Reduced Risk [23](#)
 - Simplicity [23](#)
- Traditional Network, assumptions [23](#), [24](#)
- Traditional Network, configuring [22](#)
- Traditional Network, limitations [25](#)
- Traditional Network, steps
 - Actions/Breach [27](#)
 - Exploitation [26](#)
 - Initial Access [26](#)
 - Initial Compromise [26](#)
 - Lateral Movement [27](#)
 - Research [26](#)
- Transport Layer Security (TLS) [183](#)

U

- Uber [84](#)
- Uber, case study
 - Cash App [86](#)
 - Plex [86](#)
 - Target [86](#)
- Uber, configuring [85](#)

V

- Virtual Environments [119](#)
- Virtual Environments, technologies
 - AI/ML [120](#)
 - Behavioral Biometric [120](#)
 - Cloud-Based Access Control [119](#)
 - Decentralized Identifier (DID) [120](#)
 - Edge Computing [121](#)
 - Password Authentication [119](#)
 - Regulators Across [121](#)
- Virtual Private Networks (VPNs) [142](#)
- VPNs, configuring [142](#)
- Vulnerability Management [212](#)
- Vulnerability Management, architecture [213](#)
- Vulnerability Management, case study
 - Capital One [208](#)
 - Equifax [210](#)

- Vulnerability Management, checklists [236-238](#)
- Vulnerability Management, concepts
 - Continuously Monitoring [229](#)
 - Prioritize Access [222](#)
 - Remediate [226](#)
 - Reporting [228](#)
 - Validate [227](#)
- Vulnerability Management, lifecycle [214](#)
- Vulnerability Management, techniques
 - AI-Assisted Scanning [234](#)
 - AI-Based Prioritization [234](#)
 - Automation/Orchestration [234](#)
 - CNAPPs [235](#)
 - Continuous Assessment [235](#)
 - SBOMs [233](#)
- Vulnerability Management, terms
 - Asset Discovery/Inventory Management [216](#), [217](#)
 - Configuration Management [218](#)
 - Dynamic Application Security Testing (DAST) [221](#)
 - Network Discovery [222](#)
 - Open-Source Software [220](#)
 - Software Composition Analysis (SCA) [219](#)
 - Static Application Security Testing (SAST) [221](#)
- Vulnerability Management, types [212](#), [213](#)

W

- Web Application Firewalls (WAFs) [152](#)

X

- XZ Utils [52](#)

Z

- Zero Trust [15](#)
- Zero Trust, capabilities
 - Code Secure [281](#)
 - Continuous Monitoring [291](#)
 - Secure Data [288](#)
 - Secure Identities [285](#)
 - Secure Networks [287](#)
 - Vulnerability Management [290](#)
- Zero Trust, configuring [16](#)
- Zero Trust, ensuring [17](#), [18](#)
- Zero Trust, integrating [37](#)
- Zero Trust, layers
 - Inner [37](#)
 - Middle [37](#)

- Outer [37](#)
- Zero Trust Network Access (ZTNA) [143](#)
- Zero Trust, pillars
 - Code Deploy [38](#)
 - Continuously Monitor [41](#)
 - Data/APIs [40](#)
 - Identity Access Management (IAM) [39](#)
 - Secure Network [39](#)
 - Vulnerability [41](#)
- Zero Trust, principles
 - Continuously Improve [33](#)
 - Dynamic Policies [32](#)
 - Least Privilege Access [32](#)
 - Secure All Flows [32](#)
 - Security Posture [33](#)
 - Verify Everything [31](#)
- Zero Trust, steps [33](#), [34](#)
- ZTNA, technologies [143](#)