Cyber Security – Standards and Best Practices

Securing Code. Fortifying Infrastructure. Empowering Trust.

रिज़र्व बैंक सूचना प्रौद्योगिकी प्राइवेट लिमिटेड
Reserve Bank Information Technology Pvt. Ltd.

Cyber Security
Standards and Best Practices

Version 1.0

THIS DOCUMENT IS CONFIDENTIAL. NOT FOR FURTHER DISTRIBUTION.

PUBLISHED BY:

RESERVE BANK INFORMATION TECHNOLOGY (REBIT)

DATE: 26TH SEPTEMBER 2025

Acknowledgements

This document is the result of the collaborative efforts of several individuals from RBI, ReBIT and the industry.

We express our sincere appreciation to the esteemed Technical Advisory Group (TAG) members who are subject matter experts and eminent leaders in the industry. Dr. Sandeep Shukla (Director, IIIT Hyderabad) for leading the team as Chairperson, Dr. S. Venkatesan (IIIT Allahabad), Dr. M.A.K.P Singh (IIT Kanpur), Mr. A. Gowthaman (DGM, DIT, RBI), and Mr. Dileep Mukundan (CISO, IFTAS) for their meticulous evaluation and insightful feedback, which have played a vital role in refining and enhancing the document.

We recognize the invaluable efforts of the ReBIT cyber security working group team Mr. Satish Patil (AVP-Cyber Security), Mr. Mangesh Shinde (Sr Manager-Network Security), Mr. Shreetej Sharma (Cyber Security Engineer), Mr. S. Jagadeeswaran Lead-Cyber Security), Mr. Rakesh Vishwakarma (Lead-Cyber Security), Mr. Yarlagadda Gnanateja (Cyber Security Engineer), Mr. Nikhil Pange (Manager – Network Security), and Samir Salvi (Lead-Cyber Security), Application Security team Mr. Dhananjay Mali (AVP-Cyber Security), Brahmananda Reddy (Chief Manager-Cyber Security), Abdul Kalam (Cyber Security Engineer), Rishab Lingam (Cyber Security Engineer) and Snehankit Chikhalekar (Senior Cyber Security Engineer) for their commitment and expertise. Additionally, we extend our heartfelt appreciation to the other contributors to this document including Mr. Shashank Ingawale, Mr. Amruta More, Ms. Sayali Suryawanshi, Ms. Rachita Jha, Ms. Piyali Choudhury and Mr. Sahil Solan for their contributions on design, editorial and Project Management Office.

Lastly, we would like to thank Mahesh Gharat (VP, ReBIT), whose dedication has been pivotal in shaping this document. We deeply value the time, expertise, and thoughtful recommendations provided throughout the review process. Your dedication and contributions have been essential in every stage of finalizing and upgrading this document to its final version, ensuring its relevance and excellence in establishing cyber security standards.

Santhosh George
CEO, ReBIT

Table of Contents

IT Infrastructure Security

List of Tables

List of Figures

IT Infrastructure Security Standards and Best Practices

Executive Summary

This document serves as a comprehensive guide to establish and maintain robust infrastructure security practices, aligned with OEM baselines and globally recognized standards such as the Center for Internet Security (CIS), the National Institute of Standards and Technology (NIST), and the Cybersecurity and Infrastructure Security Agency (CISA). It outlines actionable strategies, best practices and controls necessary to ensure a secure, resilient, and compliant IT environment.

The guidelines emphasize foundational security measures, including Authentication, Authorization, and Accounting (AAA), Identity and Access Management (IAM), and Zero Trust Architecture, to safeguard organizational systems against evolving threats. It highlights the importance of robust encryption, effective vulnerability management, and the implementation of comprehensive logging and monitoring frameworks for proactive threat detection and response.

Key focus areas include:
Network and Endpoint Security: Guidance on securing network infrastructure through segmentation, perimeter controls, and advanced tools such as firewalls, intrusion prevention systems, and endpoint protection solutions.

Server Security: Detailed guidance on securing server operating systems and applications through hardening, encryption & data protection and access control management. It also highlights the importance of periodic vulnerability assessments, penetration testing, and compliance with established security benchmarks (CIS, NIST SP 800-53 & NIST SP 800-171, ISO/IEC 27001) to protect sensitive data and applications.

Backup, Retention, and Disaster Recovery: Strategies to ensure data integrity and availability during disruptions, including detailed recovery objectives and testing methodologies. Database and Storage Security: Best practices for securing sensitive data, encryption techniques, and compliance with regulatory standards to minimize exposure to risks.

Email Security and Mobile Device Management: Recommendations to mitigate phishing, malware, and unauthorized access while ensuring the secure handling of corporate communications and devices.

Operational Excellence: Guidelines for patch management, hardware lifecycle planning, and periodic security assessments to maintain system stability and integrity.

This document is intended to equip security professionals, IT administrators, and organizational leaders with the tools and knowledge necessary to enhance their cybersecurity posture. By adopting the practices outlined herein, organizations can mitigate risks, ensure operational continuity, and maintain compliance with industry regulations.

For ease of implementation, the handbook provides step-by-step instructions, references to leading security frameworks and actionable recommendations tailored to address emerging challenges in a dynamic threat landscape. This ensures that security initiatives are both practical and scalable, enabling organizations to build a resilient infrastructure capable of withstanding current and future threats.

1. INTRODUCTION

1. Introduction

Effective IT infrastructure security relies on adherence to established global standards and best practices. This section introduces key frameworks—CIS, NIST, and CISA—that serve as pillars for creating resilient, secure, and compliant environments. Each framework provides unique guidelines and tools to address specific aspects of infrastructure security.

1.1 INTRODUCTION TO CIS

The Center for Internet Security (CIS) is a community-driven nonprofit organization, responsible for the CIS Controls® and CIS Benchmarks™, globally recognized best practices for securing IT systems and data. CIS leads a global community of IT professionals who continuously evolve these standards and provide products and services to proactively safeguard against emerging threats.

1.2 INTRODUCTION TO NIST

NIST develops cybersecurity standards, guidelines, best practices, and other resources to meet the needs of the U.S. industry, federal agencies, and the broader public.

NIST provides activities that range from producing specific information that organizations can put into practice immediately to longer-term research that anticipates advances in technologies and future challenges. ReBIT IT infrastructure security team has adopted CIS, CISA and NIST best practices to protect infrastructure against pervasive cyber threats.

NIST Cybersecurity Framework 2.0		
CSF 2.0 Function	**CSF 2.0 Category**	**CSF 2.0 Category Identifier**
Govern (GV)	Organizational Context	GV.OC
	Risk Management Strategy	GV.RM
	Roles and Responsibilities	GV.RR
	Policies and Procedures	GV.PO
Identify (ID)	Asset Management	ID.AM
	Risk Assessment	ID.RA
	Supply Chain Risk Management	ID.SC
	Improvement	ID.IM
Protect (PR)	Identity Management, Authentication, and Access Control	PR.AA
	Awareness and Training	PR.AT
	Data Security	PR.DS
	Platform Security	PR.PS
	Technology Infrastructure Resilience	PR.IR
Detect (DE)	Adverse Event Analysis	DE.AE
	Continuous Monitoring	DE.CM
Respond (RS)	Incident Management	RS.MA
	Incident Analysis	RS.AN
	Incident Response Reporting and Communication	RS.CO
	Incident Mitigation	RS.MI
Recover (RC)	Incident Recovery Plan Execution	RC.RP
	Incident Recovery Communication	RC.CO

Figure 1: NIST Cybersecurity Framework 2.0 (Source: NIST)

1.3 INTRODUCTION TO CISA

The Cybersecurity and Infrastructure Security Agency (CISA) is a federal agency in the United States dedicated to enhancing the nation's cybersecurity posture and protecting critical infrastructure. It plays a key role in helping organizations mitigate risks, respond to cyber threats, and enhance overall security resilience. CISA delivers actionable frameworks, such as the National Cyber Incident Response Plan, offers free vulnerability assessments, and shares timely threat intelligence through its awareness systems. Its comprehensive guidelines and tools empower organizations to implement robust IT infrastructure security practices and align with global standards like NIST and CIS, ensuring a proactive approach to managing evolving cybersecurity challenges.

2. FOUNDATIONAL SECURITY PRACTICES

2. Foundational Security Practices

2.1 AUTHENTICATION, AUTHORIZATION AND ACCOUNTING (AAA)

2.1.1 Authentication Methods

A comprehensive password policy emphasizes complexity requirements, including a minimum length, use of special characters, and a mix of uppercase and lowercase letters, ensuring resistance to brute force and dictionary attacks. Guidelines for periodic password changes further reduce the risks associated with prolonged exposure. Organizations are increasingly adopting password-less authentication methods, leveraging strategies such as biometrics and multi-factor authentication (MFA) to enhance security and reduce reliance on traditional passwords. MFA strengthens authentication by combining multiple factors like one-time passwords (OTPs), and biometric verification, providing a robust defence against unauthorized access and ensuring a secure authentication framework.

2.1.2 Authorization Mechanisms

Authorization mechanisms define how access to resources is granted based on specific rules or criteria. Three primary approaches—Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Policy-Based Access Management—offer varying levels of control and flexibility.

a. **Role-Based Access Control (RBAC)** simplifies access management by assigning permissions to roles instead of individuals. A role represents a group of permissions required to perform certain tasks, such as “Administrator” or “Manager.” Users are assigned roles, inheriting the permissions associated with those roles. RBAC enforces the Principle of Least Privilege by ensuring users have only the access they need to perform their duties. It is ideal for environments with well-defined organizational structures and static access needs.

b. **Attribute-Based Access Control (ABAC)** takes a more dynamic approach by evaluating attributes to determine access. Attributes can include user characteristics (e.g., department or job title), resource properties (e.g., data classification), and environmental factors (e.g., time of access or location). This flexibility allows ABAC to support fine-grained access control in complex scenarios where static roles are insufficient. It is particularly suited for dynamic and diverse environments, such as multi-tenant cloud platforms.

c. **Policy-Based Access Management** centralizes access decisions using predefined rules and conditions written in formats like XACML. Policies dictate access permissions by combining the principles of RBAC and ABAC, creating a hybrid approach that ensures consistency across systems. This mechanism is highly effective in organizations that require strict regulatory compliance, such as finance or healthcare. Centralized management allows for quick updates to policies, reducing the complexity of enforcing changes across multiple systems.

2.1.3 Accounting

Accounting, often referred to as auditing, is a crucial component of security that ensures transparency, accountability, and traceability in IT systems. It involves systematically tracking and recording user activities, managing sessions, and generating real-time alerts to detect and respond to unauthorized access attempts. By providing a detailed record of actions taken within a system, auditing enables organizations to monitor compliance, investigate incidents, and identify potential security breaches.

a. **User Activity Logging** is the foundation of auditing, capturing detailed records of user actions, including logins, file accesses, configuration changes, and system commands. These logs provide a chronological account of events, helping administrators understand who did what and when. Proper logging is essential for forensic analysis during incident investigations and serves as evidence in regulatory compliance audits. Logs should be centralized in a secure repository, often using tools like Security Information and Event Management (SIEM) systems, to facilitate correlation and analysis of logs generated across different systems.

b. **Session Management** ensures that user activities within a system are controlled and monitored during an active session. It involves assigning a unique session ID to each user login, tracking their actions, and enforcing timeouts or session expirations to prevent misuse. Proper session management mitigates risks such as session hijacking, replay attacks, session fixation, privilege Escalation, Cross Site Request Forgery (CSRF), Brute force on sessions and ensures that inactive sessions are terminated promptly to reduce the attack surface. Additionally, session data can be reviewed to identify unusual patterns, such as simultaneous logins from different locations, which may indicate compromised credentials.

c. **Real-Time Alerting for Unauthorized Access** Attempts adds a proactive layer to accounting by identifying suspicious activities as they occur. Alerts are triggered when predefined thresholds or conditions are met, such as multiple failed login attempts, unauthorized privilege escalation, or access from a blacklisted IP address. Real-time alerts enable security teams to respond swiftly, minimizing the potential damage from an ongoing attack. These alerts should be integrated into incident response workflows to ensure that critical events are escalated to the appropriate personnel without delay.

2.2 ZERO TRUST SECURITY

2.2.1 Overview of Zero trust in Enterprise Networks:

Zero-trust architecture (ZTA) is an enterprise cybersecurity architecture based on zero-trust principles which assumes no implicit trust in any user, device or network-whether inside or outside the organization's perimeter and designed to prevent data breaches and limit internal lateral movement. This publication discusses ZTA, its logical components, possible deployment scenarios, and threats.

2.2.2 Implementing Zero Trust in Enterprise Networks:

Zero Trust Model is a modern security strategy based on the principle never trust, always verify. Instead of assuming everything behind the corporate firewall is safe, the Zero Trust model assumes

breach and verifies each request as though it originates from an open network. When trying to implement zero-trust security, organizations need to plan around the five pillars, all of which need to be assessed and updated accordingly.

2.2.3 5 Pillars of Zero-Trust Model

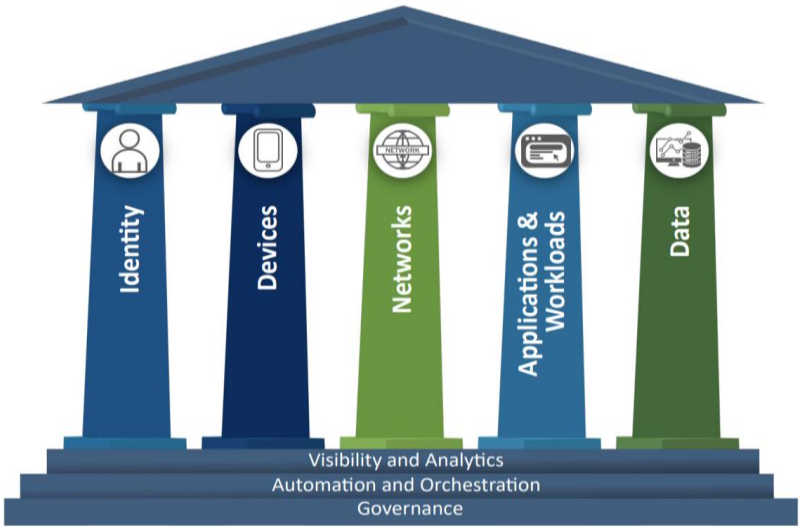

Figure 2 : Zero Trust Model Pillars (Source: CISA)

a. **Identity:** Digital identity refers to the collection of data that represents an entity online. With zero trust, organizations need to verify the digital identity of all users, devices and applications before granting them access to their network. When a human or non-human entity tries to access an organization's network, the organization must use strong risk-based authentication (adaptive access control) and behavioral analysis to verify the digital identity of that user. They should use real-time detection, automated remediation and connected intelligence solutions to monitor and respond to abnormal behavior.

b. **Device:** Zero trust requires organizations to check devices before granting them access to their network of resources. Organizations need to have a complete inventory of managed and unmanaged assets being used within their network and ensure they function properly. They need to check the health and compliance of these devices to ensure they are secure and only running approved programs.

c. **Network:** An organization's network needs to be segmented to prevent threat actors from moving laterally and accessing sensitive data if a user is breached. Network segmentation isolates parts of an organization's network to control access to sensitive data. These segments are determined based on the type of sensitive data and the users who need access to them. Segmenting the network limits users' access to only the resources they need to do their jobs. Users cannot access any other part of the network that is not within their segment.

d. **Application and Workload:** Applications need to access sensitive data and systems to do their jobs, but their access should be as limited as possible, just as access for human users should be. They also need to be regularly audited to ensure they have the latest security updates to prevent cybercriminals from exploiting security vulnerabilities and gaining unauthorized access to the organization's network. Organizations need to continuously monitor

applications and enforce runtime security policies to prevent unauthorized access.

e. **Data:** Organizations need to identify and classify their data based on how valuable it is. They need to limit access to sensitive data and only allow privileges to users who absolutely need access to do their jobs. All data should be encrypted both in rest and in transit to prevent unauthorized users from reading it. Organizations should also follow least privilege access principles when managing access to data.

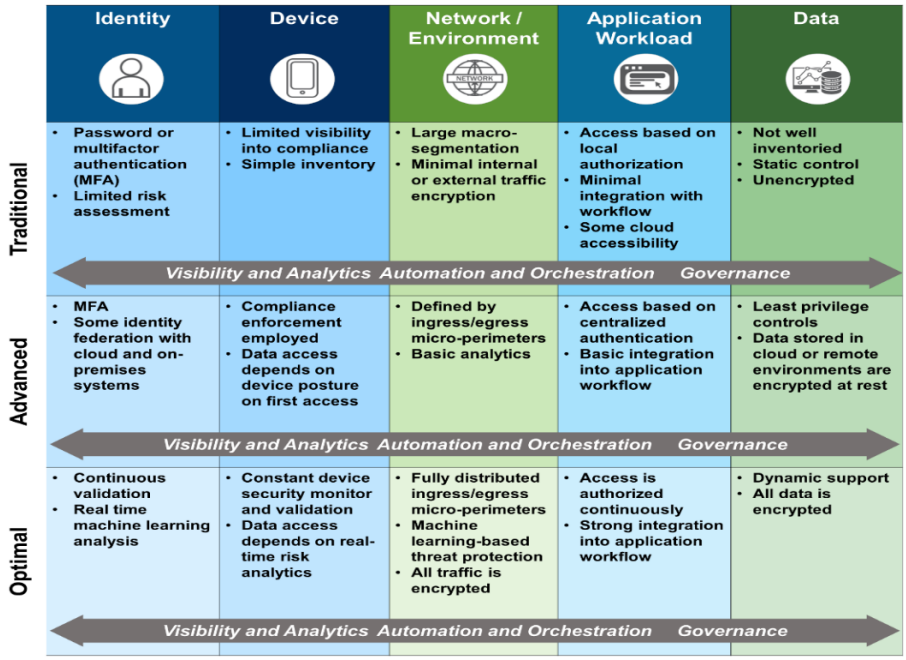

Figure 3: High-Level Zero Trust Maturity Model (Source: CISA)

(Reference: CISA Zero Trust Maturity Model: https://www.cisa.gov/sites/default/files/2023-04/zero_trust_maturity_model_v2_508.pdf)

2.3 IDENTITY AND ACCESS MANAGEMENT (IAM)

Identity Management:

Centralized Identity Providers

Centralized Identity Providers manage user identities and credentials in a unified system, acting as a central repository for authentication and access control. Examples include Active Directory (AD) and Lightweight Directory Access Protocol (LDAP) systems.

1. **Active Directory (AD)**
    a. A widely used directory service by Microsoft, AD provides centralized management of users, groups, and devices within an organization's network.
    b. It supports features such as single sign-on (SSO), group policies, and multifactor authentication (MFA).
    c. AD enables IT administrators to enforce security policies consistently, manage permissions, and monitor user activity from a single interface.

2. **Lightweight Directory Access Protocol (LDAP)**
    a. LDAP is an open-standard protocol for accessing and managing directory information, commonly used in Unix/Linux environments.
    b. It allows applications and systems to authenticate users and retrieve user details from a centralized database.
    c. LDAP is highly flexible, enabling integration with various platforms and support for multiple authentication methods.

Federated Identity

Federated Identity Systems extend identity management by allowing users to access multiple systems or services using a single set of credentials across organizational or platform boundaries. This is achieved through protocols such as Single Sign-On (SSO) and OAuth.

1. **Single Sign-On (SSO)**
    a. SSO enables users to authenticate once and gain access to multiple connected systems without needing to log in separately for each system.
    b. It uses authentication tokens to verify user identities across applications and services.
    c. Common implementations include Microsoft Azure AD SSO and Google Workspace.

2. **Open Authorization (OAuth)**
    a. OAuth is an open standard for delegated access, allowing users to authenticate through a trusted third-party provider (e.g., Google, Facebook) to access applications without sharing credentials.
    b. It is widely used in web and mobile applications to enable seamless integration and secure access.

Privilege Management

Privilege Management is a critical component of access control strategies, designed to ensure that users and systems operate with only the permissions necessary to perform their tasks. By limiting access to sensitive resources and functions, privilege management reduces the attack surface and minimizes the impact of potential security breaches. Two essential concepts within privilege management are the Principle of Least Privilege (PoLP) and Just-in-Time (JIT) access for administrators, both of which contribute to a secure and efficient access control framework.

3 Principle of Least Privilege (PoLP)

The Principle of Least Privilege (PoLP) restricts users and systems to the minimum access necessary to perform their tasks, ensuring permissions align strictly with job responsibilities. This approach minimizes the attack surface, reduces potential damage from compromised accounts. To implement PoLP, organizations can define roles via Role-Based Access Control (RBAC), conduct periodic privilege audits, and use automation tools to detect and mitigate privilege creep. PoLP forms the foundation of a secure access management framework, limiting unauthorized activities and improving overall

security posture.

4 Just-in-Time (JIT) Access for Administrators

Just-in-Time (JIT) Access grants temporary elevated privileges to administrators, ensuring permissions are available only for specific tasks and revoked immediately after. For example, a system administrator might receive access to deploy a patch but lose it upon task completion. This approach minimizes privilege exposure, enhances accountability by requiring explicit approvals and logging all actions, and prevents dormant administrative accounts from becoming attack vectors. By tying access to defined tasks with strict expiration times.

Access Review and Policies

Access review and policies are critical components of a robust access management framework, ensuring that user access aligns with organizational roles, compliance requirements, and evolving security threats. They help maintain the Principle of Least Privilege (PoLP) by regularly reviewing permissions, securely deprovisioning accounts during employee offboarding, and implementing context-aware restrictions such as time-based and location-based access controls.

1. **Periodic Role and Privilege Reviews**

Periodic reviews of user roles and privileges are essential for identifying and addressing excessive or outdated permissions. Over time, users may accumulate unnecessary privileges, leading to privilege creeping, which increases security risks. Regular audits ensure that access remains aligned with job responsibilities and sensitive resources are limited to authorized personnel. Key steps include comparing permissions against documented roles, removing redundant privileges, and leveraging automation tools to detect anomalies. These reviews support regulatory compliance, reduce the attack surface, and enhance overall security.

2. **Deprovisioning Accounts (Employee Offboarding)**

Secure and timely deprovisioning of user accounts during employee offboarding is critical to preventing unauthorized access. Dormant accounts left active after an employee departs can be exploited by attackers or insiders. Effective deprovisioning involves disabling or deleting accounts across systems, revoking shared credentials, and removing access to collaboration tools. Automated workflows integrated with HR systems streamline the process, ensuring no accounts are overlooked. Proper deprovisioning not only strengthens security but also ensures compliance with data protection regulations.

3. **Time-Based and Location-Based Restrictions**

Time-based and location-based restrictions enhance security by applying contextual access controls. Time-based restrictions limit access to specific hours, such as business hours for non-administrative users or temporary access for contractors. Location-based restrictions allow access only from approved geographic regions or IP ranges, blocking logins from high-risk locations and requiring VPNs for remote access. These policies reduce the risk of unauthorized access while ensuring legitimate users operate

within secure parameters, protecting organizational resources from compromised credentials and remote attacks.

2.4 PASSWORD AND AUTHENTICATION POLICIES

2.4.1 Password Complexity and Rotation Policies

A strong password policy is fundamental to secure authentication. Organizations should enforce complexity requirements, ensuring passwords contain a mix of uppercase and lowercase letters, numbers, and special characters. For example, a password like P@ssw0rd123! is significantly more secure than simple or sequential ones like 123456. Passwords must also have a minimum length of 15 characters to resist brute force and dictionary attacks. Additionally, common words or phrases and personal information, such as names, birthdays, or addresses, should be prohibited to prevent easy guessing or cracking.

Regular password rotation is essential to minimize the risk of long-term exposure. Password expiration policies should mandate users to update their credentials periodically, typically every 45 to 90 days, depending on organizational requirements. To prevent reuse, systems should maintain a history log and restrict users from reusing previously used passwords, ensuring unique and secure credentials with every reset.

2.4.2 Account Lockout and Recovery Policies

Account lockout policies provide an effective defence against brute force and credential stuffing attacks. These policies temporarily lock an account after a predefined number of failed login attempts, typically five. A cooldown period or manual intervention by an administrator is then required to unlock the account, significantly slowing down automated attack attempts. Notifications of repeated failed login attempts should be sent to administrators to enable proactive threat monitoring and response.

Password recovery mechanisms must also be secure and user-friendly. Recovery options should be self-service password reset (SSPR) such as email or SMS-based recovery codes, which are time-sensitive to prevent misuse. Additional identity verification steps, such as answering security questions or using multi-step authentication, should be implemented to safeguard against unauthorized password resets.

2.4.3 Multi-Factor Authentication (MFA)

MFA operates on the principle that no single authentication factor should be solely relied upon to verify a user's identity. Instead, it leverages a combination of factors to ensure that only the intended user can gain access. These factors are generally categorized into five distinct types:

- Something You Know (Knowledge Factor): The password, PIN, Passphrases, security questions.
- Something You Have (Possession Factor): A one-time passcode (OTP) sent via SMS, email, or generated by an authenticator app, or a physical hardware token like YubiKey, smartcards.
- Something You Are (Inherence Factor): Biometric identifiers, such as fingerprints or facial recognition, Iris scan.
- Somewhere You Are (Location Factor): - Use IP address, GPS, WIFI, or network location to verify the user’s location.

MFA significantly reduces the risk of unauthorized access, as attackers would need to compromise multiple factors simultaneously. It is especially critical for protecting privileged accounts, sensitive data, and systems with high-security requirements.

Category	Policy Area	Details
Servers	Password Complexity	- Minimum of 12-15 characters. - Include uppercase, lowercase letters, numbers, and special characters.
	Rotation Policy	- Change passwords every 90 days. - Use unique passwords for each server to prevent lateral movement.
	Access Restrictions	- Allow administrative access to authorized personnel only. - Disable or delete default accounts immediately post-deployment.
	Implementation	- Enforce policies using OS-level security configurations (e.g., Group Policy in Windows or PAM in Linux). - Use role-based accounts (e.g., `webadmin`, `dbadmin`) with no shared credentials.
Endpoints	Password Guidelines	- Minimum length of 12-15 characters. - Biannual updates for non-admin users; quarterly for admin users.
	Enhanced Protection	- Integrate Multi-Factor Authentication (MFA). - Enforce a history of the last 5 passwords to prevent reuse.
	Device Lockout	- Implement account lockout after 5 failed attempts. - Configure auto-lock after 10 minutes of inactivity.
	Central Management	- Enforce through Endpoint Detection and Response (EDR) and Mobile Device Management (MDM) tools.
Switches and Routers	Admin Account Configuration	- Remove default credentials immediately. - Assign unique, complex passwords for admin access.
	Privileged Access	- Use local accounts for backup but prefer centralized authentication (e.g., RADIUS or TACACS+).
	Session Management	- Set an idle timeout for management sessions (e.g., 10 minutes). - Enable logging of all administrative commands.
	Periodic Reviews	- Review passwords quarterly and update as needed. - Store credentials securely using a password manager or vault.
Email	Account Password Policy	- Require passwords to be a minimum of 12-15 characters for sensitive accounts. - Enforce MFA for all email accounts.
	Security Features	- Monitor for anomalies (e.g., logins from unusual IPs or locations). - Use OAuth or token-based authentication for email client access.
	Breach Protection	- Implement password reset policies following suspicious activity detection or breaches.

Category	Policy Area	Details
Cloud	Account Access	- Use Identity Federation for centralized management (e.g., SSO via Azure AD or Okta). - Enforce a minimum of 12-15 characters for root/admin accounts.
	MFA Requirements	- Mandatory MFA for all privileged accounts. - Prefer hardware tokens or app-based authenticators over SMS-based MFA.
	Audit and Review	- Conduct periodic access reviews to remove inactive accounts and rotate credentials.
	Access Keys	- Avoid embedding access keys in code. - Use environment variables or secret management tools.

Table 1: Password policies and best practices

Table 1: Password policies and best practices, outlines the password policies and best practices for Servers, Endpoints, Switches, Routers, Email, and Cloud systems.

2.5 LOGGING, MONITORING, AND AUDIT FRAMEWORK

Logging, monitoring, and audit frameworks are critical for maintaining security, compliance, and operational efficiency in modern IT environments. By centralizing logging, implementing retention and real-time monitoring policies, and adhering to audit guidelines, organizations can gain actionable insights into their systems, identify potential threats, and meet regulatory requirements.

2.5.1 Centralized Logging

Centralized logging is a practice of collecting, storing and analysing consolidated logs from various systems, applications, and devices into a unified platform for easier management. Tools such as Splunk and Graylog are commonly used to aggregate and parse log data from multiple sources, providing a comprehensive view of system activity. Integration of centralized logging platforms across systems ensures seamless data collection and enables organizations to detect anomalies and troubleshoot issues quickly. Centralized logging also facilitates event correlation, where data from disparate systems is linked to provide context for security incidents or operational failures. By centralizing logs, organizations can streamline their monitoring processes, reduce manual overhead, and improve response times to critical events.

2.5.2 File Integrity Monitoring (FIM)

File Integrity Monitoring (FIM) is a security control that detects unauthorized modifications to critical system file, configurations, logs, and sensitive data. It helps organizations identify potential security breaches, policy violations, or accidental changes that may impact system stability and compliance.

Importance of FIM:

1. Detects Unauthorized Changes: Identifies unexpected modifications to system files, configurations, and registry settings.
2. Compliance Requirement: Many regulatory frameworks, such as NIST SP 800-53, PCI DSS, ISO 27001, and CIS Controls, mandate FIM as part of security best practices.
3. Incident Response Support: Provides real-time alerts and audit logs to assist in forensic investigations.

4. Protection Against Malware and Insider Threats: Detects attempts to modify files due to malware infections, ransomware attacks, or unauthorized insider activity.

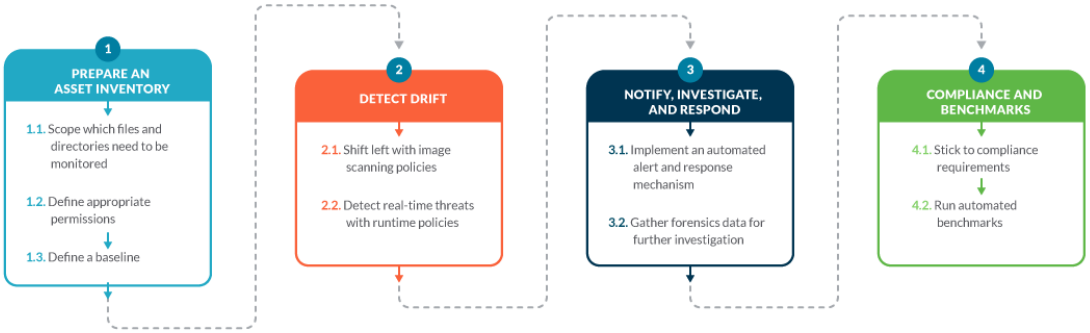

Figure 4: FIM Implementation (Source: Sysdig)

2.5.3 Retention and Real-Time Monitoring

Log retention policies ensure that logs are stored securely for a predefined period to meet compliance requirements and enable historical analysis. Compliance with the requirements protects organizations from legal penalties and provides evidence in case of a breach investigation. Real-time monitoring complements retention by continuously analysing logs for suspicious activities, such as unauthorized access attempts or unusual traffic patterns. Event correlation tools help detect and flag threats by connecting seemingly isolated events. For example, a series of failed login attempts followed by a successful login from an unusual IP address can trigger an alert. Real-time threat detection and automated alerts enable organizations to respond swiftly, mitigating potential damage.

2.5.4 Audit Guidelines

Audit guidelines establish processes for regular log reviews and forensic investigations to maintain system integrity and compliance. Regular log reviews help identify anomalies, detect insider threats, and ensure that systems are operating as expected. These reviews should focus on critical areas such as user activity, configuration changes, and failed access attempts. Forensic investigations use historical log data to analyse the root cause of security incidents, providing a detailed timeline of events. Logs must be tamper-proof and stored in a secure repository to maintain their integrity as evidence. Adhering to audit guidelines not only strengthens security but also demonstrates due diligence during external audits or legal inquiries.

2.6 BACKUP, RETENTION AND DISASTER RECOVERY

Backup and disaster recovery strategies are essential for ensuring data availability and system continuity in case of disruptions. A robust framework mitigates the impact of data loss, hardware failures or security incidents by providing mechanisms to recover operations efficiently. Key components of such a framework include backup strategies, disaster recovery objectives, testing and redundancy, and scenario-specific recovery plans.

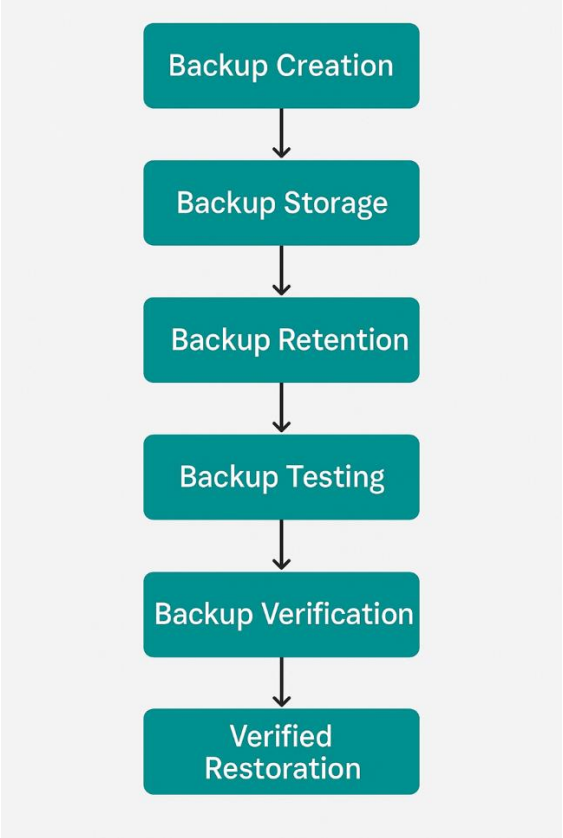

Figure 5: Backup Lifecycle

Backup Strategies

A comprehensive backup strategy combines different types of backups to ensure data can be restored effectively. Full backups involve creating a complete copy of all data at a given point in time. While these are straightforward to restore, they require significant time and storage space, making them suitable for periodic use. Incremental backups address these challenges by capturing only the data that has changed since the last backup whether full or incremental. This method is efficient in terms of storage and time but requires multiple files for restoration, increasing complexity. Differential backups on the other hand save all changes since the last full backup, balancing restoration speed and storage requirements. Combining these methods allows for flexible, efficient, and reliable data protection tailored to the organization's needs.

Backup Retention

Establishing a robust backup retention policy is crucial for balancing storage costs with recovery needs. Best practices include adhering to legal and regulatory compliance requirements, which often dictate minimum retention periods for specific data types. Implementing a tiered approach, such as the grandfather-father-son (GFS) method, allows for keeping recent backups readily available for quick restores while retaining older backups for long-term archiving or compliance. Regularly reviewing and adjusting the retention policy based on evolving business needs and data value ensures efficient storage management and adequate data protection. This involves defining retention periods for different backup types (daily, weekly, monthly, yearly) and securely disposing of backups once they reach their expiration date.

Disaster Recovery Objectives

Disaster recovery objectives define the acceptable thresholds for data loss and downtime during disruption. Recovery Time Objective (RTO) specifies the maximum allowable downtime before systems must be restored, ensuring minimal impact on operations. For example, systems critical to daily activities may have a very short RTO. Recovery Point Objective (RPO), on the other hand determines the maximum amount of data that can be lost measured in time. A lower RPO signifies that data backups need to be frequent to minimize loss. By setting clear RTO and RPO metrics, organizations can prioritize resources and recovery strategies, ensuring that essential systems are restored promptly, and data loss is minimized.

Restoration Planning, Testing and Redundancy

Regular testing and redundancy are vital to maintaining confidence in backup and recovery systems. Data replication ensures that data is copied to secondary locations, either on-premises or in the cloud, providing immediate access during disruptions. Cloud replication offers the added benefit of geographic diversity, safeguarding against local disasters. Periodic restoration testing validates that backups can be successfully restored when needed, identifying and resolving potential issues before they impact recovery. Simulations of various recovery scenarios, such as partial data restoration or full system rebuilds, ensure preparedness and reliability in real-world incidents.

Restoration must be treated as a critical business continuity process. Backups alone are not sufficient; regular validation and testing are essential. Organizations must define and meet Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) for all critical zones.

Element	Details / Guidelines
Restoration Types	Full Restore, Granular Restore, Bare-Metal Restore
Testing Frequency	Quarterly (Critical), Bi-annually (Non-Critical), Monthly (DIP Simulations)
Test Types	Planned recovery, failure simulations, point-in-time validations
Validation Checks	Checksum verification, app functionality, access revalidation
Access Control	MFA and approval workflow; full logging to SIEM
Documentation	SOPs, logs, drill records, and updated playbooks
Immutable Restore Points	Defined restore intervals (e.g., 7/30 days); no alteration allowed

Table 2: Backup Restoration Best Practices

This section outlines the restoration lifecycle that complements the backup policy. It focuses on ensuring data recoverability, integrity, and operational continuity during disruptions.

Scenario-Based Recovery Planning

Scenario-based recovery planning focuses on addressing specific threats to ensure a swift and effective response. In cases of ransomware, recovery plans emphasize isolating affected systems, restoring data from secure backups, and verifying the integrity of the restored environment. For natural disasters, plans prioritize geographic redundancy and include detailed response procedures for affected facilities. In the event of cyberattacks, recovery efforts are tailored to mitigate unauthorized access, restore compromised systems, and strengthen defences based on forensic analysis. By developing playbooks

for specific scenarios, organizations can respond efficiently, reducing downtime and protecting critical data.

Best Practices for Secure Data Backup and Storage

To ensure data integrity, availability, and resilience against failures, cyber threats, and operational disruptions, organizations must implement robust backup strategies. Proper data backup planning minimizes the risk of data loss due to hardware failures, cyberattacks (such as ransomware), accidental deletions, or natural disasters. The following best practices should be adhered to when implementing backup policies:

1) Backup Storage on Independent Machines and Locations- To prevent single points of failure, backups must never be stored on the same physical or virtual machine as the primary (production) data. Instead, organizations should:

    a) Ensure that all backups are stored on separate, dedicated backup servers or storage devices that are not directly linked to the production environment.
    b) Implement offsite or cloud-based backups as part of a comprehensive disaster recovery strategy to ensure data availability in case of a site-wide failure or cyberattack.
    c) Use network segmentation to isolate backup environments from production systems, reducing the risk of lateral movement in case of a security breach.
    d) For virtual environments, avoid storing backups on the same hypervisor or storage pool as the primary system, as this increases the risk of data loss in case of host failures.

2) Backup Redundancy and Versioning- To ensure high availability and rapid recovery, organizations should implement a multi-layered backup redundancy strategy, such as the 3-2-1 backup rule, which mandates:

    a) Maintaining at least three copies of critical data.
    b) Storing backups on two different types of media (e.g., local disk storage and network-attached storage).
    c) Keeping one backup copy offsite or in the cloud to mitigate physical disasters such as fire, flooding, or theft.
    d) Versioning should be enabled to retain multiple historical copies of critical data, allowing for restoration to a previous state in case of corruption, accidental deletion, or ransomware encryption.

3) Automated Backup Scheduling and Regular Validation- To enhance reliability and ensure operational continuity, organizations must:

    a) Automate backup processes with scheduled full, incremental, or differential backups based on business requirements.
    b) Define retention policies for backups, ensuring that obsolete backups are periodically purged to optimize storage capacity.
    c) Conduct regular validation and testing of backup restoration procedures to ensure that backups can be recovered within defined Recovery Time Objectives (RTOs).

d) Implement real-time or near real-time replication for mission-critical systems to minimize Recovery Point Objectives (RPOs).

4) Security Measures for Backup Data- As backup data often contains sensitive and business-critical information, it must be secured against unauthorized access, modification, and exfiltration. To achieve this, organizations should:

a) Encrypt backup data both in transit and at rest using industry-standard encryption algorithms (e.g., AES-256).
b) Implement strict access controls and multi-factor authentication (MFA) to restrict unauthorized personnel from accessing or modifying backup data.
c) Ensure immutable backups (write-once, read-many storage) for critical systems to prevent tampering by ransomware or insider threats.
d) Maintain detailed logging and monitoring of backup processes to detect anomalies and unauthorized access attempts.

Secure Backup Policy

This table defines backup frequency, retention, and security configurations based on environment criticality and data type. It helps ensure a standardized, risk-aware backup strategy across all IT zones.

Zone	Data Type	Backup Frequency	Storage Type	Retention Policy	Security Controls
Production (DIP)	Core DBs, financial records	Hourly Incremental + Daily Full	Encrypted Offsite (Immutable + Cloud)	7 Years / Regulatory Retention	AES-256, MFA, Access Logging, Immutable Backups
Internal Zone	Application configs, AD, logs	Daily Full	Redundant Disk + Offsite Cloud	90 Days	Encrypted Transit/At Rest, Vaulted Credentials, Change Detection
DMZ Zone	Webserver content, public DNS data	Daily	Local Snapshot + Cloud Sync	30 Days	Tokenized Secrets, IP-restricted Restore Access
Dev/Test Zone	Test data, ephemeral builds	Weekly	Cloud-Only	14 Days	No production data, isolated backup environments
Endpoint Zone	Workstation profiles, local files	Weekly Differential	Centralized Network Share	60 Days	Disk encryption, restricted restore, antivirus-integrated backup
Management Zone	Logs, SIEM configs, scripts	Daily	Local + Offsite Redundancy	180 Days	Signed Backups, Immutable Snapshots, Access Review Logs
Guest/IoT Zone	Logs (if any), minimal local data	Optional (Monthly or None)	Edge Snapshot	15 Days	Isolated backups, restore only after manual review

Table 3: Zone-Specific Secure Backup Policy

2.7 ENCRYPTION AND DATA SECURITY

Encryption and data security are foundational to protecting sensitive information from unauthorized access or exposure. By employing robust encryption standards, effective key management practices, and comprehensive data classification and protection mechanisms, organizations can safeguard data at all stages of its lifecycle. These measures collectively reduce the risk of data breaches and ensure compliance with regulatory and operational standards.

Encryption at Rest

- Secures stored data on devices, servers, and backups.
- File-level encryption protects specific files; disk-level encryption secures entire storage systems.
- Use robust encryption algorithms, such as AES-256, for strong security.
- Configure automatic encryption for new data.
- Test recovery including decryption procedures to ensure data accessibility during incidents.

Encryption in Transit

- Protects data moving across networks from interception.
- Use protocols like TLS 1.3 or latest (Transport Layer Security) and HTTPS for secure communications (e.g., emails, web browsing, APIs).
- VPNs provide secure channels for remote connections.
- Regularly renew and validate TLS/HTTPS certificates.
- Disable outdated protocols to protect against evolving threats.

Key Management

Key management is the process of generating, storing, distributing, and retiring cryptographic keys used for encryption, authentication, and digital signatures. Proper key management ensures data confidentiality, integrity, and compliance with security standards such as NIST SP 800-57, ISO 27001, PCI DSS, and CIS Controls.

Principles of Secure Key Management

1. **Confidentiality** – Keys must be protected from unauthorized access.
2. **Integrity** – Keys should not be altered or tampered with.
3. **Availability** – Authorized users should have reliable access to keys.
4. **Lifecycle Management** – Keys should be securely generated, rotated, and retired.

Key Storage Mechanisms

Keys must be securely stored to prevent unauthorized access or theft. There are several industry-standard storage solutions:

1. Hardware Security Module (HSM)

a) HSMs are tamper-resistant, dedicated hardware devices designed to generate, store, and manage cryptographic keys securely.

b) They provide physical security, ensuring keys are never exposed to unauthorized users or software.
c) Used for digital signing, encryption, and certificate management in enterprises.
d) Compliance with FIPS 140-2/3 Level 3 or higher is recommended for HSMs.

2. Trusted Platform Module (TPM)

a) TPM is a specialized chip embedded in hardware (e.g., laptops, servers) that securely stores cryptographic keys.
b) Used for device authentication, disk encryption (BitLocker, LUKS), and secure boot mechanisms.
c) Provides hardware-based attestation, ensuring system integrity by verifying firmware and OS authenticity.
d) TPM can be used to store digital certificates, encrypt credentials, and prevent key extraction even if the device is compromised.
e) Compliant with TPM 2.0 standards for modern security applications.

3. Cloud Key Management Services (Cloud KMS)

a) Cloud providers like AWS KMS, Azure Key Vault, and Google Cloud KMS offer centralized key management services.
b) Provides automated key rotation, access control policies, and auditing.
c) Integrated with cloud workloads to secure storage encryption (S3, Blob), database encryption, and TLS certificates.

4. Secure Software-Based Key Storage

a) Software-based key storage solutions, like Keystores, HashiCorp Vault, and OpenSSL, are used when hardware security is unavailable.
b) Disk encryption tools (BitLocker, VeraCrypt, eCryptfs) rely on securely stored keys.
c) Risk: Software-based storage is more vulnerable to malware and insider threats.

Key Generation and Rotation

1. Keys must be generated using strong entropy sources (e.g., HSMs, TPM, OpenSSL).
2. Key rotation policies should be enforced to limit exposure in case of compromise.
3. Symmetric keys: Rotate every 1-2 years or after a breach.
4. Asymmetric keys (RSA, ECC): Rotate every 2-5 years, depending on usage.
5. API & SSH Keys: Rotate every 90-180 days.

Secure Key Disposal

1. Key destruction must be irreversible using secure deletion methods.
2. HSMs and TPMs provide built-in key wiping functions.
3. Cryptographic wiping techniques (e.g., NIST 800-88 guidelines) should be followed for software-based keys.

Best Practices for Key Management

1. Use dedicated hardware (HSM, TPM) for key storage whenever possible.

2. Implement automated key rotation and expiration policies.
3. Restrict key access using RBAC and enforce MFA.
4. Enable logging and monitoring for key access activities.
5. Encrypt keys at rest and in transit to prevent unauthorized access.
6. Regularly audit key usage and compliance with industry standards.

Key Management Tools

Tool Name	Type	Features	Use Case	Compliance Support
AWS KMS	Cloud-native (AWS)	Automatic key rotation, envelope encryption, IAM policies	Securing S3, RDS, Lambda, API Gateway	PCI DSS, ISO 27001, FedRAMP
Azure Key Vault	Cloud-native (Azure)	HSM-backed keys, audit logging, RBAC via Azure AD	Securing Azure VMs, databases, app secrets	ISO 27001, GDPR, HIPAA
Google Cloud KMS	Cloud-native (GCP)	IAM integration, per-project separation, symmetric/asymmetric support	Encrypting GCP storage, BigQuery, secrets	SOC 2, ISO 27001, PCI DSS
HashiCorp Vault	Open-source/self-hosted	Dynamic secrets, policy-based access control, API driven	Multi-cloud/hybrid environments, secret leasing	Customizable compliance (via logging)
Thales CipherTrust	Hardware/Cloud HSM	FIPS 140-2 Level 3, centralized policy enforcement, tokenization	Highly regulated sectors with on-prem HSM needs	FIPS 140-2, GDPR, PCI DSS

Table 4: Common Key Management Tools and their capabilities

Data Classification and Data Loss Prevention (DLP):

Data classification organizes information based on its sensitivity, enabling appropriate security measures to be applied.

Common classification levels include **Public**, **Internal**, **Confidential**, and **Restricted** data.

Classification ensures that sensitive information is properly labelled and handled. Tools like metadata tagging and automation can simplify this process, while employee training reinforces adherence to classification protocols.

Figure 6: Data Classification Examples (Source: Klassify)

Data Loss Prevention (DLP) technologies complement classification by monitoring and controlling data movement to prevent unauthorized sharing. Endpoint DLP tools manage data transfers to devices like USB drives or cloud platforms, while email DLP solutions block or encrypt sensitive messages. Organizations should regularly update DLP policies to address emerging risks and compliance requirements, ensuring that protection measures remain effective against data breaches.

2.8 VULNERABILITY, COMPLIANCE MANAGEMENT AND GOVERNANCE

2.8.1 Compliance Management

The Enterprise Compliance Management Program is a comprehensive framework designed to ensure regulatory adherence and implement security best practices. Aligned with ISO/IEC 27001 and CIS Benchmarks, it manages compliance requirements, internal controls and risk assessments systematically. Leveraging Information Security Management Systems (ISMS), it promotes risk management, continuous improvement and security awareness. The program covers physical and digital assets, human resources, and third-party relationships, mapping regulatory obligations to business processes and technical controls. By following CIS Benchmarks security baselines, it enforces secure configurations for systems and services, ensuring alignment with standards & frameworks like ISO 27001, NIST.

1. Secure Configuration Baselines

Secure configuration baselines serve as the foundation for hardening systems across various platforms:

- **Endpoints**: Windows endpoints (Windows 10 & 11) adhere to CIS guidelines for secure desktop configurations.
- **Servers**: Operating systems, including Windows Servers (2022, 2019, 2016) and Linux servers (RHEL, CentOS, Ubuntu etc.), are hardened with regularly updated baselines.
- **Network Devices**: Firewalls, routers, and switches are configured using CIS Level 1 benchmarks to address critical vulnerabilities while supporting operational requirements.
- **Cloud Services**: Cloud workloads leverage CIS benchmarks & OEM security baselines for secure virtual machines and container configurations across major cloud platforms.
- **Email Systems**: Secure configurations are applied to email servers (e.g., Exchange, SMTP) to mitigate phishing risks and prevent unauthorized access.

Baselines are reviewed and updated annually to reflect changes in CIS guidelines & OEM security baselines, ensuring ongoing relevance and effectiveness.

2. Secure Configuration Assessment

Secure configuration assessments are conducted both manually and using automation to identify and address potential misconfigurations:

- **Manual Reviews**: Periodic assessments of all IT assets, including operating systems, middleware, databases, web servers, web & mobile applications, storage devices, network devices, endpoints, and cloud environments, ensure adherence to baselines.
- **Automated Scanning**: Tools such as vulnerability scanners and configuration management solutions provide automated detection of non-compliance points across the infrastructure.

3. Remediation and Reporting

- **Remediation**: Identified non-compliance points are documented and shared with relevant stakeholders for remediation. This process prioritizes critical vulnerabilities and ensures timely resolution.
- **Re-Validation and Reporting**: Post-remediation, re-validation scans are performed to verify the closure of non-compliance points. Reports are generated to track compliance improvements ensuring transparency and accountability.

2.8.2 Vulnerability Management

The Enterprise Vulnerability Management Program forms a critical foundation of proactive cybersecurity ensuring a structured and effective approach to identifying, assessing, and remediating security vulnerabilities. Aligned with NIST guidelines and industry best practices, the program addresses security risks holistically, encompassing all organizational assets from critical servers to end-user devices. This living framework evolves continually to incorporate lessons learned, industry advancements and emerging threats, ensuring a robust and adaptive security posture.

1. Asset Management and Vulnerability Assessment

Effective asset management is the cornerstone of vulnerability detection and management. The program employs a rigorous monthly scanning schedule, balancing resource allocation with comprehensive security coverage. This consistent approach ensures timely identification of security gaps across the organization's digital landscape.

The assessment process incorporates authenticated scanning, which provides detailed insights into configuration and software vulnerabilities and unauthenticated scanning, which simulates external threats. These scans leverage industry-standard tools configured to detect vulnerabilities effectively, ensuring complete coverage of assets including servers, network devices, endpoints, and cloud resources.

2. Risk Assessment and Prioritization

Risk assessment and prioritization are at the core of the remediation strategy. The program goes beyond traditional CVSS scoring by incorporating multiple factors:

- Asset Criticality: The importance of assets to business operations.
- Data Sensitivity: The type and sensitivity of data processed by the asset.
- Exploit Availability: The presence of known exploits in the wild.
- Threat Intelligence: Current information on active and emerging threats.

This comprehensive calculation enables precise prioritization of vulnerabilities. Critical vulnerabilities (9.0–10.0 on the risk scale) are remediated within a 14-days window, ensuring swift mitigation of the highest risks. High-risk vulnerabilities (7.0–8.9) are addressed within 30 days, while medium (4.0–6.9) and low-risk vulnerabilities (0.1–3.9) follow 60-days and 90-days remediation windows, respectively. This tiered approach ensures operational efficiency while focusing on the most significant risks.

3. Program Metrics and Reporting

The program's effectiveness is monitored using a robust set of metrics and reporting mechanisms ensuring transparency and accountability. Key performance indicators (KPIs) include:

- Mean Time to Remediate (MTTR): Average time to resolve vulnerabilities.
- Vulnerability Aging Statistics: Tracking how long vulnerabilities remain unresolved.
- Scan Coverage Percentages: Ensuring all assets are consistently scanned.
- SLA Compliance Rates: Adherence to remediation timelines.

Regular reporting through executive dashboards and detailed management reports keeps stakeholders informed, supporting compliance documentation requirements. This comprehensive reporting structure also aids in identifying trends, areas for improvement and resource allocation.

4. Continuous Improvement

The program undergoes quarterly assessments and annual audits to ensure its relevance and effectiveness. These evaluations include peer benchmarking and process optimization, enabling the program to adapt to changes in the threat landscape and business priorities. Feedback loops integrate lessons learned from incidents and audits, fostering continuous improvement and innovation.

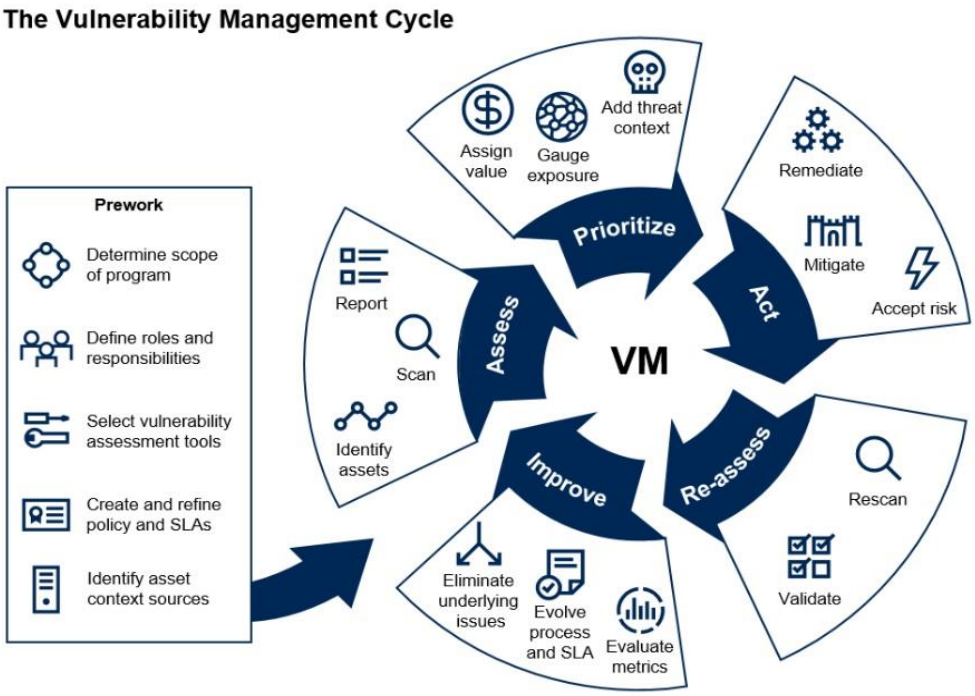

Figure 7: Vulnerability Management Lifecycle (Source: Gartner)

2.8.3 Governance

The Enterprise Governance Framework establishes accountability, transparency, and strategic alignment in managing cybersecurity and compliance requirements. This framework integrates security policies, regulatory compliance mandates, and risk management principles to support organizational objectives while safeguarding digital and physical assets. It ensures that security efforts align with business priorities, legal requirements, and industry standards.

1. Security Governance Structure

Security governance operates through a structured model that defines roles and responsibilities for different stakeholders, ensuring effective oversight and decision-making:

- Executive Oversight: The board and leadership teams set strategic security priorities, allocate resources, and ensure regulatory compliance.
- Security Committees: Dedicated groups composed of compliance officers, IT security teams, and risk management professionals govern cybersecurity initiatives.
- Operational Teams: IT, legal, and security teams execute security controls, monitor threats, and implement risk management strategies.

2. Regulatory Compliance and Policy Management

Enterprise security governance integrates multiple compliance regulations and industry standards:

- Policy Framework: Security policies covering access management, incident response, data protection, and vendor security are established and reviewed periodically.

- Compliance Alignment: Adherence to standards such as ISO 27001, CIS Benchmarks, NIST, GDPR, and local regulatory mandates ensures compliance across business units.
- Audit and Compliance Reviews: Regular audits and assessments verify compliance with cybersecurity policies, ensuring alignment with global standards and regulatory obligations.

3. Risk Governance and Decision-Making

Governance ensures that risk management integrates with organizational decision-making:

- Enterprise Risk Management (ERM): Risk assessments consider financial, operational, technological, and compliance risks affecting the business.
- Risk Treatment Strategies: Identified risks are mitigated through controls, compensatory measures, and cybersecurity enhancements aligned with business goals.
- Third-Party Risk Management: Governance policies cover vendor security assessments, contractual security obligations, and third-party cybersecurity risk evaluation.

4. Continuous Monitoring and Governance Oversight

To maintain cybersecurity posture and compliance integrity, governance ensures ongoing monitoring and improvements:

- Key Performance Indicators (KPIs): Metrics such as compliance rates, audit findings, risk resolution timelines, and threat response times are tracked.
- Security Awareness & Training: Regular training initiatives enhance security awareness among employees, executives, and third parties.
- Governance Evolution: Governance frameworks undergo periodic reviews to accommodate emerging threats, regulatory changes, and evolving business landscapes.

This comprehensive governance framework reinforces security accountability, promotes regulatory adherence, and supports continuous improvement to sustain organizational resilience in an evolving cyber threat environment.

2.9 PATCH AND UPDATE MANAGEMENT

Effective patch and update management is a critical component of maintaining secure and stable IT infrastructure. It ensures systems are protected against vulnerabilities and minimizes the risk of exploitation.

1. **Regular Patch Deployment**
    - Adopt automated patch management tools, such as WSUS, SCCM, or third-party solutions, to streamline the process of deploying updates.
    - Establish a consistent patching schedule to ensure all systems are up to date without disrupting operational continuity.
    - Prioritize zero-day vulnerabilities and critical patches to address known threats promptly.

2. **Testing and Validation**
    - Test patches in a controlled environment (e.g., UAT or staging environments) before deployment to production systems. This minimizes the risk of instability or conflicts.
    - Conduct sanity checks and rollback planning to ensure a seamless update process in case of failures.
3. **Vendor Coordination and End-of-Life Management**
    - Monitor vendor updates for security patches, firmware upgrades, and end-of-life announcements.
    - Replace unsupported hardware and software to maintain a secure and compliant infrastructure.
4. **Compliance and Reporting**
    - Maintain a detailed log of patch deployment activities to demonstrate compliance with regulatory and audit requirements.
    - Revalidate patch deployment and effectiveness using vulnerability assessment tools.
    - Leverage centralized dashboards to track patch status, identify gaps, and mitigate delays.

By incorporating these practices, organizations can maintain secure environments, reduce downtime, and enhance operational resilience.

2.10 DEVICE HEALTH MONITORING

Device health monitoring is an integral aspect of proactive IT management, focusing on identifying and mitigating potential issues before they impact operations.

1. **Real-Time Monitoring Tools**
    a. Use advanced monitoring tools (e.g., SolarWinds, Nagios, or Zabbix) to continuously track the health of critical systems and endpoints.
    b. Monitor key metrics such as CPU utilization, memory usage, disk space, and network performance.
2. **Event Logging and Alerts**
    a. Enable comprehensive logging to capture device activity and detect anomalies.
    b. Configure real-time alerts for threshold breaches or unusual activity allowing IT teams to respond promptly.
3. **Preventive Maintenance**
    a. Schedule routine hardware checks and firmware updates to ensure devices are functioning optimally.
    b. Implement a lifecycle management plan to replace aging hardware and prevent failures.

4. **Integration with Centralized Systems**

    a. Integrate health monitoring with SIEM platforms to correlate device data with broader security insights.

    b. Employ automated remediation workflows to address identified issues without manual intervention.

Proactive device health monitoring not only reduces downtime and improves reliability but also strengthens the organization's overall security posture by identifying potential vulnerabilities in real time.

3. IT-INFRASTRUCTURE RISK ASSESSMENT

3. IT-Infrastructure Risk Assessment

Risk assessment is a critical component of IT infrastructure security, providing organizations with the ability to identify, analyse, and mitigate risks that could compromise system availability, integrity, and confidentiality. This chapter outlines the risk assessment methodologies for key infrastructure components, aligned with NIST Special Publication 800-30 (Guide for Conducting Risk Assessments) and other industry standards.

3.1 RISK ASSESSMENT FRAMEWORK

A structured risk assessment framework consists of:

- Asset Identification: Cataloguing IT assets, including hardware, software, and data.
- Risk Identification: Recognizing potential risks such as cyberattacks, insider risks, and system failures.
- Vulnerability Analysis: Assessing weaknesses in IT infrastructure.
- Risk Evaluation: Quantifying risks based on likelihood and impact.
- Mitigation Strategies: Implementing controls to reduce identified risks.
- Monitoring & Review: Continuously tracking and improving security posture.

3.2 RISK ANALYSIS

Table 5: Risk Analysis of IT Infrastructure Components outlines key risks associated with different IT infrastructure components, along with mitigation strategies based on NIST guidelines:

Infrastructure Component	Vulnerabilities	Risks	Mitigation Strategies
Network Security	Weak firewall configurations, lack of segmentation, outdated protocols	DDoS attacks, Man-in-the-Middle (MITM), unauthorized access	Deploy next-generation firewalls (NGFW), intrusion detection systems (IDS), zero-trust policies (NIST SP 800-41)
Server Security	Weak authentication, misconfigured permissions	Unpatched OS, privilege escalation, unauthorized access	Implement OS hardening, regular patching, multi-factor authentication (NIST SP 800-123)
Storage Security	Lack of encryption, weak access controls	Data leaks, ransomware, unauthorized access	Use AES-256 encryption, implement role-based access control (RBAC), enforce regular backups (NIST SP 800-88)
Database Security	Poorly secured credentials, unencrypted data	SQL injection, insider risks, data corruption	Enable database encryption, secure authentication, conduct regular audits (NIST SP 800-207)
Endpoint Security	Weak endpoint protection, poor device security policies	Malware, phishing, unauthorized device access	Deploy endpoint protection platforms (EPP), enforce security patching (NIST SP 800-124)

Email Security	Lack of email filtering, weak authentication methods	Phishing, business email compromise (BEC), malware attachments	Implement secure email gateways, use DMARC, deploy MFA (NIST SP 800-177)
Cloud Security	Weak IAM policies, improper security controls	Misconfigured storage, unauthorized API access, insider threats	Enforce IAM best practices, enable cloud security posture management (CSPM), apply least privilege access (NIST SP 800-190)

Table 5: Risk Analysis of IT Infrastructure Components

These are few Risks, Vulnerabilities and its Mitigation strategies related to Network Security, Server, Storage, Database Security, Endpoint Security, Email Security and Cloud Security.

3.3 SECURITY CONTROLS AND COMPLIANCE CONSIDERATIONS

- **Network:** Secure configurations per NIST SP 800-41.
- **Server:** OS hardening following NIST SP 800-123.
- **Storage:** Data protection per NIST SP 800-88.
- **Database:** Security measures aligned with NIST SP 800-207.
- **Endpoint:** Compliance with NIST SP 800-124 for mobile security.
- **Email:** Anti-phishing controls per NIST SP 800-177.
- **Cloud:** Adherence to NIST SP 800-190 for cloud security best practices.

3.4 RISK MITIGATION STRATEGIES & COMPLIANCE CONSIDERATIONS

To ensure comprehensive IT risk management, organizations should:

- Adopt NIST-based risk assessment methodologies (SP 800-30, 800-53, 800-171).
- Implement a continuous risk monitoring framework using SIEM solutions.
- Ensure alignment with regulatory requirements (SEBI, GDPR, ISO 27001).
- Establish a structured Risk Management Framework (RMF) for IT infrastructure.

3.5 RISK MONITORING AND INCIDENT RESPONSE

- **Continuous Monitoring:** SIEM (Security Information & Event Management) systems for real-time monitoring.
- **Threat Intelligence:** Integration with threat feeds for proactive risk detection.
- **Incident Response:** Develop and test incident response plans (IRPs) to ensure quick threat mitigation.
- **Regular Audits:** Conduct periodic risk assessments to adapt to evolving threats.

(References: *NIST SP 800-30: Risk Management Guide for Information Technology Systems*, *NIST SP 800-53: Security and Privacy Controls for Federal Information Systems and Organizations*, *NIST Cybersecurity Framework (CSF) 2.0*)

4. NETWORK INFRASTRUCTURE SECURITY

4. Network Infrastructure Security

4.1 NETWORK ARCHITECTURE AND DESIGN

Network architecture and design play a crucial role in ensuring the security of an organization's digital assets. A well-designed network infrastructure provides a strong foundation for implementing effective security measures, while a poorly designed one can leave vulnerabilities that are ripe for exploitation. A secure network architecture should be built with defence in depth/layer defence in mind, meaning that multiple layers of security are employed at different points throughout the network. This includes securing the perimeter through firewalls and intrusion detection systems, encrypting data both in transit and at rest, implementing access controls based on the principle of least privilege, and regularly patching and updating software and firmware.

4.1.1 Perimeter and Internal Networking Devices

Figure 8: Networking Devices

1. Firewall: A firewall is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Its primary function is to block unauthorized traffic while allowing authorized communications. It acts as a barrier between a trusted internal network and an untrusted external network, such as the internet.

2. Switch: A switch is a networking device that connects devices together on a computer network by using packet switching to receive, process, and forward data to the destination device on a LAN (Local Area Network). Switches operate at the data link layer of the OSI model and are used to create segments or separate broadcast domains in a network. They use MAC addresses to forward data packets between devices.

3. Router: A router is a networking device that forwards data packets along networks. It connects multiple subnetworks and uses routing tables or routing protocols to determine the best path for data to travel from its source to its destination. Routers operate at the network layer of the OSI model and are used to connect different IP networks, such as the Internet and a private LAN.

4. Web Application Firewall (WAF): A web application firewall is a type of firewall specifically designed to secure web applications from attacks. It filters and monitors HTTP traffic to protect against common web exploits, SQL injection, cross-site scripting, and other types of attacks. WAFs can be deployed in front of a web server or application server to provide an additional layer of security.

5. Load Balancer: A load balancer is a networking device that distributes network or application traffic across multiple servers to ensure optimal performance, reliability, and availability. It operates at the application delivery controller level and can distribute traffic based on various algorithms, including round robin, least connections, and weighted distribution. Load balancers help prevent overloading of individual servers and improve overall system response time.

6. Intrusion Detection System (IDS) / Intrusion Prevention System (IPS): An intrusion detection system (IDS) and intrusion prevention system (IPS) are security solutions that monitor network traffic for signs of malicious activity and act when necessary. IDSs analyse traffic patterns and generate alerts when they detect suspicious behaviour-based signatures, while IPSs not only generate alerts but also actively block the attacks. Both systems operate at various layers of the OSI model, with IDS primarily focusing on the network layer and IPS operating at both the network and application layers. Their goal is to identify and mitigate threats before they cause damage to the network or systems.

    1. Installation of Perimeter and Internal Defence Assessment and Planning
        a. Identify Security Needs: Analyse your organization's specific security requirements based on existing threats, compliance regulations, and business operations.
        b. Network Layout: Map out your network architecture to determine where to place perimeter and internal defence devices.

    2. Select Appropriate Devices
        a. Perimeter Défense Devices:
            i. Firewalls: Deploy next-generation firewalls (NGFW) to monitor and filter incoming and outgoing network traffic.
            ii. Intrusion Detection and Prevention Systems (IDPS): Install IDPS to detect and respond to suspicious activities.
            iii. Unified Threat Management (UTM): Consider UTM solutions that combine multiple security features in a single device.
            iv. Web Application Firewalls (WAF): Protect web applications from common threats such as SQL injection and cross-site scripting.
        b. Internal Défense Devices:
            i. Internal Firewalls: Use internal firewalls to segment traffic between different network zones.
            ii. Network Access Control (NAC): Implement NAC solutions to enforce security policies on devices connecting to the network.
            iii. Endpoint Detection and Response (EDR): Deploy EDR solutions on endpoints for continuous monitoring and response to threats.
            iv. Data Loss Prevention (DLP): Use DLP tools to monitor and protect sensitive data within the network.

    3. Installation Process
        a. Prepare the Environment: Ensure that the physical environment is suitable for the devices (e.g., power supply, cooling, space).
        b. Connect Devices:
            i. Perimeter Devices: Connect perimeter devices at the network’s edge, between the internet and your internal network.
            ii. Internal Devices: Position internal devices strategically based on segmentation requirements.

    4. Configuration
        a. Device Settings: Configure each device according to best practices and your security policies. This may include:

i. Defining access control rules.
ii. Setting up logging and monitoring features.
iii. Implementing alerting mechanisms for suspicious activities.
iv. Integration: Ensure devices can communicate with each other and with central management systems (e.g., SIEM).

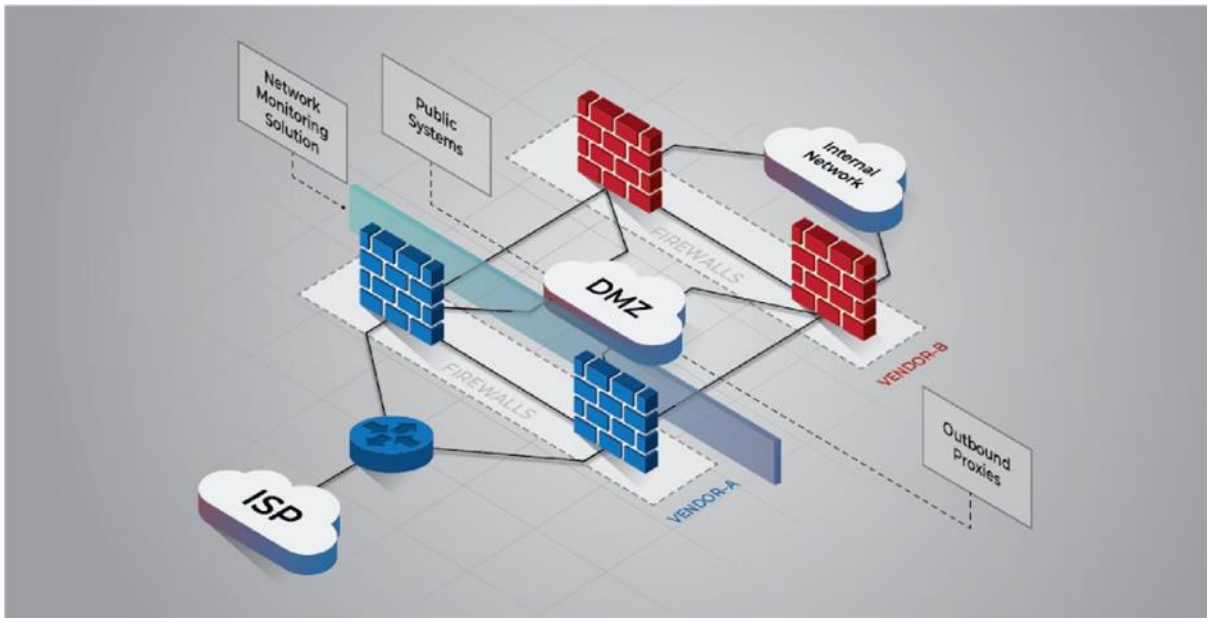

Figure 9: Network Perimeter Defence (Source: Security Planet)

4.1.2 Multi-Tier Network Architecture with DMZ

This architecture is designed to enhance security by isolating public-facing systems from sensitive internal resources. It consists of:

I. ISP (Internet Service Provider): Entry point for external traffic.

II. External Firewall: Filters and protects traffic entering the Demilitarized Zone (DMZ).

III. DMZ: A buffer zone hosting public-facing systems (e.g., web servers) while isolating them from the internal network.

IV. Internal Firewall: Restricts traffic between the DMZ and the Internal Network, which houses critical and sensitive systems.

V. Network Monitoring Solution: Tracks and analyses network activity to detect and respond to threats.

This tiered structure ensures:

- Traffic Segmentation: Limits direct communication between external and internal systems.
- Threat Containment: Compromises in the DMZ do not directly impact the internal network.
- Access Control: Strict firewall rules control data flow between each segment.

By implementing this architecture, organizations achieve improved security, controlled data flow, and minimized attack surfaces.

4.1.3 Group Similar Network Systems

A. Identify Network Systems
   a. Inventory: Create a comprehensive inventory of all network systems, including servers, applications, devices, and services.
   b. Categorization: Identify the primary functions and purposes of each system.

B. Define Grouping Criteria
   a. Functionality: Group systems based on their roles (e.g., web servers, database servers, application servers).
   b. Location: Group systems by physical or logical location (e.g., on-premises, cloud-based, branch offices).
   c. User Base: Consider who uses the systems (e.g., internal employees, external clients, partners).
   d. Security Needs: Group systems by their security requirements (e.g., high-security data systems, low-risk systems).
   e. Compliance Requirements: Group based on regulatory requirements (e.g., PCI DSS).

C. Create Logical Network Segments
   a. IT or Non-IT Segmentation: Use network segmentation to create logical groups that limit access and improve performance. Common segmentation types include:
      i. By Department: Finance, HR, IT, etc.
      ii. By Function: Development, testing, production.
      iii. By Service Type: Web services, database services, application services.
      iv. By Usage:
         1. Physical Barriers: Walls, fences, gates, and other structures that restrict access.
         2. Security Doors: Doors with advanced locking mechanisms, requiring key cards or biometric scanners for entry.
         3. Access Control Systems: Centralized systems that manage access permissions and track entry/exit events.
         4. Surveillance Systems: Cameras, motion detectors, and other devices to monitor activity and deter intruders.

4.1.4 VLANs

Implement Virtual Local Area Networks (VLANs) to logically separate traffic within a physical network. It is recommended to isolate similar systems into different subnets or virtual local area networks (VLANs) or physically separate the different subnets via firewalls or filtering routers. Workstations, servers, printers, telecommunication systems, and other network peripherals should be separate from each other. Operational technology, such as industrial control systems, typically needs to be isolated from other information technology and high-risk networks like the Internet. This physical separation provides stronger protection because the intermediate device between subnets must be compromised for an adversary to bypass access restrictions. Implement access restrictions on the internal routers, switches, or firewalls to allow only those ports and protocols that are required for network operations or valid mission needs. Access control lists (ACLs) may need to be duplicated and applied directly to the switches to restrict access between VLANs, or they can be applied to core routers where routing is performed between internal subnets.

4.1.5 Utilize Strict Perimeter Access Controls

Deploying next-generation firewalls (NGFW) is essential for inspecting traffic and providing advanced filtering and intrusion prevention. Regular updates to firewall rules help ensure only legitimate traffic is allowed. Intrusion detection and prevention systems (IDPS) monitor network traffic for suspicious activities, enabling real-time responses and alerting administrators to unauthorized access attempts.

Network segmentation enhances perimeter control by isolating sensitive areas into distinct zones, limiting access, and reducing the attack surface. Strict access policies in each segment ensure that only authorized users can access critical resources.

Physical security is also crucial; implementing access controls, like cards and biometric scanners, restricts entry to important areas. Surveillance systems like CCTV add another security layer. Regular access reviews and audits are necessary to maintain perimeter control, including promptly terminating accounts of former employees to prevent unauthorized access.

4.1.6 Limit Virtual Private Networks (VPNs)

A VPN tunnel provides an encrypted communication channel between two endpoints and should be used only when confidentiality and integrity cannot be maintained otherwise. VPN gateways are accessible from the Internet and vulnerable to threats, so unnecessary features should be disabled, and strict traffic filtering rules should be enforced. Limit access to UDP ports 500 and 4500, ESP with Authentication (ESP + AH), and known peer IP addresses where possible. If IP addresses are unknown, use an IPS to monitor IPsec traffic.

Establish clear access policies for remote access VPNs, particularly IPsec VPNs, to protect sensitive data. Implement strong authentication methods, such as multi-factor authentication (MFA), to enhance security. Additionally, ensure devices meet security standards (e.g., updated OS, antivirus) before granting access, denying those that do not comply to minimize vulnerabilities.

4.1.7 Implement a Network Access Control (NAC) Solution

Implementing a Network Access Control (NAC) solution is vital for enhancing organizational security by managing access for devices connecting to the network. The process starts with defining clear objectives, such as controlling access for internal users and guests, ensuring compliance, and protecting sensitive data.

Next, organizations must select the right NAC solution that meets their needs, evaluating options like agent-based and agentless technologies based on scalability, compatibility, and ease of integration.

After choosing a solution, a network assessment should be conducted to analyse the current infrastructure, which would help in defining access policies for device connectivity. Tailored access control policies should specify device authentication criteria and authorization levels, balancing security with user accessibility.

Deployment follows, typically in phases, to minimize disruption and allow for controlled testing. Finally, integrating the NAC solution with existing security systems, like firewalls and SIEM tools, enhances overall security and supports coordinated incident responses.

4.1.8 High Availability of all Devices

High availability (HA) is essential for ensuring continuous operation and minimizing downtime in IT systems. It involves designing systems that remain operational during hardware failures, software issues, or other disruptions, allowing uninterrupted access to services.

To achieve HA, organizations often implement redundancy by duplicating critical components, such as servers, power supplies, and network connections. Clustering is one method that allows another server to take over if one goes down.

Load balancing is vital as well, distributing workloads across multiple devices to enhance performance and resilience, preventing any single device from becoming a bottleneck.

Regular monitoring and proactive maintenance are necessary to track device health, enabling IT teams to address potential issues before they lead to failures. Additionally, failover mechanisms automatically switch to a standby device when the primary one fails, ensuring minimal disruption for users.

4.1.9 Network Segmentation Zones and Architecture

To enhance micro-segmentation and enforce least privilege access, it is recommended to establish dedicated security zones. These zones isolate workloads by trust level, function and risk profile. All inter-zone traffic mist be routed through perimeter firewalls or access control filters and audited using centralized logging tools.

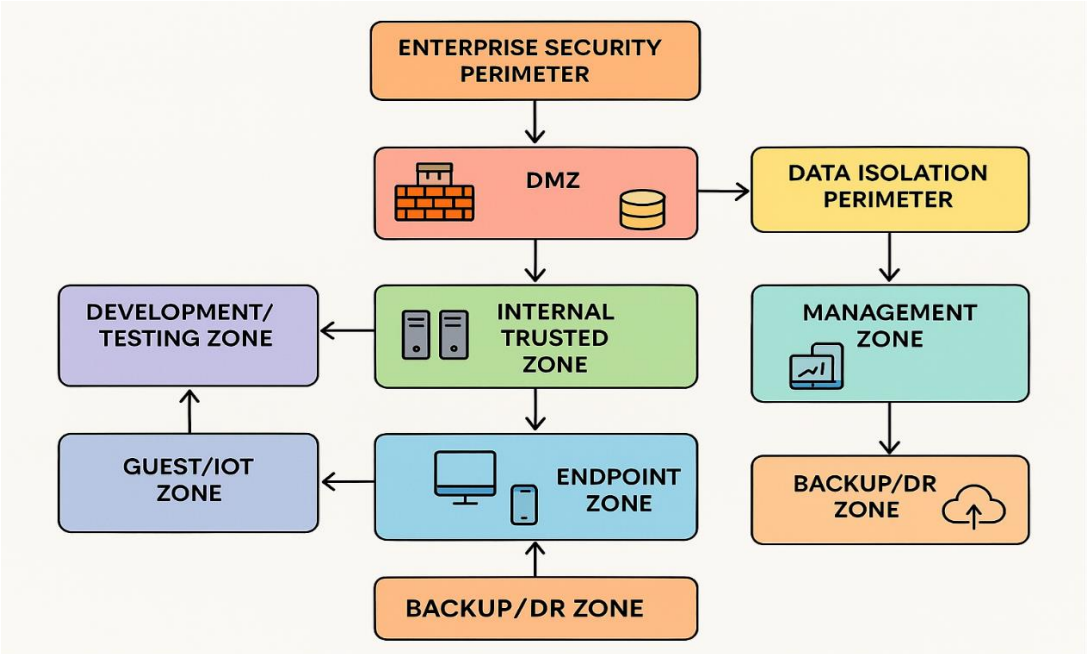

Figure 10: Enterprise Network Segmentation with Security Zones

This diagram represents a layered network segmentation model, dividing an enterprise IT infrastructure into distinct security zones based on function, risk, and trust level. It visually supports the principle of defence-in-depth by limiting lateral movement, isolating sensitive workloads, and enforcing least privilege access.

Purpose of the Layout

- Prevents east-west movement of threats within the network.
- Enforces granular access control policies between zones.
- Helps organizations comply with regulatory standards (e.g., PCI DSS, ISO 27001).
- Enhances incident containment and resilience during breaches.

Zones Segmentation

1. **Enterprise Security Perimeter (ESP)**
   - Acts as the first line of defence.
   - Protects internal systems from internet-borne threats.
   - Hosts external firewalls, intrusion prevention systems (IPS), and VPN gateways.
2. **DMZ (Demilitarized Zone)**
   - Hosts public-facing systems like web and mail servers.
   - Segregated from internal systems to contain threats.
   - Acts as a buffer between the internet and internal network.
3. **Data Isolation Perimeter (DIP)**
   - Encloses highly sensitive assets such as databases, and logs.
   - Heavily restricted; only authorized systems can communicate here.
4. **Internal Trusted/Militarized Zone (MZ)**
   - Contains core infrastructure like application servers, internal tools, and directories.
   - Accessible only to internal employees or approved systems.
5. **Management Zone:**
   - Dedicated to admin consoles, SIEM, orchestration platforms, and monitoring tools.
   - Enforces multi-factor authentication (MFA) and session monitoring.
6. **Endpoint Zone:**
   - Workstations, laptops, and mobile devices used by employees.
   - Connected to internal apps but separated from sensitive systems.
7. **Development/Testing Zone (UAT):**
   - Hosts non-production environments.
   - Segregated from production to prevent accidental data leakage or threat propagation.
8. **Guest/IoT Zone:**
   - Isolated space for visitor devices, BYOD, and IoT systems.
   - Internet-only access with strict network controls.
9. **Backup/Disaster Recovery Zone:**
   - Contains replicated data, snapshots, and offsite backups.
   - Often air-gapped or protected with immutable storage policies.

4.2 WIRELESS LOCAL AREA NETWORKS (WLANS) SECURITY

4.2.1 Strong Password and Passphrase

Strong passwords are crucial for WLAN security and protecting networks from unauthorized access. (*Refer to Section 3.4 Password and Authentication Policies).*

4.2.2 Hidden SSID

Hiding the SSID (Service Set Identifier) of a WLAN network enhances security by preventing the network name from being broadcast, making it less visible to casual users and potential attackers. While this adds a layer of obscurity and may deter some unauthorized users, it is not completely secure; determined attackers can still discover hidden networks using specialized tools.

4.2.3 Mac Address Filtering

MAC address filtering is a network security feature that allows administrators to control access to a WLAN network based on the unique hardware addresses of devices. Each device, whether it's a laptop, smartphone, or tablet, has a unique Media Access Control (MAC) address assigned by the manufacturer. Using MAC address filtering, network administrators can create a list of allowed or denied devices, enhancing network security.

When MAC address filtering is enabled, the router or access point will only allow devices with specified MAC addresses to connect to the network. This means that even if an unauthorized user knows the network's SSID and password, they will be unable to connect unless their device's MAC address is on the allowed list. Conversely, administrators can also configure the network to deny access to specific MAC addresses, preventing certain devices from connecting.

While MAC address filtering can act as a basic control to prevent unauthorized devices from accessing the network, it should not be solely relied upon. Attackers, upon identifying a whitelisted MAC address, can spoof it to gain unauthorized access. MAC addresses are not encrypted and can be easily captured using tools like packet sniffers. Therefore, MAC filtering must be augmented with continuous network monitoring, anomaly detection, and alerting mechanisms to identify spoofed device behaviour and unauthorized access attempts.

4.2.4 Encryption Protocol

Encryption protocols are the backbone of WLAN security, protecting data transmitted between devices and networks. These protocols ensure sensitive information remains secure from unauthorized access and eavesdropping. Over the years, various encryption protocols—**WEP, WPA, WPA2, and WPA3**—have been developed, each with their strengths and weaknesses.

a) Wired Equivalent Privacy (WEP)

WEP was the first encryption protocol introduced for Wi-Fi networks, designed to provide a security level equivalent to that of a wired connection. It uses static keys and a simple encryption mechanism to secure communications. However, WEP has significant vulnerabilities due to its weak initialization vector (IV) and reliance on predictable static keys. These shortcomings make it easy for attackers to crack the encryption using readily available tools. As a result, WEP is considered obsolete and is no longer recommended for securing modern networks.

b) Wi-Fi Protected Access (WPA)

WPA was introduced as an interim solution to address the critical flaws in WEP. It replaced WEP's static key system with the Temporal Key Integrity Protocol (TKIP), which dynamically generates encryption keys for each session. While WPA provided better security than WEP, it still had limitations, especially when using TKIP. Vulnerabilities like replay attacks and brute force attempts highlighted the need for a

more robust solution. WPA has since been largely phased out in favour of WPA2 and WPA3, though it remains a fallback option for older devices.

c)Wi-Fi Protected Access 2 (WPA2)

WPA2, released in 2004, became the industry standard for Wi-Fi security for more than a decade. It introduced the Advanced Encryption Standard (AES), replacing TKIP, to provide stronger encryption. WPA2 offers two operational modes:

- **WPA2-Personal** (PSK): This mode uses a pre-shared key for encryption, making it suitable for small-scale networks like home or small offices.
- **WPA2-Enterprise**: This mode employs 802.1X authentication and a RADIUS server for centralized management and stronger security, ideal for corporate or institutional networks. Despite its advancements, WPA2 is not without flaws. It is vulnerable to certain attacks, such as the KRACK (Key Reinstallation Attack), which exploits weaknesses in the four-way handshake process. While mitigations exist, these vulnerabilities have underscored the need for even stronger protocols.

d)Wi-Fi Protected Access 3 (WPA3)

WPA3, introduced in 2018, represents the latest and most secure Wi-Fi encryption standard. It was developed to address the vulnerabilities of WPA2 while improving ease of use and security. WPA3 includes two modes:

- **WPA3-Personal**: This mode replaces the traditional pre-shared key (PSK) mechanism with Simultaneous Authentication of Equals (SAE). SAE is a key exchange protocol resistant to offline dictionary attacks and provides stronger encryption.
- **WPA3-Enterprise**: Designed for large-scale and highly sensitive networks, WPA3-Enterprise offers 192-bit encryption for compliance with stringent regulatory requirements. It also supports Perfect Forward Secrecy (PFS), ensuring that even if session keys are compromised, past communications remain secure.

WPA3 also simplifies device onboarding with features like Enhanced Open, which encrypts traffic even on open networks without the need for a password. It is backward-compatible with WPA2, ensuring a smooth transition for devices while encouraging widespread adoption of its enhanced features.

For modern WLAN networks, WPA3 is the recommended protocol due to its robust security features and advanced encryption capabilities. For enterprise environments, **WPA3-Enterprise** should be adopted to meet the demands of large networks and sensitive data. While WPA2 remains viable for environments where WPA3 is not yet fully supported, older protocols like WEP and WPA should be entirely avoided due to their inherent vulnerabilities.

4.2.5 Guest Network Security

Guest network security is crucial for providing internet access to visitors without risking the main network. Key strategies include segmenting the network with a separate SSID for guests, establishing a firewall, and using VLANs to isolate traffic. This segmentation helps prevent unauthorized access to sensitive resources.

Password management is also important; a strong frequently updated password should be used and QR codes can simplify sharing. Monitoring network activity through logging and auditing is essential to identify potential security threats.

To prevent rogue devices, MAC address filtering can restrict access and disabling peer-to-peer traffic helps avoid malware sharing. Utilizing WPA3 encryption enhances data security, and a captive portal can be configured for guest authentication before network access. Through these measures a guest WLAN network can remain secure while allowing controlled access.

4.2.6 Rogue Access Point Detection and Mitigation

Rogue access point detection and mitigation are vital for network security as rogue APs can significantly threaten an organization's infrastructure. A rogue access point either intentionally set up by attackers or inadvertently by employees can compromise data and disrupt operations.

Detection methods include Wireless Intrusion Detection Systems (WIDS), which monitor the wireless environment for unauthorized devices. When a rogue AP is detected, alerts are triggered for network administrators to act.

Mitigation involves promptly disabling or removing the rogue AP either physically or remotely. Wireless Intrusion Prevention Systems (WIPS) can also block communication between rogue APs and their clients. Additionally, MAC filtering can help prevent unauthorized devices, while strong encryption protocols like WPA3 and authentication mechanisms such as 802.1X enhance overall network security reducing the risk of exploitation by rogue APs.

4.3 SECURITY MAINTENANCE

4.3.1 Verify Software and Configuration Integrity

An adversary can compromise network devices by altering operating system files, executable code, or firmware. This can lead to data integrity violations, sensitive information exfiltration, and denial of service (DoS).

To prevent this, verify the integrity of operating system files by comparing their cryptographic hashes with the genuine hashes provided by the vendor. When upgrading, ensure that all files are verified both before and after installation.

An adversary may opt to change device configurations instead of complex modifications. Implementing a configuration change control process, including secure backups and documentation of changes (with authorization and justification), is essential. Periodically compare current configurations with backups to detect unauthorized changes and verify that no suspicious modifications were authorized.

4.3.2 Maintain Proper File System and Boot Management

Many network devices have active configurations in memory and saved ones in persistent storage. To avoid inconsistencies after a reboot or power loss, permanent changes must be saved. For temporary changes, add comments explaining their purpose and when to remove them. If the device lacks comment support, document them in a backup copy. Always use secure protocols like SFTP or SCP for remote configuration transfers and protect backup repositories from unauthorized access. Remove unnecessary old operating system files and outdated backup configurations, as multiple software versions can expose vulnerabilities that have been patched in newer releases.

4.3.3 Stay Current with Vendor-Supported Hardware

Vendors eventually stop supporting specific hardware platforms and, if a failure occurs, these end-of-life devices cannot be serviced. In addition to device instability and memory requirement concerns, there is an increased risk of an adversary exploiting the device due to a lack of software updates to fix known vulnerabilities. Newer, vendor-supported hardware platforms have improved security features, including protection from known vulnerabilities.

Once a vendor publishes an end-of-life notice or announces that a device will no longer be supported, It is recommended to construct a plan to upgrade or replace affected devices with newer equipment, according to vendor recommendations. Outdated or unsupported devices should be immediately upgraded or replaced to ensure the availability of network services and security support.

4.4 ROUTING

4.4.1 Disable IP Source Routing

IP source routing allows the sender to specify a route for packets, which can be exploited by attackers to bypass security measures like ACLs. This feature should be disabled on all devices, not just routers, as it is unnecessary for normal operations. Different vendors may require disabling each source route option individually, and similar features in IPv6 must also be disabled.

4.4.2 Enable Unicast Reverse-Path Forwarding (uRPF)

uRPF helps protect against IP spoofing by checking if the source address of an incoming packet matches the expected return path in the routing table. It should be enabled on external interfaces of perimeter routers, given that Cisco Express Forwarding (CEF) is active. However, uRPF should not be used on internal interfaces or routers with asymmetrical routing.

4.4.3 Enable Routing Authentication

Dynamic routing protocols can be vulnerable to route manipulation by unauthorized sources. To protect against this, routing authentication must be enabled for any dynamic routing protocol that receives updates from other devices. For example, OSPF routing authentication should be enabled for each OSPF area and the corresponding interfaces.

4.5 INTERFACE PORTS

4.5.1 Disable Dynamic Trunking

Dynamic trunking refers to the ability of switches to automatically negotiate a trunk link using Dynamic Trunking protocol. This allows switch ports to dynamically decide whether they should operate as an access port or trunk port. We can disable dynamic trunking by configuring the switch port as an access port or set it to trunk mode with negotiation off.

4.5.2 Enable Port Security

Switch port security is feature that helps protect network access by restricting which devices can connect to a switch port. It prevents unauthorized devices, MAC flooding attacks and spoofing by allowing only specified MAC addresses on port.

4.5.3 Disable Default VLAN

The default VLAN (VLAN 1) cannot be deleted or fully disable on switches but can mitigate risk by not using VLAN 1 for any access port, disabling VLAN 1 on trunk links, Blocking VLAN 1 traffic and assigning management interfaces to a separate VLAN.

4.5.4 Disable Unused Ports

Leaving unused ports enabled can allow unauthorized devices to connect and compromise the network. All unused ports should be disabled and assigned to a non-default unused VLAN. Verify that a port is truly unused before shutting it down, ensuring no devices are connected. Once verified, shut down all unused interfaces and assign access and native trunking VLANs to disabled VLANs.

4.5.5 Disable Port Monitoring

Port monitoring, or mirroring, allows traffic from one switch port to be copied to another for analysis. However, this can be exploited by adversaries to capture sensitive network traffic. It's advisable to disable all inactive port monitoring sessions and only enable it on necessary ports.

4.5.6 Disable Proxy Address Resolution Protocol (ARP)

Proxy ARP allows a proxy server to respond to ARP requests for IP addresses, not on the same network, which can help devices reach remote subnets but also poses security risks, enabling spoofing and interception of packets. Disable proxy ARP on all interfaces unless the device is explicitly used for LAN bridging or inbound NAT.

4.6 COMPREHENSIVE NETWORK SECURITY PRACTICES

To strengthen network security and mitigate risks associated with unauthorized access, malware, and data breaches, organizations should implement the following measures:

1. Network Security in UAT Environments

User Acceptance Testing (UAT) environments are often used for validating applications before deployment. However, when exposed to the internet, they become potential attack vectors if not properly secured. To address this:

1. Enforce firewall policies and network segmentation to isolate UAT environments from production and other critical systems.
2. Restrict public access by implementing VPN-based access controls and ensuring that only authorized users can connect.
3. Monitor and log all traffic passing through UAT environments using Intrusion Detection and Prevention Systems (IDS/IPS) to detect and mitigate potential threats.
4. Disable unnecessary services and ports in UAT instances to reduce attack surfaces.
5. Ensure strict access control and limit administrative privileges only to essential personnel.

2. Ensuring Signature Updates in Firewalls and Antivirus Solutions

Keeping security systems up to date is critical in defending against evolving cyber threats. To ensure that firewalls and antivirus solutions remain effective:

1. Enable automatic updates for firewall rules, antivirus definitions, and threat intelligence feeds to mitigate newly discovered vulnerabilities.

2. Regularly verify update logs to confirm that updates are correctly applied and functioning as expected.
3. Conduct periodic security audits to assess the effectiveness of signature updates and detect any anomalies.
4. Implement fail-safe measures to ensure continued protection even if update mechanisms temporarily fail.

3. Configuring IPS in Blocking Mode Instead of Monitoring Mode

Intrusion Prevention Systems (IPS) play a crucial role in detecting and mitigating security threats in real-time. Instead of operating in a passive monitoring mode, organizations should:

1. Configure IPS to actively block malicious traffic rather than merely logging threats for later review.
2. Set up alert thresholds and automated responses to respond to high-risk incidents immediately.
3. Regularly review IPS policies and rules to minimize false positives while ensuring effective protection.
4. Test new IPS rules in a controlled environment before deploying them into production to avoid unintended disruptions.

4. Conducting Periodic Reviews of Security Policies in Firewalls

Firewall rules govern network traffic and must be continuously reviewed to remain effective. To maintain strong security posture:

1. Establish a formal review process for firewall policies, ensuring that outdated, redundant, or overly permissive rules are removed.
2. Conduct firewall audits on a quarterly or bi-annual basis to verify compliance with security best practices.
3. Assign responsibility to security teams and network administrators for evaluating, updating, and approving firewall rule modifications.
4. Document all firewall changes as part of the security governance framework to maintain accountability and compliance.
5. Use automated tools to analyse firewall rules for anomalies, misconfigurations, or conflicts that could weaken security controls.

4.7 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Network Security	Define strategies security roles and assign authority for enforcing firewall policies, VPN restrictions, and segment strategies and continuous review for	Inventory of network devices	Configure firewalls, IDS/IPS, and Zero Trust	Monitor network traffic and detect anomalies	Incident response for network breaches	Backup network configurations and disaster recovery plans

Table 6: Network Security NIST Framework mapping

References:

NIST Special Publication 800-41 Rev. 1: https://cSource.nist.gov/publications/detail/sp/800-41/rev-1/final

NSA Network Infrastructure Security Guide: CTR_NSA_NETWORK_INFRASTRUCTURE_SECURITY_GUIDE_20220615.PDF

NIST Special Publication 800-41 Rev. 5: https://doi.org/10.6028/NIST.SP.800-53r5

NIST SP 800-44 Version 2: https://doi.org/10.6028/NIST.SP.800-44ver2

NIST SP 800-83 Rev. 1: https://doi.org/10.6028/NIST.SP.800-83r1

NIST SP 800-41 Rev. 1: https://doi.org/10.6028/NIST.SP.800-41r1

5. SERVER SECURITY

5. Server Security

An organization's servers provide a wide variety of services to internal and external users, and many servers also store or process sensitive information for the organization. Some of the most common types of servers are Web, email, database, infrastructure management, and file servers.

This security handbook for server security has been prepared with reference NIST SP 800-123 to secure servers. Servers are a frequent target of cyber attackers. To ensure the security of a server and the supporting network infrastructure, the following practices should be implemented:

- Organization-wide information system security policy
- Configuration/change control and management
- Risk assessment and management
- Standardized software configurations that satisfy the information system security policy
- Security awareness and training
- Contingency planning, continuity of operations, and disaster recovery planning
- Certification and accreditation.

5.1 SECURING THE SERVER OS

The security of an organization's IT infrastructure fundamentally relies on the robustness of its server operating systems. Servers are primary targets for cyber threats, necessitating stringent security controls to ensure the integrity, confidentiality, and availability of critical systems and data. This section outlines essential best practices for installing, hardening, and maintaining secure server operating systems.

For foundational security concepts such as Authentication, Authorization, and Accounting (AAA), Zero Trust Security, Identity and Access Management (IAM), and Logging & Monitoring, refer to Chapter 2: Foundational Security Practices.

5.1.1 Operating System Selection and Installation

The foundation of a secure server begins with the careful selection and controlled deployment of its operating system.

- **Utilize Supported and Secure OS Versions**

  Organizations must deploy only actively supported operating systems that receive regular security updates. End-of-life (EOL) versions should be strictly avoided due to their vulnerability to unpatched security flaws. Examples of enterprise-grade, security-focused OS versions include:
  - Windows Server (2019, 2022 Long-Term Servicing Channel)
  - Linux Distributions (RHEL 9, Ubuntu LTS, SUSE Enterprise Server, Amazon Linux 2023)

- **Adopt a Minimal Installation Approach**

  To reduce the attack surface, organizations should perform custom installations that exclude non-essential components. For Windows environments, Server Core should be leveraged whenever possible. For Linux-based deployments, package managers (e.g., dnf, apt, yum) should be utilized to install only required software.

- **Secure Boot and Firmware Hardening:**
  Secure Boot should be enabled in the Unified Extensible Firmware Interface (UEFI) to prevent unauthorized code execution during the boot process. Additionally:
  - BIOS/UEFI access should be password-protected.
  - Legacy boot methods (BIOS mode) should be disabled to prevent bootkit attacks.
  - Bootloader security should be enforced, such as configuring GRUB passwords in Linux environments.

5.1.2 Patch and Update Management

A comprehensive patch management strategy is critical for mitigating vulnerabilities. This process should be aligned with industry best practices, such as those outlined in NIST SP 800-40 (Guide to Enterprise Patch Management Technologies).

For a broader discussion on patch and update management, refer to Section 2.9: Patch and Update Management. The following server-specific patching practices should be considered:

- **Automated Patch Deployment**
  Security patches should be applied within 14 days for critical updates, following the risk-based approach recommended by NIST SP 800-40. Patch deployment should be automated using enterprise solutions, such as:
  - Windows Server Update Services (WSUS) and Microsoft Endpoint Configuration Manager (MECM)
  - Linux-based automated patching (e.g., dnf-automatic, unattended-upgrades)

- **Testing and Validation Prior to Production Deployment**
  Updates must be rigorously tested in staging environments to assess system stability before deployment to production servers. Organizations commonly adopt an N-1 patching policy, where the previous stable patch is used to minimize compatibility issues.

- **Kernel and Driver Security**
  - Regular kernel updates should be applied to address vulnerabilities.
  - Unnecessary kernel modules (e.g., Bluetooth, FireWire) should be disabled to reduce the attack surface.

5.1.3 Secure Configuration and Hardening

To enhance resilience against cyber threats, servers must be hardened using security baselines and configuration best practices. For general hardening principles, refer to Section 2.8: Vulnerability, Compliance Management, and Governance, which discusses system baseline configurations.

- **Disable Non-Essential Services and Ports**
  Services that are not required for operational purposes should be disabled to minimize the attack surface.
  - For Windows: Use PowerShell (Get-Service | Where-Object {$_.Status -eq "Running"}) to identify and disable unnecessary services.
  - For Linux: Utilize systemctl or chkconfig to disable non-essential daemons.

- Unused network ports should be blocked via firewalls (Windows Defender Firewall, iptables, UFW.

- **Implement Role-Based Access Control (RBAC) and Privileged Access Management (PAM):**
  - Default administrative accounts (e.g., "Administrator," "root") should be disabled or renamed to prevent enumeration attacks.
  - Access should be granted based on the principle of least privilege (PoLP) (Refer to Section 2.3: Identity and Access Management (IAM)).
  - Just-in-Time (JIT) access controls should be implemented using tools such as Microsoft LAPS for Windows environments and sudoers policies for Linux.

- **File System and Storage Hardening:**
  - Separate partitions should be allocated for critical directories (/var, /home, /tmp) to mitigate unauthorized data access.
  - Mount options such as noexec, nodev, and nosuid should be enforced for security-sensitive directories to restrict execution of unauthorized binaries.
  - File Integrity Monitoring (FIM) tools such as AIDE, Tripwire, or OSSEC should be deployed to detect unauthorized modifications (Refer to Section 2.5: Logging, Monitoring, and Audit Framework for details on file integrity monitoring).

5.1.4 Logging, Monitoring, and Incident Detection

Proactive monitoring and logging facilitate the early detection of security incidents and provide forensic evidence in the event of a breach. For a detailed overview of logging practices, refer to Section 2.5: Logging, Monitoring, and Audit Framework.

- Centralized Log Management
  Organizations should aggregate logs using Security Information and Event Management (SIEM) solutions, such as:
  - Splunk, Elastic Security (ELK Stack), Microsoft Sentinel, or IBM QRadar
  - Windows Event Forwarding (WEF) for centralizing Windows logs
  - Linux logging frameworks (rsyslog, journald, auditd)
- Real-Time Anomaly Detection
  - Host-based Intrusion Detection Systems (HIDS) such as OSSEC, Wazuh, and Tripwire should be deployed.
  - Process Activity Monitoring should be configured using:
    - Windows Sysmon for process execution tracking.
    - Linux auditd for system call monitoring.

5.1.5 Compliance with Security Standards

Organizations must align their server security posture with internationally recognized frameworks to ensure regulatory compliance.

For baseline security compliance, refer to Section 2.8: Vulnerability, Compliance Management, and Governance, which discusses CIS benchmarks and NIST standards.

- CIS Benchmarks and Secure Configuration Frameworks

- Adopt CIS Level 1 or Level 2 hardening for Windows Server, Linux, and cloud workloads.
- Automate compliance assessments using Lynis, OpenSCAP, or CIS-CAT Pro.

5.2 SECURING SERVER SOFTWARE

Beyond securing the underlying operating system (OS), it is essential to ensure that all software running on the server is hardened, properly configured, and regularly maintained to mitigate security vulnerabilities. This section outlines best practices for securing installed applications, managing software dependencies, implementing runtime security controls, and ensuring compliance with security frameworks.

5.2.1 Secure Software Installation and Configuration

A server's security can be significantly compromised by improperly installed or misconfigured software. To minimize risks, the following practices should be observed:

- Install Only Necessary Software
  - Avoid installing default packages that may introduce vulnerabilities.
  - For Linux environments, use minimal installations (dnf install --setopt=install_weak_deps=False) to prevent the inclusion of unnecessary dependencies.
  - For Windows, leverage PowerShell DSC (Desired State Configuration) to enforce software installation policies.
- Verify Software Integrity and Authenticity
  - Download software only from trusted sources, such as vendor repositories (e.g., Microsoft, Red Hat, Ubuntu, NIST-certified repositories).
  - Use digital signatures and checksums (SHA-256, GPG verification) to validate software authenticity before installation.
  - Avoid third-party or unofficial repositories unless they are rigorously vetted.
- Restrict Software Execution Privileges
  - Applications should be executed with the least privilege necessary (Refer to Section 2.3: Identity and Access Management for least privilege enforcement).
  - Configure AppLocker (Windows) or SELinux/AppArmor (Linux) to control execution policies.

5.2.2 Software Patch Management and Vulnerability Mitigation

Unpatched applications present a critical security risk that can be exploited by attackers. Effective patch management ensures that known vulnerabilities are promptly mitigated.

For details on patch management strategies, refer to Section 2.9: Patch and Update Management. The following server-specific software update best practices should be observed:

- Implement Automated Patching Mechanisms
  - Configure Windows Update Services (WSUS) or Linux package managers (dnf-automatic, apt-cron) to automatically apply security patches.
  - Ensure third-party applications (e.g., Apache, MySQL, PostgreSQL) are updated alongside OS updates.

- Monitor Vulnerabilities via CVE Databases
    - Regularly check for software vulnerabilities in databases such as NIST National Vulnerability Database (NVD) or MITRE CVE database.
    - Deploy vulnerability scanners (Nessus, OpenVAS, Qualys) to identify and remediate software vulnerabilities.
- Apply Hotfixes and Security Patches with Minimal Downtime:
    - Utilize live patching solutions (e.g., KernelCare for Linux, Windows Hot Patching) where applicable.
    - Schedule patch deployment during maintenance windows to minimize operational disruptions.

5.2.3 Secure Configuration of Critical Server Applications

Properly configuring server applications is crucial for minimizing security risks. Applications such as web servers, database servers, and application servers require specific security measures to mitigate known attack vectors.

Web Server Security (Apache, Nginx, IIS)

For a broader discussion on network security principles, refer to Section 4.1: Network Architecture and Design. The following security best practices should be implemented:

- Disable Unused HTTP Methods
    - Restrict methods such as TRACE, PUT, and DELETE unless explicitly required.
    - Configure web servers to support only HTTPS with TLS 1.2/1.3 encryption.
- Implement Secure Headers
    - Enforce security headers (Content-Security-Policy, X-XSS-Protection, Strict-Transport-Security) to prevent clickjacking, cross-site scripting (XSS), and man-in-the-middle attacks.
- Use Web Application Firewalls (WAF)
    - Deploy ModSecurity (Apache/Nginx) or Microsoft Defender for IIS to filter malicious traffic.

Database Server Security (MySQL, PostgreSQL, Microsoft SQL Server)

Refer to Section 7: Database Security for an in-depth discussion on securing databases. The following general security configurations should be enforced:

- Restrict Direct Database Access
    - Bind database services to localhost or internal subnets only.
    - Implement network segmentation to prevent unauthorized access.
- Enforce Strong Authentication and Access Controls
    - Utilize Role-Based Access Control (RBAC) to assign granular privileges.
    - Implement Multi-Factor Authentication (MFA) for database administrative accounts.
- Enable Auditing and Logging
    - Configure database activity monitoring to log unauthorized access attempts, privilege escalations, and SQL injection attempts.

5.2.4 Runtime Security and Execution Control

Once installed, software must be continuously monitored to detect anomalies, prevent unauthorized execution, and enforce security baselines.

For details on logging, monitoring, and auditing, refer to Section 2.5: Logging, Monitoring, and Audit Framework. The following runtime security controls should be enforced:

- Implement Application Sandboxing and Process Isolation
  - Utilize Docker or Kubernetes namespaces to contain application processes.
  - Enable Linux namespaces (cgroups, seccomp filters) to restrict process capabilities.
- Monitor Process Execution and System Calls
  - Deploy Sysmon (Windows) or auditd (Linux) to track process creation and modifications.
  - Utilize Security Enhanced Linux (SELinux) or AppArmor to enforce execution control policies.
- Enable Behaviour-Based Threat Detection
  - Leverage Extended Detection and Response (XDR) solutions for anomaly detection.
  - Integrate SIEM solutions (e.g., Splunk, ELK, QRadar) for real-time threat intelligence.

5.2.5 Compliance with Security Standards

Organizations must ensure that all installed software adheres to recognized security frameworks and regulatory requirements.

For a broader discussion on compliance and security governance, refer to Section 2.8: Vulnerability, Compliance Management, and Governance. The following guidelines should be followed:

- Enforce Secure Configuration Baselines
  - Utilize CIS Benchmarks for Web Servers, Databases, and Application Servers.
  - Implement SCAP (Security Content Automation Protocol) checks for compliance validation.
- Align with Industry Frameworks
  - NIST SP 800-123: Security guidelines for managing server applications.
  - PCI-DSS (if handling payment data): Secure application controls and logging requirements.
  - ISO/IEC 27001: Application security and software development lifecycle (SDLC) compliance.

5.3 PHYSICAL SECURITY

Ensure the servers and network devices are secured with a locked rack or in an area with restricted access. Ensure the Server and network devices are set up in restricted network access. Disable/restrict removable media access in the servers and endpoints.

5.4 FORENSIC READINESS OF SERVERS

Forensic readiness refers to the proactive measures taken to collect, protect, and preserve digital evidence in a structured manner, ensuring a rapid and effective response to security incidents. A well-prepared forensic strategy allows organizations to:

- Minimize incident response time by maintaining readily available logs and evidence.
- Ensure legal admissibility of digital evidence by following forensic best practices.
- Reduce the risk of data loss or tampering by implementing secure log retention policies.
- Support compliance with security frameworks such as NIST SP 800-86 (Guide to Integrating Forensic Techniques), ISO/IEC 27037 (Digital Evidence Handling), and GDPR Incident Reporting.

5.4.1 Key Components of Forensic Readiness

I. Secure and Comprehensive Logging

Forensic investigations rely heavily on logs to trace security events. Organizations should:

a. Enable detailed logging of system activities, including:
   i. User authentication and session logs.
   ii. File access and modifications.
   iii. Privileged access and sudo command execution.
b. Forward logs to a centralized SIEM (Security Information and Event Management) system for real-time analysis and long-term retention (Refer to Section 2.5: Logging, Monitoring, and Audit Framework).
c. Protect logs from unauthorized modification by:
   i. Using append-only storage (Linux chattr +a).
   ii. Enforcing WORM (Write Once, Read Many) storage policies.

II. Controlled Access and Audit Trails

Access control plays a vital role in forensic readiness (Refer to Section 2.3: Identity and Access Management (IAM)). To enhance forensic preparedness:

a. Enforce role-based access control (RBAC) to restrict privileged access.
b. Enable audit trails for administrative commands and critical system changes (e.g., sudo logs, Windows Event Viewer security logs).
c. Use session recording tools (e.g., TTY logging for Linux, Microsoft LAPS for Windows) to capture forensic evidence of privileged user actions.

III. Network and Endpoint Monitoring

Proactive network monitoring helps detect security incidents before they escalate. Organizations should:

a. Deploy host-based intrusion detection systems (HIDS) such as OSSEC, Wazuh, or Tripwire.
b. Implement packet capture solutions (e.g., Zeek, Suricata, Wireshark) to log network activity for forensic analysis.
c. Use endpoint detection and response (EDR) tools for continuous monitoring of server processes.

IV. Incident Response Playbooks and Chain of Custody

Forensic readiness requires a structured incident response plan with defined procedures for:

a. Preserving evidence (e.g., isolating compromised systems, creating forensic disk images).
b. Following legal chain-of-custody procedures to ensure evidence integrity.
c. Using forensic tools such as Autopsy, FTK Imager, and Volatility for memory analysis and investigation.

(References: NIST SP 800-86: Guide to Integrating Forensic Techniques into Incident Response, NIST SP 800-92: Guide to Computer Security Log Management,)

5.5 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Server Security	Define server security roles, assign authority for patch management, enforcing hardening policies and incident response and continuous review process for improving compliance,	Maintain server inventory and classify critical servers	Harden OS, restrict access, enable EDR	Log monitoring and SIEM alerts	Forensic analysis and isolate compromised servers	Restore server images and test recovery

Table 7: Server Security NIST Framework mapping

Reference: NIST SP 800-123: https://cSource.nist.gov/publications/detail/sp/800-123/final

6. STORAGE SECURITY

6. Storage Security

This chapter establishes effective risk mitigation through a consistent approach to the planning, design, documentation, and implementation of data storage security. It covers the protection of stored information and its security during transmission across communication links Storage security is a fundamental concern for all individuals involved in data storage, including senior managers, procurement specialists, non-technical users, and information security professionals. It is also relevant for managers overseeing storage operations and the overall security program, as well as those involved in storage network security architecture.

6.1 STORAGE TECHNOLOGIES OVERVIEW

Storage technologies can be categorized into different types based on their architecture and data access methods. This section outlines the key storage systems that need securing:

6.1.1 Block Storage

- Block Storage Service: This is commonly used in Storage Area Networks (SANs), which provide block-level access to storage devices over a high-speed network, such as Fiber-Channel or iSCSI.
- SANs are ideal for mission-critical applications that require low-latency access.
- Storage Virtualization: Many SANs implement storage virtualization to aggregate physical storage into virtual pools, providing flexibility in resource allocation.

6.1.2 File Storage

- File Storage Service: Often implemented as Network Attached Storage (NAS), file storage services present storage as files and folders, typically shared across networks using protocols like NFS, SMB, or CIFS. File storage is ideal for sharing data across multiple clients.
- Advanced NAS: Features such as multi-protocol support and parallel NAS (pNFS) enhance scalability and provide a higher level of performance for large-scale environments.

6.1.3 Object Storage

Object storage is designed for scalability and managing large amounts of unstructured data. Data is stored as objects in containers, identified by unique IDs and metadata. It is often used in cloud environments and large data repositories like content libraries.

6.1.4 Software-Defined Storage (SDS)

This architecture separates storage management software from the physical storage hardware. SDS allows flexibility in using heterogeneous storage devices and simplifies provisioning, replication, and data services. SDS is commonly deployed in cloud and hyper-converged infrastructure.

6.1.5 Cloud Storage

Cloud-Based Storage Systems: Cloud storage offers scalable, on-demand storage solutions, providing object, file, and block storage services over the internet. Cloud storage enables features like replication, high availability, and archiving with minimal physical infrastructure requirements.

Advanced Cloud Services: Many cloud services provide additional features such as managed databases, data lakes and shared filesystems, designed to store and process both structured and unstructured data.

6.2 RISKS AND THREATS TO STORAGE SECURITY

Data Breaches: Unauthorized access to sensitive data stored within a storage system can lead to significant losses and legal repercussions.

Data Corruption: Errors in writing, reading, or transmitting data may cause corruption, which can lead to data loss or system failures.

Temporary or Permanent Loss of Availability: System failures, misconfigurations, or attacks can render data inaccessible, causing operational disruptions.

Failure to Meet Regulatory Compliance: Not adhering to legal or regulatory requirements can lead to penalties, fines, and loss of trust.

Storage Hardware: Increasing risk in common types of storage hardware (HDDs, SSDs, NAS, SAN, Cloud storage) due to increase attack surface on cloud and hybrid environment. Storage hardware is highly vulnerable to threats such as Physical threats, Cyber threats such as malware, unauthorized access, insider threats & data corruption.

Secure Disposal of Storage Media: Importance of securing storage media disposal to prevent deleted data recovery, mitigate data breach risk, comply with legal & regulatory requirements and reduce insider threats & third-party risks.

To mitigate these risks, organizations should implement robust storage security measures.

6.3 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Storage Security	Data governance policies and encryption standards	Identify storage assets and classify sensitive data	Encrypt data, restrict access, implement DLP	Monitor access logs and detect anomalies	Respond to unauthorized access and ransomware	Restore from secure backups

Table 8: Storage Security NIST Framework mapping

7. DATABASE SECURITY

7. Database Security

Database security ensures the protection of data confidentiality, integrity, and availability by implementing strict access controls, encryption mechanisms, and audit frameworks. This section outlines key strategies to secure databases, with cross references to foundational security practices to avoid redundancy.

7.1 INSTALLATION AND PATCH MANAGEMENT

Database Versioning: Ensure the use of the latest and most secure database versions compatible with operational requirements. Avoid unsupported or end-of-life (EOL) versions.

Patching: Regularly apply vendor released patches to address vulnerabilities (see Patch and Update Management, Section 3.9 for details).

Placement on Non-System Partitions: Configure databases to store data on non-system partitions, ensuring separation from critical OS files.

7.2 ACCESS CONTROLS AND PRIVILEGES

Role Based Access Control (RBAC): Define granular roles to enforce the principle of least privilege (Refer Section 3.3).

Authentication Standards: Enforce strong password policies, MFA and centralized authentication through solutions like Active Directory or LDAP (Refer Section 3.4).

Monitoring Elevated Access: Implement controls to audit and restrict superuser privileges and track all high privilege activities (Refer Section 3.5).

7.3 ENCRYPTION STANDARDS

Encryption for Data in Transit: Secure data transmission using TLS/SSL to prevent interception and ensure communication integrity.

Encryption for Data at Rest: Employ Transparent Data Encryption (TDE) or similar methods to encrypt databases at the tablespace or column level.

Key Management: Follow robust key management practices, such as the use of hardware security modules (HSMs) or secure software-based solutions, as described in Encryption and Data Security, Section 3.7.

7.4 AUDITING AND LOGGING

- Unified Auditing: Enable auditing to track activities, including user logins, data access, privilege changes and modifications to database objects.
- Centralized Log Management: Utilize SIEM tools to consolidate logs, monitor for anomalies and generate compliance reports (Refer to Logging and Monitoring Framework, Section 3.5).

7.5 BACKUP AND RECOVERY

- Data Backup Policies: Implement full and incremental backups to ensure recovery during data loss or corruption events. Refer to Backup, Retention, and Disaster Recovery, Section 3.6.

- Testing Recovery Plans: Regularly test recovery objectives (RTO and RPO) to validate backup integrity and system reliability during incidents.

7.6 FILE AND DIRECTORY PERMISSIONS

Restricted Access: Assign proper file permissions to database directories, binaries, log files, and backups. Follow principles outlined in Access Review and Policies, Section 3.3.

Protect Logs: Secure log files from unauthorized access to prevent sensitive data leakage.

7.7 THREAT DETECTION AND MITIGATION

- RealTime Monitoring: Use database activity monitoring (DAM) tools to detect suspicious behaviour and unauthorized access attempts (see RealTime Monitoring, Section 3.5).
- Anomaly Detection: Leverage advanced analytics to identify unusual patterns in query behaviour or access locations.
- Incident Response: Develop playbooks for specific scenarios, such as SQL injection or privilege abuse, as described in Incident Response Planning, Section 3.5.

7.8 SECURE CONFIGURATION MANAGEMENT

Secure Configuration Management for databases is a critical component of overall database security. It involves establishing, maintaining and monitoring configurations to ensure databases remains in secure state throughout lifecycles.

- Establishing secure Baselines: Define and document a secure configuration baseline for all database system. This baseline should reflect industry best practices and OEM recommendations.
- Hardening the Database environment: Disable or remove non-essential features, services and components (e.g., unused stored procedures, default accounts) to reduce the attack surfaces.
- Patch & Update Management: Establish a routine process to apply security patches and updates to both the database software and its underlying OS.
- Data Protection and Encryption: Enable encryption for stored data and ensure the data in transit is protected via secure protocols and also protect configuration files containing sensitive information with strict access control and encryption where applicable.

7.9 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Database Security	Define DB security policies and retention policies	Maintain DB inventory and classify data	Encrypt data, implement RBAC	Monitor queries and detect anomalies	Block malicious queries and rollback transactions	Restore database and perform integrity checks

Table 9: Database Security NIST Framework mapping

8. ENDPOINT SECURITY

8. Endpoint Security

8.1 INTRODUCTION

Endpoint security refers to the process of securing endpoints of end-user devices such as desktops, laptops, and mobile devices from being exploited by malicious actors. It involves deploying security measures and tools to protect these devices against threats like malware, ransomware, and unauthorized access, ensuring data integrity and network safety.

8.1.1 Importance of Endpoint Security

Endpoint Security is crucial for safeguarding sensitive data and preventing cyber-attacks especially in a connected network environment. It helps protect devices from

- Malware
- Phishing
- Unauthorized access
- Data breaches
- Financial loss
- Reputational damage

Additionally, endpoint security supports regulatory compliance by meeting data protection standards, and it ensures business continuity by minimizing the impact of potential security incidents on daily operations.

8.1.2 Security Risks and Threats to Endpoints

Endpoints face various security risks and threats that can compromise their integrity and the network they connect to

- Malware: Malicious software designed to disrupt, damage or gain unauthorized access to systems. This includes viruses, worms, and spyware.
- Ransomware: A type of malware that encrypts files on a device, demanding payment for decryption this can lead to significant data loss and operational downtime.
- Phishing Attacks: Social engineering tactics that trick users into revealing sensitive information such as login credentials by masquerading as trusted entities.
- Unauthorized Access: Instances where attackers gain access to devices through weak passwords, unpatched vulnerabilities or exploitation of insecure networks, leading to data breaches. Endpoints devices can be lost or stolen which may lead to unauthorized access to data.
- Insider Threats: Risks posed by employees or contractors who may intentionally or unintentionally compromise security either through negligence or malicious actions.

8.1.3 Benefits of Implementing Strong Endpoint Security

Implementing strong endpoint security provides several key benefits including:

- Comprehensive Protection: Shields devices from various cyber threats, reducing the risk of data breaches and malware infections.

- Data Integrity: Ensures the confidentiality and integrity of sensitive information by preventing unauthorized access.
- Regulatory Compliance: Helps organizations meet industry standards and regulations, reducing the risk of legal penalties.
- Increased Productivity: Minimizes downtime from security incidents, allowing employees to work efficiently.
- Centralized Management: Enables IT teams to monitor and manage security across all endpoints effectively.

8.2 ENDPOINT SECURITY ARCHITECTURE AND DESIGN

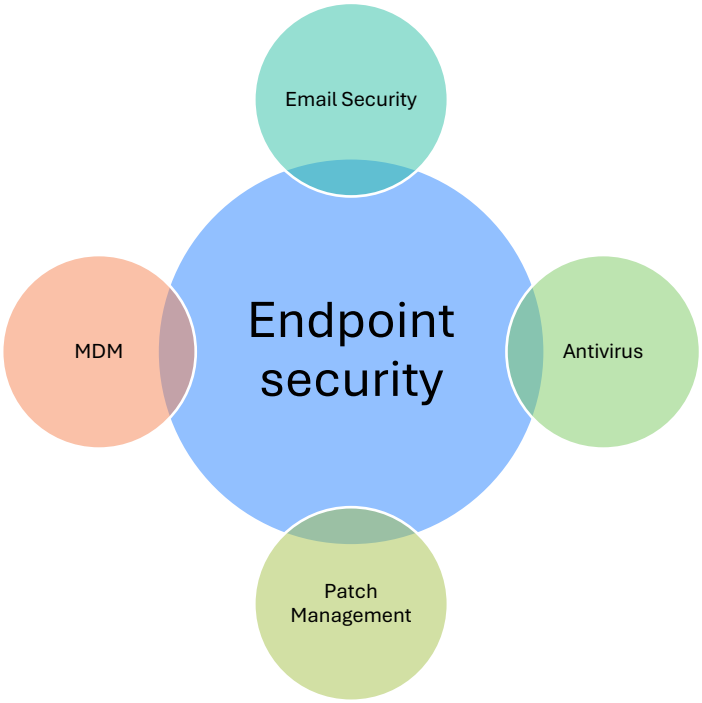

Figure 11: Components of Endpoint Security Architecture

8.2.1 Centralized Management Console

A centralized management console enhances the effectiveness of endpoint security by providing comprehensive visibility and control over all devices in the network. Enables administrators to create, implement and enforce security policies across all endpoints, ensuring consistent security measures. Allows continuous tracking of endpoint activities providing instant visibility into potential security threats and vulnerabilities. Centralized visibility allows for better monitoring of security events, while automated response capabilities enable quick actions against threats.

8.2.2 Integration with Network and Cloud Security

Integrating endpoint security with network and cloud security creates a unified defence strategy. This integration enables holistic threat detection across endpoints and network traffic, ensuring consistent security policies are enforced across all environments. It also enhances data protection for sensitive

information stored in the cloud through encryption and secure access controls. Overall, this integration strengthens the organization's security posture against evolving cyber threats.

8.2.3 Design Considerations for Endpoint Security Solutions

When designing endpoint security solutions several key considerations should be considered to ensure effectiveness and adaptability:

- Scalability: The solution should be easily scaled to accommodate an increasing number of endpoints as the organization grows, ensuring consistent protection without performance degradation.
- Compatibility: Ensure compatibility with existing hardware, software, and operating systems to facilitate seamless integration and minimize disruption during deployment.
- User Experience: The solution should be user-friendly providing minimal impact on productivity while ensuring security measures are effective and straightforward for end users.
- Comprehensive Coverage: Design the solution to protect various types of endpoints, including desktops, laptops, mobile devices, and servers against a wide range of threats.
- Real-Time Monitoring and Reporting: Incorporate capabilities for real-time monitoring and reporting to quickly identify and respond to security incidents and vulnerabilities.
- Centralized Management: Enable centralized management to streamline security policy enforcement, updates and incident response across all endpoints.
- Integration Capabilities: Consider how the endpoint security solution will integrate with other security tools such as network security and cloud security solutions, to provide a unified defence approach.
- Cost Effectiveness: Evaluate the total cost of ownership, including licensing, maintenance, and support, to ensure the solution provides value without exceeding budget constraints.
- Storage Encryption: Data protection should be considered in case of physical compromise of endpoints such as laptops, mobile devices and USB drives are prone to lost or stolen. Storage encryption ensures that even if an unauthorized party gains physical access to the device, the data remains unreadable without the proper decryption keys.

8.3 HARDWARE SECURITY CONTROLS OF LAPTOPS AND DESKTOPS

8.3.1 BIOS and UEFI Security

BIOS (Basic Input/Output System) and UEFI (Unified Extensible Firmware Interface) are critical firmware interfaces for booting computers. Key security aspects include:

- Firmware Integrity: Protect against unauthorized modifications.
- Secure Boot: Only allows trusted software to load.
- Password Protection: Restrict access to BIOS/UEFI settings.
- Firmware Updates: Regularly patch vulnerabilities.
- Hardware-based Security: Use TPM for cryptographic key protection.

8.3.2 Secure Boot and Device Integrity

Secure Boot ensures only trusted software runs at startup, preventing malware. Device integrity verification confirms hardware and software configurations remain authorized. Regular integrity checks validate system files against unauthorized changes. The Trusted Platform Module (TPM) plays

a key role in system integrity. In addition to securely storing cryptographic keys, the TPM maintains Platform Configuration Registers (PCRs), which capture the integrity measurements of boot components. These values are then compared against expected measurements during the measured boot process, ensuring that the system has not been tampered with before control is handed over to the operating system.

8.4 OPERATING SYSTEM SECURITY

8.4.1 OS Hardening Techniques

OS hardening techniques enhance security through various mechanisms, though their applicability may depend on hardware features, organizational needs and the context of use. Key techniques include:

- **Hardware Security Modules (HSMs):** Protect cryptographic keys and perform encryption/decryption tasks securely. These modules are highly effective in environments requiring secure key management, but their implementation depends on hardware support, integration efforts and budget availability. When HSMs are not feasible, software-based key management solutions can serve as an alternative.
- **Virtualization:** Isolates environments using hypervisors to protect against malware and unauthorized access. This technique is broadly applicable and highly effective in environments leveraging virtual machines or containerized workloads, aligning with modern infrastructure practices.
- **Trusted Execution Environments (TEEs):** Create isolated areas within the CPU for executing sensitive code securely. TEEs are hardware-dependent and particularly relevant in scenarios requiring strong data confidentiality and integrity such as financial transactions or secure communications.
- **Secure Boot:** Validates firmware and software at startup to ensure only trusted code is executed. Widely supported by modern hardware, this technique enhances system integrity during the boot process and aligns with security frameworks like NIST SP 800-53.
- **Memory Protection:** Utilizes techniques like Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) to mitigate vulnerabilities. These protections are widely supported and help defend against common memory-related attacks, making them an essential part of OS security baselines.

8.4.2 Service and Application Configuration

- Regular Updates: Ensure operating systems and applications are updated frequently to address vulnerabilities.
- Automated Patch Deployment: Use automation tools to streamline the deployment of patches across multiple endpoints, reducing manual effort and errors.
- Testing and Validation: Test patches in a controlled environment before widespread deployment to ensure compatibility and prevent disruptions.
- Inventory Management: Maintain an inventory of software and hardware to track which systems require updates.
- User Communication: Inform users about upcoming updates and potential impacts to minimize disruption during installation.

8.5 SECURE INFORMATION HANDLING

8.5.1 File and Disk Encryption Solutions

File and disk encryption ensures data confidentiality by converting data into a secure format, readable only with a decryption key. Common solutions include BitLocker (Windows) and File Vault (macOS) for full-disk encryption, VeraCrypt for open-source cross-platform use and enterprise tools like Symantec Endpoint Encryption and McAfee Complete Data Protection which offer centralized management. Lightweight options like 7-Zip and AxCrypt provide file-specific encryption. Best practices include using strong passwords, centralized management for organizations, regular backups, and compliance with regulations to safeguard data effectively.

8.5.2 Secure Data Storage Practices

Secure data storage involves encrypting data at rest and in transit, using access controls like role-based access and multi-factor authentication, and performing regular backups stored separately. Practices like data masking and tokenization add extra security layers, especially in non-production environments. For cloud storage, choose trusted providers with strong security measures and perform regular audits to identify vulnerabilities. These practices ensure data safety and compliance with regulations.

8.5.3 Data Loss Prevention (DLP) Policies

Data Loss Prevention (DLP) policies help protect sensitive information from unauthorized access, sharing, or leaks. Key elements include:

- Data Identification: Classify sensitive data like personal, financial or proprietary information.
- Access Control: Restrict data access based on roles and need-to-know principles.
- Monitoring and Alerts: Track data transfers and trigger alerts for suspicious activities.
- Data Encryption: Encrypt sensitive data to ensure security during storage and transfer.
- Automated Actions: Automatically block, quarantine or redact data to prevent leaks.

8.5.4 Data Classification and Discovery

Data Discovery helps to identify and locate the sensitive and critical data across organization infrastructure. Data Classification helps to categorize data based on sensitivity, criticality and regulatory requirements.

- Risk Management and Threat Mitigation: Understanding Data sensitivity in organization to apply the right security measure based on data value and risk exposure. This enables the proactive data-centric security by identifying and securing high-risk data assets from potential breaches.
- Data Protection and Encryption: Critical classified data can be protected using encryption and strict access control. Data masking and tokenization can also be used for confidential and sensitive data, implementing anonymization techniques enhances privacy protection.

8.6 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Endpoint Security	Define endpoint security policies (AV, patching)	Identify managed and unmanaged endpoints	Apply patches, deploy EDR/XDR	Detect malware and suspicious activities	Isolate infected endpoints and respond to malware	Reimage endpoint and restore user data

Table 10: Endpoint Security NIST Framework mapping

Reference:

NIST SP 800-46 Rev. 2: https://cSource.nist.gov/publications/detail/sp/800-46/rev-2/final

NIST SP 800-124 Rev. 2: https://cSource.nist.gov/pubs/sp/800/124/r2/final

9. EMAIL SECURITY

9. Email Security

9.1 DEFINITION OF EMAIL SECURITY

Email security refers to the measures and technologies used to protect email accounts, communications and data from unauthorized access, attacks, and threats such as phishing, malware, spam, and data breaches. This includes encrypting email messages authenticating email senders, filtering malicious content and implementing policies to ensure that sensitive information is securely transmitted and stored. The goal is to safeguard both individuals and organizations from email-related vulnerabilities and cyber-attacks.

9.1.1 Benefits of Email Security

1. Protection Against Phishing Attacks: Email security helps prevent employees from falling victim to phishing scams by filtering suspicious emails and flagging potential threats.
2. Prevention of Malware and Ransomware: Advanced email security techniques detect and block malicious attachments or links, reducing the risk of malware and ransomware infections.
3. Data Loss Prevention (DLP): Email security can prevent sensitive information from being sent to unauthorized recipients, protecting confidential company and customer data.
4. Regulatory Compliance: Email security ensures compliance with industry-specific regulations, such as GDPR by securing email communications and safeguarding sensitive information.
5. Business Continuity: Email security helps maintain smooth operations by preventing disruptions caused by attacks, ensuring that employees can communicate and collaborate safely.
6. Reduced Spam: By filtering out spam and unwanted emails, email security improves productivity and reduces the risk of spam-related threats.
7. Email Encryption: Ensures that sensitive information transmitted via email is encrypted and protected from unauthorized access during transmission.
8. Prevents Account Compromise: Tools like multifactor authentication (MFA) and email monitoring help prevent unauthorized access to email accounts, reducing the risk of account takeovers.

9.2 EMAIL CONTENT SECURITY

9.2.1 Email Filtering

Email filtering is the process of automatically sorting emails based on predefined rules and policies. It includes:

1. Spam Filtering: Blocks unsolicited and irrelevant emails (spam) from reaching user inboxes, reducing distractions and potential security risks.

2. Phishing Detection: Identifies and blocks phishing emails, which are designed to deceive users into sharing sensitive information like login credentials or financial data.

3. Malware Scanning: Filters out emails that contain malicious attachments or links, preventing malware, ransomware, and other threats from infiltrating the network.

4. Blacklist and Whitelist Management: Filters emails from known malicious or suspicious senders by using blacklists, while allowing trusted communications through whitelists.

5. Advanced Threat Protection (ATP): Scans emails for advanced threats using techniques like sandboxing, where suspicious attachments are opened in an isolated environment to detect hidden malware.

9.2.2 Content Control

Content control refers to monitoring and managing the actual content within emails, ensuring that sensitive or inappropriate information is not transmitted in violation of company policies or regulations. This includes:

- Data Loss Prevention (DLP): Prevents unauthorized sharing of sensitive information, such as intellectual property, customer data, or financial information, via email. DLP policies detect specific keywords, patterns, or types of content (like credit card numbers) and block or flag the email.
- Compliance with Regulations: Helps organizations enforce compliance with legal regulations (e.g., GDPR) by ensuring that email content is handled and transmitted securely. This includes ensuring that emails with confidential information are encrypted or restricted from being sent outside the organization.
- Preventing Insider Threats: Monitors outgoing emails for suspicious activities, such as attempts by employees to share confidential information with external recipients.
- Blocking Inappropriate Content: Filters email content for inappropriate language, harmful links, or materials that violate corporate policies, maintaining a professional communication environment.
- Information Rights Management for Email Security: - IRM enforces persistent protection by applying encryption and usage policies to email content, ensuring data security even after delivery by preventing forwarding, restricting copy/paste & print, expiration access, track and revoke access, applying offline protection which ensures security even after email is downloaded.

9.2.3 Email Encryption

Email encryption is the process of encoding email messages and their attachments so that only authorized recipients can read them. This ensures that the content remains confidential during transmission and cannot be intercepted or accessed by unauthorized parties. Email Encryption Protocols & Technologies: -

- TLS (Transport Layer Security): - Encrypts email in transit.
- S/MIME (Secure/Multipurpose Internet Mail Extensions): - Digital signatures and encryption for email authentication.
- DKIM, DMARC, SPF: - Preventing email spoofing and ensuring email authenticity.

9.2.4 Email Archiving and Retention Policies

Benefits of Email Archiving and Retention Policies:

- Regulatory Compliance: Helps meet industry regulations and avoid fines for non-compliance.
- Operational Efficiency: Reduces storage costs and improves access to important information.

- Legal Protection: Provides a verifiable record of email communication, which can be crucial in legal disputes.
- Data Security: Protects archived emails from unauthorized access or loss.
- Data Governance: Ensures that the organization manages data responsibly, retaining only necessary emails and deleting outdated information.

9.3 SANDBOXING

Sandboxing is a critical control to prevent malicious email attachments or links from compromising systems. A sandbox provides an isolated environment to analyze suspicious content without risking harm to the actual network or endpoints. Here are the controls and best practices to implement sandboxing effectively in email security:

Security Controls

1) Attachment Analysis:

   a) Isolate Suspicious Attachments Configure the sandbox to quarantine email attachments until scanned.

   b) File Type Filtering Define a policy to prioritize high-risk file types like .exe, .js, .pdf, `.docx, .xlsx, etc.

9.4 EMAIL SECURITY MAINTENANCE

9.4.1 Spam and Phishing Prevention

Spam and phishing prevention are critical components of an organization's cybersecurity strategy. Spam refers to unsolicited and often irrelevant emails sent in bulk, while phishing involves fraudulent attempts to steal sensitive information such as login credentials or financial data by impersonating legitimate entities. Both can lead to serious security breaches if not properly managed.

9.4.2 Malware Detection and Prevention

- Prevents Infections: Strong detection and prevention measures stop malware before it can infect devices, reducing the risk of ransomware, data breaches, or system compromise.
- Mitigates Phishing Attacks: Many phishing attacks deliver malware via email. Effective malware detection complements phishing prevention efforts by identifying malicious attachments and links.
- Protects Sensitive Data: Prevents malware from exfiltrating sensitive information or using email systems to propagate across the network.
- Reduces Downtime and Costs: Early detection of malware reduces the impact of potential attacks, minimizing downtime and the financial cost of recovery efforts.

(Reference: NIST SP 800-45 Rev. 2: https://cSource.nist.gov/pubs/sp/800/45/ver2/final)

9.5 MOBILE DEVICE MANAGEMENT

9.5.1 Introduction

Mobile Device Management (MDM) refers to software solutions used by organizations to manage, secure, and monitor mobile devices such as smartphones, tablets, and laptops that access corporate networks and data. MDM is a critical component of enterprise mobility management (EMM), enabling organizations to enforce security policies, control access, and protect sensitive data on both corporate-owned and personal devices (BYOD) ensuring compliance with security protocols and reducing the risk of data breaches or unauthorized access. Employees can securely access company resources (such as emails, apps, and documents) on their mobile devices, improving flexibility and productivity without impacting performance.

9.5.2 MDM Use Cases

Mobile Device Management (MDM) use cases illustrate how organizations can utilize MDM solutions to enhance security, improve device management, and ensure compliance. Some common MDM use cases include:

1. BYOD (Bring Your Own Device): Employees use personal devices to access corporate resources. MDM allows organizations to secure and manage these devices by enforcing policies like encryption, separating personal and work data, and allowing remote wipes for corporate data if needed.

2. Lost or Stolen Device Protection: If a device is lost or stolen, MDM enables administrators to remotely lock or wipe the device, ensuring that sensitive corporate data is not compromised.

3. App Management: MDM can control which applications can be installed on devices. This ensures that only approved apps are used and that malicious apps are blocked. It also allows for the remote installation or removal of applications as needed.

4. Compliance and Security Enforcement: Organizations can enforce regulatory compliance by ensuring devices adhere to security policies, such as requiring encryption, enabling multi-factor authentication (MFA), or preventing jailbroken/rooted devices from accessing corporate networks.

5. Remote Workforce Management: MDM supports employees working remotely by allowing secure access to corporate resources (e.g., email, VPN, files) on mobile devices. It ensures that security protocols are followed, even outside the corporate network.

6. Data Loss Prevention (DLP): MDM helps prevent data leaks by controlling the sharing, storage, and backup of corporate data on mobile devices. Sensitive data can be restricted from being copied, shared, or uploaded to unapproved cloud services.

7. Device Inventory and Monitoring: MDM provides a centralized view of all mobile devices in the organization, tracking device health, status, software versions, and compliance with security policies. This helps with proactive device management and troubleshooting.

8. Software and OS Updates: MDM allows IT teams to remotely deploy software patches and operating system updates to devices, ensuring that all devices are running the latest, most secure versions without needing manual updates by users.

9. Mobile Email Security: MDM can secure email access on mobile devices by enforcing email encryption, controlling which devices can connect to the email server, and managing email settings to prevent unauthorized access or data leaks.

10. Kiosk Mode for Devices: For specific use cases, like retail stores or public-facing services, MDM can lock devices into a single app or limited set of apps, ensuring that users cannot access other features or settings.

9.6 MDM LOCAL ADMINISTRATOR ACCOUNTS AND PASSWORDS

9.6.1 Disabling Local Administrator Accounts

Use of MDM for disabling local administrator accounts is a security measure used to limit unauthorized access to critical system settings and data on mobile devices or endpoints. By disabling local administrator accounts, organizations prevent users from making changes to the device that could compromise its security, such as installing unauthorized software, modifying system settings, or disabling security protocols.

Key Benefits of Disabling Local Administrator Accounts via MDM:

1. Enhanced Security: Reduces the risk of malware or ransomware being installed by users with local admin privileges.

2. Prevents users from tampering with security settings (e.g., disabling antivirus or firewalls) or making unauthorized changes to the system.

3. Controlled Device Configuration: Ensures that only IT administrators, through the MDM system, can make critical changes to device settings, maintaining consistency across all devices.

4. Prevents the installation of unapproved applications or software that could introduce vulnerabilities.

5. Compliance Enforcement: Helps maintain compliance with industry regulations by preventing users from bypassing security policies and ensuring that devices meet the required security standards.

6. Reduced Risk of Insider Threats: Limits the possibility of users with admin privileges intentionally or accidentally compromising the system by making unauthorized changes or accessing sensitive data.

7. Streamlined IT Management: IT teams maintain complete control over device configurations, updates, and security patches, reducing the likelihood of misconfigurations that could lead to security vulnerabilities.

9.7 ANTIVIRUS AND MALWARE PROTECTION

9.7.1 Installation of Antivirus and Anti-Malware Solutions

1) Assessment and Planning

a) Identify Requirements: Assess the specific security needs of the organization, considering factors such as the number of devices, types of operating systems, and the nature of data being protected.
b) Select the Right Solution: Choose an antivirus or antimalware solution that fits the organization's requirements. Consider factors such as detection rates, performance impact, usability, customer support, and cost.

2) Deployment Methods
   a) On-Premises Installation: For organizations that prefer to manage their security infrastructure, install antivirus and antimalware software directly on each endpoint or server. This may involve manual installation or using an automated deployment tool.
   b) Cloud-Based Solutions: Many modern antimalware solutions offer cloud-based deployment, allowing centralized management and monitoring from a web interface. This can simplify updates and reduce the administrative burden.
   c) MDM Integration: For mobile devices, antivirus and antimalware solutions can often be integrated with Mobile Device Management (MDM) platforms, allowing for centralized control over installations and policies.

3) Configuration
   a) Initial Setup: Configure the antivirus and antimalware software according to organizational policies. This includes setting up real-time scanning, scheduled scans, and automatic updates.
   b) Exclusion Policies: Define exclusion policies for specific files or directories that may produce false positives or are known to be safe (e.g., certain software installations).
   c) User Permissions: Set user permissions and roles within the software to restrict access to critical features (e.g., disabling the software or changing configurations).

4) Regular Updates
   a) Automatic Updates: Ensure that the antivirus and antimalware software is set to receive automatic updates for virus definitions and software patches. This is crucial for protecting against the latest threats.
   b) Manual Checks: Regularly verify that updates are occurring as expected, especially in environments where devices may not have consistent internet access.

5) Ongoing Monitoring and Management
   a) Centralized Dashboard: For solutions that offer centralized management, use the dashboard to monitor the security status of all devices, review alerts, and respond to detected threats.
   b) Report: Generate regular reports on malware detection, response actions taken and overall health system. This can help identify trends and inform future security strategies.

6) Incident Response
   a) Malware Removal: If malware is detected, follow the antivirus/anti-malware software's recommendations for quarantine or removal. Train IT staff on procedures for dealing with infections.
   b) Forensic Analysis: In cases of severe infections or breaches, perform forensic analysis to understand the scope of the attack, the vector of infection and the data affected.

9.7.2 Best Practices for Antivirus and Anti-malware Protection

1. Layered Security Approach: Use antivirus and antimalware solutions as part of a broader security strategy that includes firewalls, intrusion detection/prevention systems, and user training.
2. User Education: Educate employees about safe browsing habits, recognizing phishing attempts, and the importance of keeping antivirus software enabled.
3. Regular Audits: Conduct audits of antivirus and antimalware deployments to ensure compliance with organizational policies and effectiveness in detecting threats.
4. Testing and Evaluation: Periodically test the effectiveness of the antivirus and antimalware solutions by simulating attacks or using independent testing services to evaluate detection rates.
5. Response Planning: Develop and maintain an incident response plan that includes procedures for malware outbreaks, ensuring that staff know how to respond quickly and effectively.

9.7.3 Challenges in Antivirus and Antimalware Installation

1. Performance Impact: Some antivirus solutions can slow down system performance, especially during scans. Choose solutions that minimize resource consumption without sacrificing protection.
2. False Positives: Antivirus software may occasionally flag legitimate applications as malicious, which can disrupt workflows. It's important to manage exclusion lists effectively.
3. Compatibility Issues: Ensure that the selected antivirus and antimalware solutions are compatible with existing software and operating systems to avoid conflicts.
4. User Compliance: Ensuring that all users keep antivirus software active and regular updates can be challenging, particularly in BYOD (Bring Your Own Device) environments.

9.7.4 Regular System Scanning and Monitoring

- Threat Detection: Regular scans help identify malware, viruses, and other malicious software that may have infiltrated the system. This is crucial for early detection and remediation.
- Vulnerability Identification: Scans can detect vulnerabilities within the operating system and installed applications, allowing for timely patching and updates to reduce the risk of exploitation.
- Performance Monitoring: Regular scans can assess system performance and identify any unusual behavior or degradation caused by malware or other issues.
- Compliance Requirements: Many regulatory frameworks mandate regular scanning for malware as part of compliance with data protection and security standards.

Types of Scans

1. Quick Scan: A fast scan that checks the most critical areas of the system (e.g., memory, startup items, and common file locations) for potential threats. This is useful for routine checks and identifying immediate issues.
2. Full System Scan: A comprehensive scan that examines all files, applications, and settings on the device. While it takes longer, it provides a complete assessment of the system's security status.
3. Custom Scan: Allows users to specify files, folders, or drives to scan. This can be helpful when a specific area is suspected to be infected or compromised.

4. Scheduled Scans: Automated scans can be configured to run at regular intervals (e.g., daily, weekly) to ensure consistent monitoring without requiring manual initiation.

Monitoring

1. Real-Time Protection: Most modern antivirus solutions offer real-time protection that continuously monitors system activity for suspicious behavior, blocking threats as they are detected.
2. Behavioral Analysis: Advanced antivirus solutions use behavioral analysis to monitor applications and processes for unusual activities that may indicate malware infection, even if the malware signature is not yet known.
3. Alerts and Notifications: Antivirus software should provide alerts for detected threats, scan results, and other critical security updates, allowing administrators to respond promptly.
4. Centralized Management: For organizations, centralized monitoring dashboards enable IT administrators to oversee the security status of all devices in the network, ensuring comprehensive visibility and control.

9.7.5 Behavioral Analysis and Threat Detection

Behavioral analysis involves monitoring user, application, and system behaviors to identify suspicious activities. This approach helps detect threats without known signatures or those using advanced evasion techniques.

Behavioral analysis involves monitoring user, application, and system behaviors to identify suspicious activities. This approach helps detect threats without known signatures or those using advanced evasion techniques.

Key Features

- Anomaly Detection: Establishes a baseline of normal activity. Significant deviations, like an employee downloading large files late at night, may trigger alerts.
- Real-Time Monitoring: Continuous observation allows for immediate detection of suspicious actions, such as unauthorized file access or unusual logins, prompting rapid responses.
- Contextual Analysis: Incorporates contextual information (user role, location, access time) to differentiate between benign and malicious activities. For example, a login from a new location may require additional authentication.
- User Behavior Analytics (UBA): Focuses on user behaviors to identify insider threats or compromised accounts through unusual activity patterns.

Threat Detection Techniques

- Signature-Based Detection: Traditional method matching files against known malware signatures, effective for known threats but not for new variants.
- Heuristic Analysis: Evaluates file behavior to identify harmful actions, flagging suspicious behavior akin to known malware.
- Machine Learning and AI: Modern systems use machine learning to improve threat detection by recognizing patterns from historical data and adapting to evolving threats.

9.7.6 Automated Remediation and Quarantine

- Threat Identification: When malware or suspicious activity is detected, antivirus or security solution identifies the threat based on predefined criteria or heuristics.
- Predefined Response Actions: Organizations can configure their security solutions to take specific actions automatically in response to detected threats. Common actions include:
- Quarantining the Threat: Moving the malicious file to a secure area where it cannot harm the system or spread further.
- Deleting the Threat: Permanently removing the malware or infected file from the system.
- Rolling Back Changes: Reverting any changes made by the malicious software to restore the system to its previous state.
- Blocking Network Access: Preventing the infected device from accessing the network to limit potential lateral movement of the threat.
- Integration with Other Security Solutions: Automated remediation often works in conjunction with other security tools, such as firewalls and intrusion detection systems, to provide a more comprehensive response to threats.

Key Features of Quarantine

- Isolation: Quarantined files are moved to a designated area within the security solution where they cannot execute or interact with other files. This prevents the spread of malware.
- User Notification: Users and system administrators are often notified when items are quarantined, allowing them to review the actions taken and decide further steps.
- Review and Restoration Options: Administrators can analyze quarantined files to determine if they are truly malicious. A file can be restored to its original location if it is deemed safe. If it's confirmed as malware, it can be permanently deleted.

9.8 PRINTER SECURITY MAINTENANCE

9.8.1 Firmware and Software Updates for Printers

1. Firmware Updates Defined

Firmware is low-level software programmed into the printer's hardware, controlling its functions and features. It is essential for the printer's operation, providing instructions for the printer to communicate with computers and process print jobs.

Key Features of Firmware Updates

- Bug Fixes: Updates often resolve known issues or bugs that may affect printer performance or cause errors.
- Performance Enhancements: Manufacturers may release updates that optimize the printer's performance, improving print speed or quality.
- New Features: Firmware updates can introduce new functionalities, such as support for additional print formats, connectivity options, or enhanced security protocols.
- Security Patches: Firmware updates often include fixes for security vulnerabilities, helping to protect the printer from potential attacks or exploits.

2. Software Updates Defined

Printer Software includes the drivers and applications that allow computers and devices to communicate with the printer. This can encompass user interfaces, print management software, and tools for configuring printer settings.

Key Features of Software Updates include:

- Driver Updates: Keeping printer drivers up to date ensures compatibility with the latest operating systems and applications, improving overall performance and reliability.
- User Interface Improvements: Software updates may enhance the user interface, making it more intuitive or adding features for easier access to printer settings.
- Compatibility: Updates may address compatibility issues with new operating systems, applications, or network configurations.
- Enhanced Features: Software updates can introduce new features such as improved print management tools, mobile printing capabilities, and support for cloud printing services.

9.8.2 Regular Security Patching and Maintenance

1) Importance of Regular Security Patching
   a) Vulnerability Mitigation: Printers can be susceptible to various vulnerabilities, including those that allow unauthorized access or data breaches. Regular patching helps address these vulnerabilities before they can be exploited.
   b) Data Protection: Printers often handle confidential documents. Keeping firmware and software up to date reduces the risk of data leaks and protects sensitive information.
   c) Compliance: Many industries have regulatory requirements regarding data security. Regular patching helps organizations comply with these regulations, reducing the risk of fines or legal issues.
   d) Enhanced Features: Security patches may also include updates that enhance the functionality of printers, improving overall user experience and efficiency.

2) Maintenance Tasks

In addition to patching, regular maintenance of printers is essential for their optimal performance. Key maintenance tasks include:

   a) Firmware Updates: Regularly check for and apply firmware updates provided by the manufacturer to fix known security vulnerabilities and enhance performance.
   b) Software Updates: Ensure that printer drivers and associated software are updated regularly to maintain compatibility with operating systems and applications.
   c) Network Security Configuration:
      i) Change default passwords and usernames to enhance security.
      ii) Disable unused protocols and ports (e.g., Telnet, FTP) to reduce the attack surface.
      iii) Implement strong access controls to restrict who can access the printer and its settings.
   d) Physical Security Measures
      i) Place printers in secure locations to prevent unauthorized physical access.
      ii) Implement user authentication (e.g., PIN codes, swipe cards) for accessing the printer.

e) Regular Audits and Monitoring: Conduct periodic security audits to assess the printer's configuration and security posture. Monitor printer logs for suspicious activities.

9.8.3 Configuring Printer Security Policies

Configuring printer security policies is essential for protecting sensitive information, preventing unauthorized access, and ensuring secure printing practices within an organization. Here are the key steps and considerations for establishing effective printer security policies:

1) Assessing Security Needs
   a) Identify Sensitive Information: Understand what types of documents and data will be printed and the associated risks if that information is compromised.
   b) Evaluate Current Security Posture: Conduct a security audit to identify vulnerabilities in existing printer configurations and user practices.
2) Defining User Access Controls
   a) User Authentication: Implement user authentication methods such as PIN codes, swipe cards, or secure print release systems to control access to printers. Consider role-based access controls to limit access to sensitive printing features based on user roles.
   b) Group Policies: Use group policies to define who can access specific printers or print jobs based on departmental needs and security clearance levels.
   c) Guest Access Management: Set clear policies for guest access to printers, restricting their ability to print sensitive documents.

3) Configuring Network Security
   a) Change Default Credentials: Change the default username and password for printer management interfaces to prevent unauthorized access.
   b) Secure Network Connections: Ensure that printers are connected to secure networks. Use VLANs to separate printer traffic from sensitive internal networks.
   c) Enable encryption protocols (e.g., IPsec, SSL/TLS) for secure data transmission.
   d) Disable Unused Protocols: Disable protocols such as Telnet or FTP that are not in use to reduce potential attack vectors.

4) Implementing Print Job Security
   a) Secure Print Release: Use secure print solutions that require users to authenticate at the printer before their documents are printed, minimizing the risk of sensitive documents being left unattended.
   b) Data Encryption: Ensure that data sent to printers is encrypted, protecting it from interception during transmission.
   c) Audit Logging: Enable logging of print jobs to monitor usage and identify any suspicious activity. Review logs regularly to detect potential security breaches.

9.8.4 Enforcing Print Job Authentication and Encryption

Print job authentication requires users to verify their identity before their documents are printed. This prevents unauthorized access to printed materials and reduces the risk of sensitive information being left unattended.

Methods of Authentication include:

- User ID and Password: Require users to log in to a secure print system using their credentials before they can send print jobs.
- PIN Code Authentication: Users enter a unique PIN at the printer to release their print jobs. This method is simple and effective for managing access.
- Card-Based Authentication: Implement card swipe systems where users can swipe an ID card or employee badge at the printer to authenticate and release print jobs.
- Mobile Authentication: Use mobile apps that allow users to authenticate their identity through their smartphones, enabling secure access to printers.
- Biometric Authentication: Incorporate biometric systems, such as fingerprint or facial recognition, for high-security environments where sensitive information is printed.

9.8.5 Device Health Monitoring and Maintenance

Device health monitoring and maintenance for printers are crucial for ensuring optimal performance, extending the lifespan of printers, and minimizing downtime. Effective monitoring allows organizations to proactively address issues before they escalate into significant problems.

1) Importance of Device Health Monitoring
   a) Early Problem Detection: Continuous monitoring helps identify issues such as low ink or toner levels, paper jams, or hardware malfunctions before they lead to operational disruptions.
   b) Performance Optimization: Monitoring device performance metrics can help optimize print quality and speed, ensuring that printers operate at their best.
   c) Cost Management: By tracking usage and performance, organizations can manage printing costs more effectively and avoid unexpected expenses related to repairs or replacements.
   d) Enhanced Security: Monitoring can also help detect unauthorized access or unusual printing patterns that may indicate security breaches.

2) Key Metrics to Monitor
   a) Print Volume: Track the number of pages printed to identify usage patterns and forecast maintenance needs.
   b) Toner and Ink Levels: Monitor remaining toner or ink levels to avoid interruptions due to empty cartridges.
   c) Error Rates: Keep track of error messages or failed print jobs to identify recurring issues that need addressing.
   d) Device Temperature: Overheating can lead to performance issues; monitoring temperature can help prevent overheating-related failures.
   e) Network Connectivity: Ensure that printers maintain a stable connection to the network for reliable performance.
3) Tools for Device Health Monitoring
   a) Printer Management Software: Use centralized printer management solutions (e.g., Papercut, Printer Logic, or UniPrint) to monitor and manage printer health across the organization. These tools provide insights into printer usage, status, and alerts for potential issues.
   b) Simple Network Management Protocol (SNMP): Implement SNMP to collect data from networked printers, enabling real-time monitoring of device status and performance.

c) Remote Monitoring Solutions: Utilize remote monitoring tools that allow IT teams to track printer health and status from a centralized dashboard, reducing the need for on-site visits.

9.9 PRINTER LOCAL ADMINISTRATOR ACCOUNTS AND PASSWORDS

9.9.1 Managing Printer Administrator Accounts

Managing printer administrator accounts is crucial for maintaining the security, accessibility, and efficiency of printing environments within an organization. Proper management of these accounts helps prevent unauthorized access, ensures compliance with security policies, and enables effective oversight of printer operations.

1) Establishing Administrative Roles
   a) Define Roles and Responsibilities: Clearly define roles for printer administrators, including those who have the authority to make changes to printer settings, manage user access, and perform maintenance tasks.
   b) Role-Based Access Control (RBAC): Implement RBAC to assign different levels of access to administrator accounts based on job responsibilities. For example, some users may only need basic management access, while others may require full administrative privileges.
2) Secure Authentication Practices
   a) Strong Password Policies: Enforce strong password policies that require complex passwords, regular password changes, and unique credentials for each administrator account.
   b) Multi-Factor Authentication (MFA): Implement MFA for printer administrator accounts to add a layer of security. This could involve requiring a second form of verification, such as a mobile authentication app or a one-time passcode.
3) Limiting Access:
   a) Minimize Administrator Accounts: Limit the number of accounts with administrative access to reduce the risk of unauthorized changes or security breaches. Only essential personnel should have administrative rights.
   b) Access Control Lists (ACLs): Use ACLs to specify which users or groups have administrative privileges for specific printers or print management systems.

9.9.2 Configuring Secure Password Policies

Configuring secure password policies for printers is crucial for safeguarding sensitive information and maintaining overall security. (Please refer requirements given in section 3.4

1) Monitoring and Logging:
   a) Change Logs: Record password changes and access attempts.
   b) Alerts for Failed Attempts: Set alerts for multiple failed login attempts.
2) Secure Default Credentials:
   a) Change Default Passwords: Update default passwords upon installation to prevent unauthorized access.

9.9.3 Disabling Unnecessary Accounts and Services

Disabling unnecessary accounts and services for printers is a critical security measure to minimize vulnerabilities and reduce the risk of unauthorized access in an organization's printing environment. By ensuring that only essential accounts and services are active, organizations can enhance their overall security posture.

1) Importance of Unnecessary Accounts and Services:
   a) Reduce Attack Surface: Limiting active accounts and services decreases the number of potential entry points for attackers, reducing the risk of exploitation.
   b) Minimize Misconfiguration Risks: Fewer active accounts and services mean less chance of misconfiguration that could lead to security vulnerabilities.
   c) Enhance Accountability: By having fewer active accounts, it becomes easier to track and audit user activities, improving accountability.
2) Identifying Unnecessary Accounts:
   a) Default Accounts: Many printers come with default accounts (e.g., admin, guest) that may not be needed. These should be disabled or renamed to enhance security.
   b) Inactive Accounts: Review user accounts periodically to identify and disable any accounts that are no longer in use, such as those belonging to former employees or users who have changed roles.
   c) Temporary Accounts: Disable accounts that were created for temporary access or projects that have been completed.
3) Disabling Unused Services:
   a) Evaluate Services: Identify and evaluate the services running on the printer (e.g., FTP, Telnet, SNMP). Disable any services that are not necessary for the printer's operation.

   b) Common Services to Disable:

      i) Telnet: An insecure protocol that should be disabled in favor of more secure options like SSH.

      ii) FTP: If not needed, disable FTP access, especially if sensitive documents are being printed.

      iii) SNMP: Disable SNMP if not actively used for monitoring or secure it with proper community strings and access controls.

      iv) Web Interfaces: If the printer's web management interface is not required, consider disabling it or securing it behind a firewall.

4) Implementing Access Controls:
   a) Limit Administrative Access: Only allow essential personnel to have administrative access to printers. Use role-based access control (RBAC) to manage permissions effectively.
   b) Strong Password Policies: Ensure that accounts that remain active have strong, complex passwords and are subject to regular password changes.
5) Monitoring and Auditing:
   a) Regular Audits: Conduct regular audits of active accounts and services to ensure that only necessary ones are enabled. Document the reasons for disabling any accounts or services.
   b) Log Monitoring: Enable logging for account access and service usage. Monitor logs for any unauthorized access attempts or unusual activities.

9.9.4 Monitoring and Managing Administrator Access

- Enable Logging: Enable logging of all administrative actions taken on printers. This includes changes to settings, user access modifications, and service configurations.

- Audit Logs: Regularly review audit logs to monitor administrative activities. Look for unusual patterns or unauthorized changes that may indicate a security issue.
- Real-Time Alerts: Configure real-time alerts for specific actions, such as failed login attempts or changes to critical settings, to facilitate timely responses to potential security threats.

9.10 INFORMATION RIGHTS MANAGEMENT (IRM)

Information Rights Management (IRM) is a technology and set of practices that protect sensitive information from unauthorized access, use, or distribution. It is often used in corporate and organizational environments to secure confidential data such as documents, emails, and intellectual property. IRM ensures that only authorized individuals can view, edit, share, or print specific content, and it often includes advanced control mechanisms to track and revoke access when needed.

9.10.1 Security Controls of IRM

- Access Control: Restricts who can access a document and define their level of interaction (e.g., view-only, edit, print).
- Encryption: Ensures that data is secure both in transit and at rest.
- Access Validity: Allows content owners to set time limits on access to specific files or information.
- Dynamic Permissions: Enables modification of access rights even after the file has been distributed.
- Labeling: Adds identifying marks to documents to discourage unauthorized sharing or help trace leaks.
- Tracking and Auditing: Monitors who accessed the content and what actions were performed, providing an audit trail.

9.11 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Email Security	Define email security policies and phishing awareness	Identify email security risks (phishing, spoofing)	Implement SPF, DKIM, DMARC, secure gateways	Monitor email threats and detect phishing	Quarantine emails and alert users	Recover from email-based attacks and revoke compromised accounts

Table 11: Email Security NIST Framework mapping

10. CLOUD SECURITY

10. Cloud Security

10.1 INTRODUCTION

Cloud computing has transformed IT infrastructure, enabling organizations to leverage managed services to achieve operational efficiency, scalability, and flexibility. This shift reduces the burden of managing physical hardware and associated maintenance tasks, such as patching, backups, and monitoring, by utilizing the expertise of cloud providers.

Cloud Deployment Models

1. **Private Cloud:** Infrastructure maintained by a single organization, offering full control and compliance, suitable for highly sensitive workloads.
2. **Public Cloud:** Infrastructure provided by service providers like AWS, Microsoft Azure, and Google Cloud Platform (GCP), ideal for non-sensitive workloads (e.g. Internet Facing Public Websites) requiring scalability and cost efficiency.
3. **Hybrid Cloud:** Combines private and public cloud infrastructures to achieve a balance between security, scalability, and operational flexibility.
4. **Multi-Cloud:** Involves the use of multiple cloud providers to mitigate vendor lock-in, enhance redundancy, and optimize resource allocation.

Use Cases for Public Cloud: Public cloud environments are suitable for:

- Development and testing environments.
- Collaboration tools and web hosting.
- Non-sensitive workloads where scalability and cost-effectiveness are priorities.

For critical workloads or regulated data, private or hybrid cloud models are recommended to ensure enhanced control, compliance, and security.

Cloud Services

To effectively secure cloud environments, it is essential to understand the four primary service models: On-Premises, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Each model offers a different level of abstraction, control, and division of responsibility between the organization and the service provider.

- On-Premises: This traditional model involves deploying and managing the entire IT stack in-house. The organization is fully responsible for everything—from physical infrastructure and networking to operating systems, applications, and data security. While this model offers maximum control, it also requires substantial resources for maintenance, upgrades, and security.
- Infrastructure as a Service (IaaS): In this model, the cloud provider delivers basic compute, storage, and network infrastructure. The customer is responsible for installing and managing the operating systems, applications, data, and related security controls. This model offers flexibility and scalability while offloading physical infrastructure management to the provider.

- Platform as a Service (PaaS): PaaS provides a managed application platform, including the operating system, runtime environment, and middleware. The customer only manages their applications and data, allowing development teams to focus on coding and deployment without worrying about infrastructure maintenance or patching.
- **Software as a Service (SaaS):** SaaS offers complete, ready-to-use applications hosted and managed by the provider. Users access the service via web interfaces or APIs. The provider manages the entire stack, including security, availability, and updates. The customer's primary responsibility is user access control and proper usage of the application.

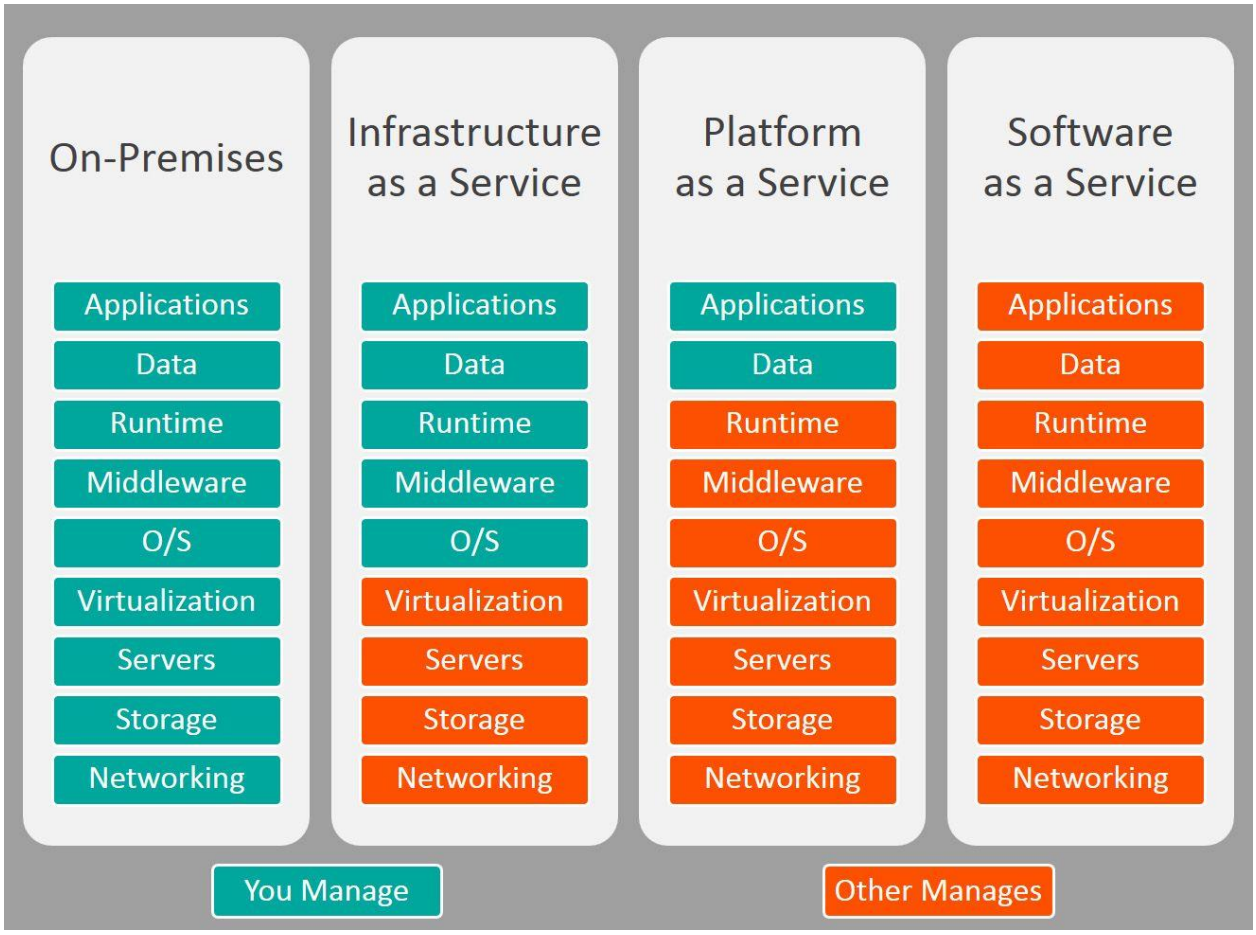

Figure 12: Cloud Services (Source: RedHat)

10.2 FOUNDATIONAL SECURITY PRACTICES

The principles of **Authentication, Authorization, and Accounting (AAA)**, **Zero Trust Security**, and **Identity and Access Management (IAM)** are essential for cloud security. These foundational practices are discussed in **Chapter 3: Foundational Security Practices** and should be applied across all cloud deployment models to maintain robust access control and ensure accountability.

10.3 STORAGE AND DATABASE SECURITY

Effective storage and database security practices, as detailed in **Chapter 5: Storage Security** and **Chapter 6: Database Security**, are critical for protecting sensitive information in cloud environments. Best practices include:

- **Encryption:** Enable encryption at rest and in transit using tools such as AWS EBS encryption, Azure Disk Encryption, and GCP Persistent Disk Encryption.
- **Access Control:** Implement fine-grained IAM roles to restrict database access to authorized users.
- **Backups and Recovery:** Regularly create automated snapshots and configure geo-redundancy to ensure data availability.
- **Provider-Specific Tools:** Leverage security features like AWS RDS Security Groups, Azure Defender for SQL and GCP Cloud SQL IAM for database and storage protection.

These practices ensure data integrity, confidentiality, and availability, irrespective of the cloud provider.

10.4 ENCRYPTION

Encryption remains a cornerstone of cloud security and is thoroughly discussed in **Chapter 3.7: Encryption and Data Security**. Key recommendations for cloud environments include:

- Use **TLS 1.3 or higher** for securing communications.
- Enable encryption for all storage volumes and backups (e.g., AWS S3, Azure Blob Storage, GCP Cloud Storage).
- Centralize key management with tools like **AWS KMS**, **Azure Key Vault**, or **GCP Cloud KMS**.
- Implement key rotation policies to minimize the impact of compromised keys.

Adhering to these encryption standards ensures compliance with industry regulations and protects sensitive data from unauthorized access.

10.5 NETWORK AND LOGICAL SEGMENTATION

Strong network security practices are critical for securing cloud environments. Details in **Chapter 4.1: Network Architecture and Design** provide guidance on network segmentation, which applies universally across AWS, Azure, and GCP. Best practices include:

- **Restrict Access:** Ensure no security groups or firewall rules allow unrestricted ingress (e.g., from 0.0.0.0/0 or: :/0) to remote administration ports.
- **Least Privilege Routing:** Configure VPC routing tables with least-access policies.
- **Provider-Specific Controls:** Use tools like AWS Security Groups, Azure Network Security Groups, and GCP VPC Firewall Rules to enforce logical segmentation and restrict network traffic.

10.6 LOGGING AND MONITORING

Comprehensive logging and monitoring are essential for maintaining visibility and detecting anomalies in cloud environments. As described in **Chapter 3.5: Logging, Monitoring, and Audit Framework**, these practices ensure accountability and support compliance. Key recommendations include:

- Enable logging across all regions and services, such as AWS CloudTrail, Azure Monitor, and GCP Cloud Logging.

- Use encryption for log storage to protect sensitive data.
- Regularly review logs and configure alerts for suspicious activity.

10.7 DISASTER RECOVERY

Effective disaster recovery ensures business continuity in the event of failure. Detailed practices are discussed in **Chapter 3.6: Backup, Retention, and Disaster Recovery**. Specific to cloud environments:

- Use tools like AWS Backup, Azure Site Recovery, or GCP Backup and DR Service.
- Regularly test recovery plans to validate their effectiveness.
- Implement distributed data processing and geo-redundant storage for critical workloads.

10.8 IMPLEMENTATION AND GOVERNANCE

Governance frameworks ensure security policies are consistently applied across cloud environments. As detailed in **Chapter 3.8: Vulnerability, Compliance Management, and Governance**, these include:

- Enforcing role-based access control (RBAC).
- Conducting security awareness training for all users.
- Aligning practices with recognized standards like **NIST SP 800-53**, **CIS Benchmarks** and **OEM Best Practices** for multi-cloud applicability.

While much of this guidance aligns with NIST SP 800-53 and CIS Benchmarks, specific configurations are included for AWS, Azure, and GCP etc. to ensure practical applicability across platforms.

10.9 CLOUD SECURITY BEST PRACTICES

Security Domain	Purpose	Implementation Guidance
Shared Responsibility Model	Defines who manages which security controls	Understand and document roles for IaaS, PaaS, and SaaS; clearly separate responsibilities between provider and customer; maintain a shared responsibility matrix.
Identity & Access Management (IAM)	Controls who can access cloud resources and what they can do	Enforce least privilege, enable MFA, use RBAC/ABAC models, implement identity federation (e.g., with SSO), and audit IAM policies regularly.
Encryption (at Rest & Transit)	Protects data confidentiality from unauthorized access	Use AES-256 for data at rest, TLS 1.2+ for data in transit; enforce encryption across all services; manage keys using KMS or HSM with proper rotation and access logging.
Network Segmentation	Limits lateral movement and isolates workloads	Use virtual networks (VPCs/VNets), subnetting, security groups, private endpoints, and restrict inbound/outbound traffic; apply least-access routing and zero trust segmentation.

Logging & Monitoring	Enables detection of misconfigurations, intrusions, or policy violations	Enable audit logging, access logs, and system logs; centralize logs in SIEM/SOAR; create alerts for anomalies; retain logs for compliance and forensic investigations.
Backup & Disaster Recovery	Ensures continuity and fast recovery during failure or breach	Implement automated backups with geo-redundancy; define RTO/RPO; periodically test recovery plans; encrypt and restrict access to backup storage.
Cloud Posture Management (CSPM)	Identifies and remediates misconfigurations	Use CSPM tools to benchmark against CIS/NIST; continuously scan for policy violations; enforce corrective workflows.
API Security	Secures cloud-native service interactions	Use OAuth2.0 or API keys with rate-limiting; validate inputs; encrypt API traffic; apply WAF for inspection and anomaly detection; monitor API access patterns.
Data Classification & Governance	Applies appropriate protections based on data sensitivity	Define sensitivity levels (e.g., public, internal, restricted); enforce controls per classification; train users and apply DLP policies for sensitive data.
Compliance Alignment	Ensures adherence to internal and external regulatory requirements	Align controls with ISO/IEC 27017, NIST SP 800-144, ISO/IEC 27018, and sector-specific laws; conduct regular assessments and provider due diligence.

Table 12: Cloud Security Measures: Best Practices to Follow

10.10 NIST CYBER SECURITY FRAMEWORK 2.0 MAPPING

Security Domain	Govern	Identify	Protect	Detect	Respond	Recover
Cloud Security	Define cloud security policies and compliance requirements	Maintain cloud asset inventory and classify data	Implement IAM, encryption, secure configurations	Monitor cloud activity and detect unauthorized access	Respond to cloud breaches and revoke access	Restore cloud workloads and ensure service continuity

Table 13: Cloud Security NIST Framework mapping

(Reference: NIST SP 500-291: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-291r2.pdf, NIST SP 800-210: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-210.pdf, CIS Microsoft Azure Foundations Benchmark v3.0.0 , CIS Google Cloud Platform Foundation Benchmark v3.0.0, CIS Amazon Web Services Foundations Benchmark v4.0.1)

11. APPLICATION SECURITY AND ACCESS CONTROLS

11. Application Security and Access Controls

As organizations increasingly depend on internet-based applications, securing access to these tools is essential. Without proper controls, unrestricted internet access and unauthorized application usage can lead to data leakage, insider threats, and compliance violations. This chapter outlines best practices for restricting internet access, managing application usage, and implementing robust security controls to safeguard corporate data.

11.1 CONTROLLING INTERNET ACCESS

a. Restricting Uncontrolled Internet Usage

To minimize security risks, organizations should implement the following measures:

1. Deploy Secure Web Gateways (SWG) such as Cisco Umbrella, Zscaler, or Forcepoint to filter and monitor internet traffic.
2. Enforce proxy-based internet access to route all traffic through controlled gateways with SSL/TLS inspection.
3. Apply role-based internet access policies (RBAC) to limit exposure based on job function.
4. Block access to unauthorized web applications, including personal email and unapproved cloud storage services.

b. Network-Level Restrictions

1. Use Next-Generation Firewalls (NGFWs) to enforce domain filtering and deep packet inspection.
2. Implement DNS Security to prevent connections to phishing or command-and-control (C2) servers.
3. Isolate critical infrastructure in segmented networks to eliminate direct internet exposure.
4. Adopt Zero Trust Network Architecture (ZTNA) to enforce strict access controls.

11.2 APPLICATION WHITELISTING AND USAGE CONTROL

a. Establishing an Approved Application List

To regulate software usage, organizations should:

1. Define an explicit list of authorized applications based on security and business needs.

2. Use Application Control Policies within Microsoft Defender, CrowdStrike, or Carbon Black to enforce execution rules.

3. If a new application or software is being considered, a Software Risk Assessment, along with Vulnerability Assessment (VA) and Penetration Testing (PT), must be conducted before granting approval.

4. Regularly review and update the whitelist to phase out outdated or vulnerable applications.

b. Enforcing Usage Restrictions

1. Use Windows AppLocker or Microsoft Intune to block execution of unapproved applications.

2. Leverage Endpoint Protection Platforms (EPP) to monitor and restrict software installation.

3. Configure Group Policy Objects (GPOs) or Mobile Device Management (MDM) to enforce restrictions across enterprise devices.

c. Monitoring and Auditing Application Usage

1. Integrate monitoring with SIEM platforms (Splunk, ELK, QRadar) to track application usage patterns.
2. Enable application logging and alerting for unauthorized access attempts.
3. Conduct regular security audits to detect and address policy violations.

11.3 SECURING AUTHORIZED APPLICATIONS

a. Preventing Data Leakage in Business Applications

To prevent unauthorized data sharing:

1. Disable chat and file transfer features in messaging platforms such as LinkedIn, WhatsApp, and Telegram.
2. Implement Data Loss Prevention (DLP) policies to block unauthorized sharing of sensitive files.
3. Restrict access to unapproved cloud storage services, ensuring employees use corporate-managed platforms (e.g., OneDrive for Business, Google Workspace Drive).

b. Implementing Data Loss Prevention (DLP)

1. Monitor and block unauthorized file transfers using solutions like Microsoft Purview, Symantec DLP, or Forcepoint DLP.
2. Deploy content inspection tools to detect and prevent transmission of sensitive information.
3. Enforce encryption policies for email communication via PGP, S/MIME, and end-to-end encryption (E2EE).

c. Role-Based Access Controls (RBAC) for Applications

1. Assign minimum access privileges to users based on their role.
2. Require Multi-Factor Authentication (MFA) for accessing sensitive applications.
3. Implement Just-In-Time (JIT) access to minimized standing privileges for administrators.

Effective management of internet access, application whitelisting, and security controls is critical for preventing unauthorized data exposure. By combining network restrictions, application security policies, and compliance frameworks, organizations can reduce risk and strengthen their overall cybersecurity posture. Regular reviews and updates of these controls are necessary to address evolving threats and regulatory requirements.

(References: NIST SP 800-53: Security and Privacy Controls for Federal Information Systems and Organizations, NIST SP 800-171 Rev. 2)

12. THE ROAD AHEAD: SECURITY TRENDS FOR NETWORK, CLOUD, AND BEYOND

12. The Road Ahead: Security Trends for Network, Cloud, and Beyond

The chapter explores the future of infrastructure security, emphasizing the integration of advanced technologies and adaptive strategies. Key trends include AI-powered threat detection, Zero Trust architectures, and unified frameworks like SASE for securing distributed environments. It highlights the importance of quantum-resistant cryptography and proactive measures for addressing emerging challenges in IoT, 5G, and compliance. Emphasis is placed on automation and collaboration through platforms like SOAR and threat intelligence sharing. Together, these trends ensure resilience and readiness against evolving cybersecurity threats.

12.1 NETWORK SECURITY TRENDS

As organizations face an increasingly sophisticated threat landscape, network security is evolving to integrate advanced technologies, innovative frameworks, and dynamic approaches. Below are the five key trends shaping the future of network security.

1. Software-Defined Architectures and Zero Trust

- **Software-Defined Networking (SDN):** SDN introduces agility by decoupling control and data planes, enabling centralized policy enforcement and dynamic network segmentation. This enhances scalability and provides granular control over traffic flows.
    - **Example**: Organizations use SDN to isolate workloads dynamically and enforce security policies based on real-time traffic analysis.
- **Zero Trust Network Access (ZTNA):** Complementing SDN, the Zero Trust model enforces "never trust, always verify" principles by authenticating and authorizing every user, device, and application, regardless of their location.
    - **Impact**: Together, SDN and Zero Trust minimize lateral movement, ensuring secure access and reducing exposure to internal and external threats.

2. AI-Powered Threat Detection and Automation

- **Artificial Intelligence (AI) and Machine Learning (ML)**
    - AI enhances real-time threat detection by identifying anomalies, such as unusual login patterns or traffic spikes, that traditional systems might miss.
    - ML models continuously refine themselves to detect evolving threats like Advanced Persistent Threats (APTs) and insider attacks.
- **Behavioral Analytics:** By monitoring user and device behavior, AI-driven systems identify deviations that could indicate compromised accounts or malicious activity.
- **Automation with SOAR (Security Orchestration, Automation, and Response):** SOAR platforms automate routine tasks like blocking IP addresses, enabling rapid mitigation of threats and freeing up resources for strategic initiatives.

3. Advanced Encryption and Quantum Resilience

- **Quantum-Resistant Cryptography:** The impending rise of quantum computing necessitates transitioning from traditional encryption (e.g., RSA, ECC) to quantum-resistant algorithms such as lattice-based cryptography. Hybrid cryptographic methods are emerging as transitional solutions.
    - **Significance**: Proactive adoption ensures the long-term security of sensitive communications and data.
- **Secure DNS and Traffic Encryption:** Technologies like DNSSEC (Domain Name System Security Extensions) safeguard against DNS hijacking, while Transport Layer Security (TLS) ensures encrypted traffic flows, mitigating risks of interception and tampering.

4. Integrated Frameworks for Distributed Security

- **Secure Access Service Edge (SASE):** Combining SD-WAN with security services (e.g., Secure Web Gateways, Cloud Access Security Brokers), SASE enables unified security for hybrid and distributed networks.
    - **Example**: Enterprises with remote workforces leverage SASE to securely connect users to cloud resources while enforcing consistent security policies.
- **Micro-Segmentation:** Beyond traditional VLANs, micro-segmentation isolates workloads at granular levels using software-defined perimeters and identity-based segmentation.
    - **Impact**: Reduces the attack surface and limits lateral movement within networks.
- **IoT and OT Integration:** The convergence of IT, IoT, and operational technology (OT) environments demand enhanced security measures like device profiling, network isolation, and real-time monitoring.
    - **Example**: ISA/IEC 62443-compliant frameworks address unique security needs in industrial IoT.

5. Resilience Against Evolving Threats

- **5G and DDoS Mitigation**
    - The high speed and low latency of 5G networks require robust security measures such as network slicing and enhanced endpoint authentication to prevent exploitation.
    - AI-powered DDoS mitigation services dynamically filter malicious traffic, ensuring service availability during volumetric and application-layer attacks.
- **Threat Intelligence Sharing:** Collaborative platforms enable organizations to share threat insights, improving collective defences and enabling faster responses to emerging risks.
- **Compliance-Driven Security:** With increasing regulatory mandates (e.g., GDPR, DPDP Act), organizations are adopting real-time compliance monitoring and reporting tools to ensure adherence and secure sensitive data.

12.2 SERVER, STORAGE, AND DATABASE SECURITY TRENDS

The foundation of modern IT infrastructure lies in servers, storage systems, and databases. As cyber threats evolve and regulatory demands increase, securing these critical components is essential. Below are the key trends shaping the future of server, storage, and database security, grouped into related topics for clarity.

1. **Zero Trust for Server, Storage, and Database Access**
    a. Continuous Authentication and Authorization
        i. Adopting Zero Trust principles ensures that only verified and authorized users, devices, and applications can access servers, storage systems, or databases. Policies are enforced dynamically based on real-time risk assessments.
        ii. Impact: Reduces unauthorized access and mitigates lateral movement in case of compromise.
    b. Just-in-Time (JIT) Access
        i. Temporary privilege escalation allows users or applications to access critical resources only when needed, minimizing persistent access risks.
2. **Advanced Threat Detection with AI and Behavioral Analytics**
    a. AI-Driven Monitoring
        i. AI technologies are being integrated into server and database environments to detect abnormal behaviors such as unusual query patterns, unauthorized privilege escalations, or suspicious file access.
        ii. Example: Database Activity Monitoring (DAM) tools like Oracle Audit Vault identify insider threats and external attacks in real time.
    b. Behavioral Analytics
        i. Servers and storage systems are monitored for deviations in normal operational behavior, such as unexpected spikes in CPU usage or unusual data transfer activities.
3. **Encryption Across Data States**
    a. Data at Rest
        i. Full-disk encryption and database-level encryption ensures sensitive data remains protected, even if physical drives or storage devices are stolen.
        ii. Example: Transparent Data Encryption (TDE) is widely used in databases like Oracle and SQL Server.
    b. Data in Transit
        i. Secure protocols like TLS and IPSec protect data exchanged between servers, storage systems, and clients from interception or tampering.
    c. Data in Use
        i. Confidential computing technologies protect data during processing using Trusted Execution Environments (TEEs), ensuring even administrators cannot access sensitive workloads.
        ii. Examples: Intel SGX, AMD SEV and Azure Confidential Computing.
4. **Ransomware Defence for Storage and Databases**
    a. Immutable Backups and Snapshots
        i. Storage systems now feature immutable backups and write-once-read-many (WORM) snapshots that protect against ransomware attacks.

ii. Example: Solutions like NetApp Snapshot or AWS Backup Vault Lock ensure that backups cannot be altered or deleted by attackers.

b. Automated Recovery Mechanisms
   i. Modern systems offer automated rollback features to restore storage volumes or databases to a pre-attack state, minimizing downtime.
   ii. Compliance and Auditing Automation

c. Automated Compliance Checks
   i. Regulatory frameworks such as GDPR, PCI DSS, and India's DPDP Act require strict adherence to security practices. Automation tools continuously monitor servers, storage, and databases for compliance violations.
   ii. Example: Platforms like AWS Config or Azure Policy provide real-time auditing and reporting.

d. Audit Trails
   i. Robust logging and monitoring systems create immutable audit trails for databases and storage, facilitating forensic investigations and compliance requirements.

5. Storage Security Integration with Cloud and Edge

a. Cloud-Based Storage Security
   i. Cloud providers are enhancing storage security with features like encrypted object storage, secure access controls, and real-time threat detection.
   ii. Example: AWS S3 Block Public Access prevents accidental exposure of storage buckets.

b. Edge Computing Storage Security
   i. As edge computing grows, securing localized storage closer to data sources is critical. This includes encrypting data at edge nodes and securely syncing it with central systems.

6. Database Security Innovations

a. Database Activity Monitoring (DAM)
   i. DAM tools monitor and analyze all activities within a database, such as queries, privilege escalations, and schema changes, to detect malicious behavior.
   ii. Example: IBM Guardium and Imperva DAM provide real-time alerts for suspicious database activities.

b. Privacy-Enhancing Computation (PEC)
   i. Technologies like homomorphic encryption and secure multiparty computation allow sensitive data to be processed securely without exposing it to unauthorized access.
   ii. Resilience Against Insider Threats

c. Role-Based Access Control (RBAC)
   i. Restricting access to resources based on roles ensures that users and applications only have permissions necessary for their functions.
   ii. Example: Databases implement RBAC to prevent unauthorized access to sensitive tables or schemas.

d. Privileged Access Management (PAM)
    i. PAM solutions monitor and control administrative access to servers, databases, and storage systems, ensuring accountability and reducing the risk of misuse.
    ii. Integration with Broader Security Ecosystems
e. Extended Detection and Response (XDR)
    i. Server, storage, and database telemetry are integrated into XDR platforms, enabling holistic threat detection and response.
        1. Benefit: Allows security teams to correlate server or database anomalies with network or endpoint activity for faster incident resolution.
f. SIEM and SOAR Integration
    i. Security Information and Event Management (SIEM) tools collect logs from servers, storage, and databases, while Security Orchestration, Automation, and Response (SOAR) platforms automate remediation workflows.

12.3 ENDPOINT SECURITY TRENDS

With the rise of remote work, BYOD (Bring Your Own Device) policies, and increasingly sophisticated attacks, endpoints have become critical targets for cyber threats. Modern endpoint security focuses on combining advanced technologies, Zero Trust principles, and real-time threat detection to mitigate risks. Below are five key trends shaping the future of endpoint security.

1) Unified Endpoint Management (UEM) and Zero Trust

a) Unified Endpoint Management (UEM):
    i) UEM platforms unify the management and security of endpoints across diverse operating systems, including Windows, macOS, Linux, Android, and iOS. This consolidation simplifies device provisioning, compliance enforcement, and patch management.
    ii) Example: Tools like Microsoft Intune and VMware Workspace ONE ensure centralized visibility and control over endpoints.
b) Zero Trust Endpoint Security:
    i) Zero Trust principles enforce continuous authentication and authorization for endpoint access, ensuring that only compliant and verified devices can interact with corporate resources.
    ii) Impact: Reduces the risk of compromised endpoints spreading threats laterally within the network.

2) Advanced Threat Detection with AI and Behavioral Analytics

a) AI-Powered Threat Detection
    i) AI and machine learning are integral to modern endpoint security solutions, identifying sophisticated attacks such as fileless malware and ransomware by analyzing behavior instead of relying solely on signatures.
    ii) Example: Endpoint Detection and Response (EDR) tools like CrowdStrike Falcon and Sentinel One proactively monitor endpoint activity to detect anomalies.
b) Behavioral Analytics
    i) Monitoring user and device behavior helps identify deviations that may indicate compromised endpoints, such as unusual file access patterns or geographic login anomalies.

ii) Benefit: Behavioral detection enhances the identification of insider threats and zero-day attacks.

3) Proactive Endpoint Hardening and Resilience

a) Firmware and BIOS Security
   i) As attackers target lower layers of the computing stack, securing firmware and BIOS has become critical. Features like self-healing BIOS and secure boot ensure endpoints remain tamper resistant.
   ii) Example: Technologies like HP Sure Start and Intel Boot Guard safeguard firmware integrity.

b) Immutable Infrastructure
   i) Endpoint security is shifting toward immutable configurations, where devices are reset to a known good state rather than manually remediating issues.
   ii) Impact: Prevents persistence of malware and ensures rapid recovery after incidents.
   iii) Endpoint Integration with Broader Security Ecosystems

c) Extended Detection and Response (XDR)
   i) XDR platforms integrate endpoint telemetry with data from other sources like network and email systems, enabling holistic threat detection and response.
   ii) Example: Palo Alto Cortex XDR correlates endpoint activity with broader network patterns to identify and respond to threats more effectively.

d) Secure Collaboration Tools
   i) With the growth of remote work, endpoint security extends to tools like video conferencing and shared workspaces, ensuring secure communications and data sharing.
   ii) Benefit: Prevents data leakage and ensures endpoint devices adhere to corporate security policies during collaboration.

4) Ransomware Defence and Data Protection

a) Ransomware-Resilient Endpoints:
   i) Endpoint solutions are enhancing resilience against ransomware by introducing features like automated rollback, which restores systems to pre-attack states without requiring backups.
   ii) Example: Solutions like Sophos Intercept X and Carbon Black include automated file recovery and exploit prevention.

b) Endpoint Data Loss Prevention (DLP)
   i) DLP solutions prevent unauthorized data exfiltration by monitoring sensitive file movements on endpoints and blocking high-risk activities.
   ii) Encryption and Secure Wipe: Full-disk encryption and remote wiping capabilities ensure endpoint data remains secure even in cases of theft or loss.
   iii) Use Case: Financial institutions use encryption to comply with regulatory standards while allowing secure remote work.

12.4 EMAIL SECURITY TRENDS

Email remains one of the most common vectors for cyberattacks, with threats such as phishing, ransomware, and business email compromise (BEC) evolving in complexity. Modern email security

focuses on leveraging AI, advanced threat intelligence, and robust policies to protect against these risks. Below are five key trends shaping the future of email security.

1) Advanced Threat Detection with AI and Behavioral Analytics

a) AI-Driven Phishing Detection

i) AI technologies are enhancing the detection of phishing emails by analyzing language patterns, sender behavior, and contextual anomalies. This enables identification of increasingly sophisticated attacks, including spear phishing and deepfake scams.

ii) Example: Tools like Microsoft Defender for Office 365 and Proofpoint identify and quarantine suspicious emails before they reach users.

b) Behavioral Anomaly Detection

i) By monitoring user behavior, email security solutions can detect compromised accounts and insider threats. Unusual login locations, unusual message volumes, or unauthorized data sharing trigger alerts for further investigation.

ii) Impact: Enhances protection against account takeovers and insider-driven data breaches.

2) Protection Against Ransomware and Malware

a) Attachment and URL Sandboxing

i) Modern email security platforms sandbox attachments and URLs to detect malicious payloads or phishing links before they are delivered to recipients.

ii) Example: Mimecast and Palo Alto Networks' email security solutions scan and emulate attachment behaviors to block threats.

b) Content Disarm and Reconstruction (CDR)

i) CDR technology removes potential malicious code from email attachments by sanitizing the file while retaining its original functionality.

ii) Benefit: Reduces the risk of zero-day exploits through seemingly legitimate attachments.

3) Email Data Loss Prevention (DLP) and Encryption

a) Data Loss Prevention (DLP)

i) Email DLP solutions monitor outbound communications to prevent unauthorized sharing of sensitive information, such as customer data or intellectual property.

ii) Example: Configuring rules in DLP systems to block emails containing unencrypted sensitive data or flagged keywords.

b) End-to-End Encryption

i) Encryption technologies, such as Secure/Multipurpose Internet Mail Extensions (S/MIME) or Pretty Good Privacy (PGP), ensure that email content remains confidential from sender to recipient.

ii) Use Case: Organizations in regulated industries use encryption to meet compliance requirements for secure communications.

4) Business Email Compromise (BEC) and Spoofing Defence

a) Domain-Based Message Authentication, Reporting, and Conformance (DMARC):

i) Email authentication protocols like DMARC, SPF (Sender Policy Framework), and DKIM (DomainKeys Identified Mail) protect against domain spoofing and email impersonation.

ii) Example: DMARC policies can be configured to reject spoofed emails, ensuring that only legitimate messages from an organization's domain are delivered.

b) AI-Powered BEC Prevention

i) AI systems analyze email content and context to detect subtle signs of BEC attacks, such as requests for wire transfers or sensitive data from impersonated executives.

ii) Impact: Provides proactive defence against high-value targeted attacks.

5) Integration with Broader Security Ecosystems

a) Secure Email Gateways (SEGs)

i) SEGs serve as the first line of defence by filtering incoming and outgoing emails for threats like spam, malware, and phishing.

ii) Example: Solutions like Barracuda Email Security Gateway integrate with broader security frameworks to provide layered protection.

b) Extended Detection and Response (XDR)

i) Email telemetry is integrated with endpoint, network, and cloud data in XDR platforms to provide holistic threat detection and response.

ii) Benefit: Enables correlation of email threats with broader attack patterns for more effective incident management.

12.5 CLOUD SECURITY TRENDS

As organizations increasingly adopt both public and private cloud environments, securing these infrastructures becomes paramount. The future of cloud security focuses on addressing unique challenges posed by scalability, dynamic workloads, and hybrid architecture. Below are five key trends shaping cloud security across public and private domains.

1) Cloud-Native Security and Containerization

a) Containers and Microservices Security

i) Cloud-native technologies like containers and Kubernetes demand robust security tools to safeguard workloads. These include runtime protection, vulnerability scanning, and image integrity checks.

ii) Example: Solutions like Aqua Security or Prisma Cloud monitor runtime behaviors to detect anomalies in containerized applications.

b) Serverless Computing Security

i) Serverless architectures (e.g., AWS Lambda, Azure Functions) are gaining popularity, necessitating security measures to prevent excessive permissions, injection attacks, and misconfigurations.

ii) Impact: Reduces attack surfaces while allowing developers to focus on innovation without compromising security.

2) Unified Security for Multi-Cloud and Hybrid Environments

a) Centralized Frameworks:

i) Organizations increasingly adopt multi-cloud and hybrid models, necessitating unified security solutions that provide visibility, policy enforcement, and threat detection across diverse platforms.

ii) Example: Platforms like VMware NSX and Palo Alto Prisma Cloud enable consistent security controls for workloads in both public and private clouds.

b) Zero Trust for Cloud:

i) Zero Trust models dynamically verify all access requests based on user identity, device posture, and behavior, providing strong defences against unauthorized access.

ii) Benefit: Mitigates risks associated with lateral movement across interconnected cloud environments.

3) Automated Security Posture and Threat Management

a) Cloud Security Posture Management (CSPM)

i) CSPM tools continuously monitor cloud environments to detect misconfigurations, non-compliance, and vulnerabilities, offering automated remediation.

ii) Example: CSPM tools like Wiz or Dome9 ensure compliance with frameworks such as GDPR, SOC 2, and PCI DSS.

b) Cloud Workload Protection Platforms (CWPP)

i) CWPP solutions focus on securing workloads (e.g., VMs, containers) through features like malware detection and file integrity monitoring.

c) AI-Powered Threat Detection

i) Machine learning models analyze cloud telemetry data to identify advanced threats, such as insider attacks and cryptojacking.

ii) Impact: Reduces time to detect and respond to evolving cloud-based threats.

4) Confidential Computing and Advanced Encryption

a) Data Encryption in Use

i) Confidential computing technologies protect data during processing by keeping it encrypted within Trusted Execution Environments (TEEs).

ii) Use Case: Intel SGX and AMD SEV enable secure computation of sensitive workloads in public clouds without exposing data to cloud providers.

b) Quantum-Resistant Cryptography

i) As quantum computing advances, organizations must adopt encryption mechanisms that can resist quantum attacks, such as lattice-based cryptography.

c) Secure DNS and Traffic Encryption

i) Technologies like DNSSEC and Transport Layer Security (TLS) ensure the integrity of communications, reducing risks from domain hijacking or eavesdropping.

5) Resilience, Compliance, and Governance

a) Compliance Automation:

i) Automated tools for real-time compliance checks (e.g., AWS Config, Azure Policy) streamline adherence to regulatory frameworks such as India's DPDP Act, ISO 27001, and PCI DSS.

ii) Policy as Code (PaC): Organizations codify governance policies to automate enforcement across cloud resources, ensuring consistency and reducing manual errors.

b) Ransomware Protection

i) Immutable backups, snapshot-based recovery, and advanced monitoring are critical in mitigating ransomware risks in both public and private clouds.

ii) Example: Public cloud providers like AWS offer ransomware-resilient storage solutions with features like versioning and point-in-time recovery.

c) Private Cloud Resilience

i) Private clouds prioritize data sovereignty and security for regulated industries, while edge computing integration enhances local data processing security.

13. ABBREVIATIONS

13. Abbreviations

AAA	Authentication, Authorization and Accounting
ABAC	Attribute-Based Access Control
AD	Active Directory
AES	Advanced Encryption Standard
AH	Authentication Header
AI	Artificial Intelligence
AIDE	Advanced Intrusion Detection Environment
AMD	Advanced Micro Devices
AP	Access Point
API	Application Programming Interface
ARP	Address Resolution Protocol
ASLR	Address Space Layout Randomization
ATP	Advanced Threat Protection
AWS	Amazon Web Services
BEC	Business Email Compromise
BIOS	Basic Input/Output System
BYOD	Bring Your Own Device
CAT	Common Attack Pattern
CCTV	Closed-Circuit Television
CDR	Content Disarm and Reconstruction
CEF	Cisco Express Forwarding
CIFS	Common Internet File System
CIS	Center For Internet Security
CISA	Cybersecurity And Infrastructure Security Agency
CPU	Central Processing Unit
CSF	Cybersecurity Framework
CSPM	Cloud Security Posture Management
CSRF	Cross Site Request Forgery
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWPP	Cloud Workload Protection Platform
DAM	Database Activity Monitoring
DEP	Data Execution Prevention
DKIM	DomainKeys Identified Mail
DLP	Data Loss Prevention
DMARC	Domain-Based Message Authentication, Reporting, And Conformance
DMZ	Demilitarized Zone
DNS	Domain Name System
DPDP	Digital Personal Data Protection
DR	Disaster Recovery
DSC	Digital Signature Certificate
DSS	Digital Signature Standard
EBS	Elastic Block Store
ECC	Elliptic Curve Cryptography

EDR	Endpoint Detection and Response
ELK	Elasticsearch
EMM	Enterprise Mobility Management
EOL	End-Of-Life
EPP	Endpoint Protection Platforms
ESP	Encapsulating Security Payload
FIM	File Integrity Monitoring
FIPS	Federal Information Processing Standards
FTK	Forensic Toolkit
FTP	File Transfer Protocol
GCP	Google Cloud Platform
GDPR	General Data Protection Regulation
GFS	Grandfather-Father-Son
GPG	Gnu Privacy Guard
GPS	Global Positioning System
GRUB	Grand Unified Bootloader
HA	High Availability
HIDS	Host-Based Intrusion Detection Systems
HP	Hewlett-Packard
HR	Human Resource
HSM	Hardware Security Modules
HTTP	Hypertext Transfer Protocol Secure
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity And Access Management
IBM	International Business Machines
ID	Identification
IDPS	Intrusion Detection and Prevention Systems
IDS	Intrusion Detection System
IEC	Compliance
IIS	Web Server Security Vendor
IP	Internet Protocol
IPS	Intrusion Prevention System
IRM	Information Rights Management
ISA	International Society of Automation
ISMS	Information Security Management Systems
ISO	International Organization for Standardization
ISP	Internet Service Provider
IT	Information Technology Security
IV	Initialization Vector
JIT	Just-In-Time
KMS	Cloud Provider VENDOR
KRACK	Key Reinstallation Attack
LAN	Local Area Network
LAPS	Local Administrator Password Solution
LDAP	Lightweight Directory Access Protocol
LTS	Ubuntu LTS

LUKS	Disk Encryption
MAC	Media Access Control
MDM	Mobile Device Management
MECM	Microsoft Endpoint Configuration Manager
MFA	Multi-Factor Authentication
MIME	Multipurpose Internet Mail Extensions
MITRE	MITRE Adversarial Tactics, Techniques, And Common Knowledge
ML	Machine Learning
MTTR	Mean Time To Remediate
NAC	Network Access Control
NAS	Network Attached Storage
NAT	Network Address Translation
NFS	Network File System
NGFW	Next Generation Firewall
NIST	National Institute of Standards And Technology
NSA	National Security Agency
NSX	Network Virtualization and Security Platform
NVD	National Vulnerability Database
Oauth	Open Authentication
OEM	Original Equipment Manufacturer
OS	Operating System
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OSSEC	Open Source HIDS Security
OT	Operational Technology
OTP	One Time Password
PAM	Privilege Access Management
PCI-DSS	Payment Card Industry Data Security Standard
PEC	Privacy-Enhancing Computation
PFS	Perfect Forward Secrecy
PGP	Pretty Good Privacy
PIN	Personal Identification Number
PSK	Pre-Shared Key
PT	Penetration Testing
QR	Quick Response
RBAC	Role-Based Access Control
RDS	Relational Database Service
RHEL	Red Hat Enterprise Linux
RMF	Risk Management Framework
RPO	Recovery Point Objective
RSA	Rivest–Shamir–Adleman
RTO	Recovery Time Objective
SAE	Simultaneous Authentication of Equals
SAN	Storage Area Networks
SASE	Secure Access Service Edge
SCAP	Security Content Automation Protocol

SCCM	System Center Configuration Manager
SCP	Secure Copy
SDLC	Software Development Life Cycle
SDN	Software-Defined Networking
SDS	Software-Defined Storage
SD-WAN	Software-Defined Wide Area Network
SEBI	Securities And Exchange Board of India
SEV	Secure Encrypted Virtualization
SFTP	Secure File Transfer Protocol
SGX	Software Guard Extensions
SHA	Secure Hash Algorithm
SIEM	Security Information and Event Management
SLA	Service Level Agreement
SMB	Server Message Block
SMS	Short Messaging Service
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAR	Security Orchestration, Automation, And Response
SOC	Security Operations Center
SP	Special Publication
SPF	Sender Policy Framework
SQL	Structured Query Language
SSH	Secure Socket Shell
SSID	Service Set Identifier
SSL	Secure Sockets Layer
SSO	Single Sign-On
SSPR	Self-Service Password Reset
SUSE	Software Und System-Entwicklung
SWG	Secure Web Gateways
TDE	Transparent Data Encryption
TKIP	Temporal Key Integrity Protocol
TLS	Transport Layer Security
TPM	Trusted Platform Module
TTY	Teletypewriter
UAT	User Acceptance Testing
UBA	User Behavior Analytics
UDP	User Datagram Protocol
UEFI	Unified Extensible Firmware Interface
UEM	Unified Endpoint Management
URL	Uniform Resource Locater
USB	Universal Serial Bus
UTM	Unified Threat Management
VA	Vulnerability Assessment
VLAN	Virtual Local Area Network
VPC	Virtual Private Cloud
VPN	Virtual Private Network

WAF	Web Application Firewall
WAN	Wide Area Network
WEF	Windows Event Forwarding
WEP	Wired Equivalent Privacy
WIDS	Wireless Intrusion Detection Systems
WIFI	Wireless Fidelity
WIPS	Wireless Intrusion Prevention Systems
WLAN	Wireless Local Area Network
WORM	Write Once, Read Many
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access 2
WPA3	Wi-Fi Protected Access 3
WSUS	Windows Server Update Services
XACML	Extensible Access Control Markup Language
XDR	Extended Detection and Response
XSS	Cross Site Scripting
ZTA	Zero Trust Architecture
ZTNA	Zero Trust Network Architecture
ZTS	Zero Trust Security

Application Security Standards and Best Practices

Executive Summary

The Application Security best application security practices document is a comprehensive blueprint for organizations to test, improve, and maintain the security of their web applications, APIs, and mobile applications. It provides clear guidance for developers, security teams, and stakeholders to ensure applications remain resilient against cyber threats while adhering to industry standards.

This document benefits the organizations by Providing:

- Reduce risk exposure: Proactively addressing security weaknesses.
- Improves development efficiency: Integrating security early in the application lifecycle.
- Builds customer trust: Strengthening the protection of sensitive data.
- Ensure compliance: Adhering to regulatory and industry standards

Key features of this document are:

- Comprehensive Coverage: It includes requirements for authentication, authorization, session management, Input validation, cryptography, and more.
- Industry adoption: Helps security professionals to assess the application security maturity and effectiveness.

1 . INTRODUCTION

Introduction

In today's world, application security is essential for protecting software from vulnerabilities and threats. As web applications, APIs, thick client applications, and mobile applications become increasingly integral to IT systems, securing them against potential attacks is critical. This document outlines best practices for testing application security controls, as well as the security controls within the environment that are relied upon to protect against injection attacks, broken authentication, security misconfiguration, logging vulnerabilities, encryption weaknesses, and similar issues. These best practices aim to establish confidence in the security of web applications, web services, mobile applications, and thick client applications.

These best practices were written based on OWASP, NIST CSF 2.0, and PCI DSS with the following objectives in mind:

1. Use as metric:

Provide application developers and application owners with best practices for assessing the level of trust that can be placed in their applications.

2. Use as guidance:

This document provides developers with a clear roadmap for building secure applications that adhere to required best practices, ensuring both the applications and their environments are well-protected against potential threats.

This document provides a proactive approach to cyber security that helps prevent threats at the initial stages rather than a reactive approach.

Adequate security procedures must be implemented from the initial stages of application development and maintained throughout the application's production lifecycle. These measures, such as automated security scanning, aim to provide a robust defence against evolving cyber threats. This document outlines practices designed to ensure maximum security resilience.

The chapters are arranged based on some of the NIST CSF 2.0 functions. This document does not cover the following two functions of NIST CSF2.0:

Respond (RS)

- Incident Management
- Incident Response Reporting
- Incident Response Reporting and Communication

Recovery (RC)

- Incident Recovery Plan Execution
- Incident Recovery Communication

Function	Category	Category Identifier
Govern (GV)	Organizational Context	GV.OC
	Risk Management Strategy	GV.RM
	Roles, Responsibilities, and Authorities	GV.RR
	Policy	GV.PO
	Oversight	GV.OV
	Cybersecurity Supply Chain Risk Management	GV.SC
Identify (ID)	Asset Management	ID.AM
	Risk Assessment	ID.RA
	Improvement	ID.IM
Protect (PR)	Identity Management, Authentication, and Access Control	PR.AA
	Awareness and Training	PR.AT
	Data Security	PR.DS
	Platform Security	PR.PS
	Technology Infrastructure Resilience	PR.IR
Detect (DE)	Continuous Monitoring	DE.CM
	Adverse Event Analysis	DE.AE
Respond (RS)	Incident Management	RS.MA
	Incident Analysis	RS.AN
	Incident Response Reporting and Communication	RS.CO
	Incident Mitigation	RS.MI
Recover (RC)	Incident Recovery Plan Execution	RC.RP
	Incident Recovery Communication	RC.CO

Fig1: NIST CSF 2.0

Core Functions

- Govern
- Identify
- Protect
- Detect
- Respond
- Recovery

2. DISCOVERY AND PASSIVE INFORMATION DISCLOSURE

Discovery and Passive Information Disclosure

This section addresses sensitive information displayed on the UI screen, such as PCI/PII data. It also reviews the configuration of security headers in the response header. Additionally, it examines whether the application interacts with external domains or URLs and checks if web services are called, ensuring that WSDLs cannot be accessed without authentication.

This section further evaluates whether the response code exposes any sensitive information, such as developer names, private email addresses, IP addresses, defect numbers, hard-coded passwords, or any other sensitive data left in comments for testing purposes. It also identifies any hidden functionality that is not actively used in the application.

The following are the best practices used for discovery and passive information disclosure.

2.1 Application should set sufficient anti-caching headers such as no-cache and no-store attributes in the response header, so that sensitive data is not cached in modern browsers.

Code Snippet in Java

```
HttpServletResponse response;
// Set the Cache-Control header
response.setHeader("Cache-Control", "no-store, no-cache, must-revalidate, max-age=0");
// Set the Pragma header to no-cache to support HTTP 1.0
response.setHeader("Pragma", "no-cache");
// Set the Expires header to 0 to prevent caching at the proxy server
response.setDateHeader("Expires", 0);
```

2.2 PCI and PII data should not be displayed in a clear text format in the UI, HTTP response, exported files, reports or email receipts. It can be shown in the masked data format. (PCI DSS Standard)

- Aadhar number
- SSN Number
- TAX ID
- Driving License
- Card Number
- DDA
- PAN Card Number
- CVV
- DOB
- Chip Information
- Passport Number

2.3 **To Fetch the complete information of PCI/PII data, application should provide an option.** When the user clicks the option, application securely fetches the data and displays it to the user. Make sure that this activity should be logged into the log file/ audit trail.

2.4 Sensitive information, which is mentioned below should not be disclosed in the UI response (PCI DSS Standard)

- Username and password
- Security question answers.
- Email or Phone number shown in full on public pages e.g. during password reset pages
- Encryption keys / algorithms.
- Client-side code data comments
- Developer names,
- Defect numbers.
- Absolute Paths to server-side resources
- Version disclosed in response body or meta tags.

2.5 Application should not reveal any sever information in the response header. (OWASP guidelines)

.

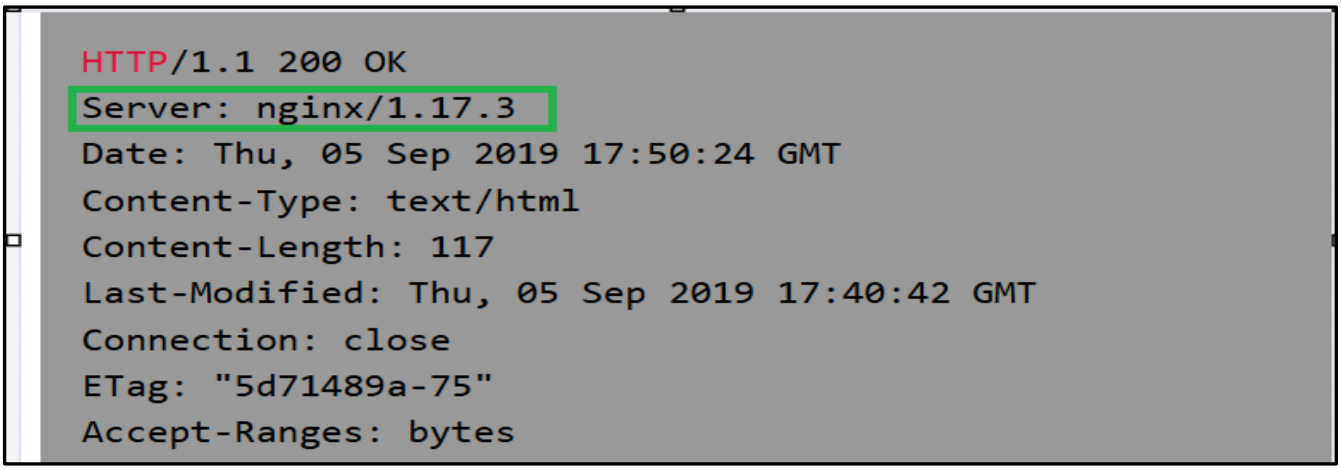

Fig 2: Server Information is disclosed in the above screen shot

2.6 Content Security Policy (CSP) Header in the response should not contain non-RBI/ReBIT domains. (OWASP guidelines).

HTTP security headers are directives sent by the server to the client (typically the web browser) through the HTTP response. They help enhance the security of web applications by providing instructions on how the browser should behave when interacting with the application. Their primary role is to mitigate security risks such as Cross-Site Scripting (XSS), clickjacking, and other vulnerabilities

Content security policy (CSP) is a feature to protect web applications from various attacks, such as XSS and data injection attacks. It allows developers to specify which resources (style sheets, scripts, images) the browser is permitted to load for a given web page.

Below examples show scripts, style sheets and images are loaded from external domain other than RBI/ReBIT.

Ex: Content-Security-Policy: default-src 'self'; script-src 'self' *https://scripts.example.com*; style-src 'self' https://styles.example.com; img-src 'self' https://images.example.com.

2.7 **Application should not pass sensitive data in the URL in clear text format (GET request or query string of POST request). Application should send the sensitive information in encrypted format or as reference number** (Ex: Reference number can be sent instead of full account number) (OWASP guidelines)

2.8 **WSDL or WADL information should not be disclosed while accessing the end point.** (OWASP guidelines)

2.9 VIEWSTATE should be signed and encrypted (VIEWSTATE is used in .NET application)

2.10 Security headers should be present in the response header. (OWASP guidelines)

- X-Content-Type-Options: *nosniff*
  Prevents browsers from interpreting files as a different MIME type than what is declared, reducing the risk of attacks.

- Content-Security Policy: *default-src 'self'*
  This specifies specific configurations based on the application's security needs.

- Strict-Transport-Security: *max-age:31536000; includeSubDomains;preload*
  Enforces HTTPS connections to the server and prevents downgrade attacks.

- Permission-Policy: *geolocation=(), microphone=(),camera=()*
  Controls which feature (e.g., geolocation, camera) can be used by your website or application.

- Referrer-Policy: *no-referrer or strict-origin-when-cross-origin*
  Limits how much referrer information is shared in the requests.

- X-Frame-Options: *DENY or SAMEORIGIN*
  Prevents the site from being embedded in an iFrame, reducing the risk of clickjacking attacks.

- X-XSS-Protection: *1; mode=block*
  Enables basic XSS protection and instructs the browser to prevent rendering the page if XSS attack is detected rather than sanitizing it.

- Content-Type: *<text/html>; charset=UTF-8* (Note: charset specifies the character encoding of HTML document which is used to Prevent XSS issue. Text/html can be any of the possible MIME type like text/css etc. It is recommended to set the content-type header correctly. MIME type specifies content type being sent to the browser)

Note: Please refer to the link below for more information.
https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html

2.11 The information below should not be disclosed in the UI as well as in the response.

- Private / Individual email Id
- IP address

2.12 Third party scripts or Non-RBI/ ReBIT domains such as Google analytics should not be loaded into ReBIT or RBI sites.

2.13 Avoid Hidden or obfuscated paths and functionalities through the analysis of metadata files like robots.txt.

Robots.txt are often used by websites to inform search engines about which parts of the site should not be crawled.

```
# Block all web crawlers from the /admin/ and /private/ directories
User-agent: *
Disallow: /admin/
Disallow: /private/

# Allow Googlebot to access everything
User-agent: Googlebot
Disallow:

# Block a specific file for all crawlers
Disallow: /secret-file.html
```

Fig 3: robots.txt file example

2.14 Web page comments, meta data and redirect bodies should not disclose any sensitive data (OWASP guidelines)

2.15 Java Script files should not disclose any sensitive information (OWASP guidelines)

3. HTTP REQUEST HEADER VALIDATION

HTTP Request Header Validation

In a cross-origin request, the header specifies the domain (including protocol and port) of the client that initiated the request. This header is included in every cross-origin HTTP request, enabling the server to identify where the request is coming from. It plays a crucial role in the Cross-Origin Resource Sharing (CORS) protocol. CORS uses the header to determine whether the server allows the requested resource to be accessed by the specified domain. The server responds with appropriate CORS headers if the request is permitted

The following are the best practices used for HTTP Request Header Validation:

3.1 Accept OPTIONS for preflight requests: Configure the server to process OPTIONS requests when they are valid pre-flight requests. Reject unnecessary methods, only allow HTTP methods that API uses (GET, POST etc.). Monitor and log invalid HTTP requests, including unusual or malicious attempts, for analysis and alerting.

3.2 The supplied Origin header is not used for authentication or access control decisions, because it can be easily spoofed or manipulated by an attacker. It is primarily used for identifying the origin of the request in the context of CORS and determining whether the server should allow the request based on its Origin policy.

3.3 The Cross-Origin Resource Sharing (CORS) Access-Control-Allow-Origin header uses a strict list of trusted domains and subdomains to match against and does not support the "null" origin.

3.4 Access-Control-Request-Method sent during a preflight request; indicates which HTTP method will be used in the actual request.

3.5 The HTTP headers added by a trusted proxies or Single Sign-On (SSO) devices, such as a bearer token for authentication, the application must validate and authenticate these headers to ensure security.

4. CRYPTOGRAPHY

Cryptography

Cryptography is a technique of securing communication by converting plain text into cipher text. It involves various algorithms and protocols to ensure data confidentiality, integrity, authentication, and non-repudiation. The purpose of cryptography is to secure and protect sensitive information by encoding it in a way that only authorized parties can understand.

Applications need to be designed with strong cryptography to protect data assets.

- All cryptographic modules fail in a secure manner. When they encounter an issue, they do not expose sensitive data or weaken security. Proper error handling ensures that failures do not lead to vulnerabilities, such as unintended information leaks or compromised encryption keys.
- Suitable random number generator is used.
- Access to keys is securely stored.

Types of Cryptography

1) **Symmetric Key Cryptography**: It is an encryption system where the sender and receiver use a single common key to encrypt and decrypt messages. Symmetric key is faster and simpler. The problem is Sender and Receiver have to exchange keys securely.

   Popular Symmetric Key Cryptography Systems are:
   - DES (Data Encryption Standard)
   - AES (Advanced Encryption Standard)

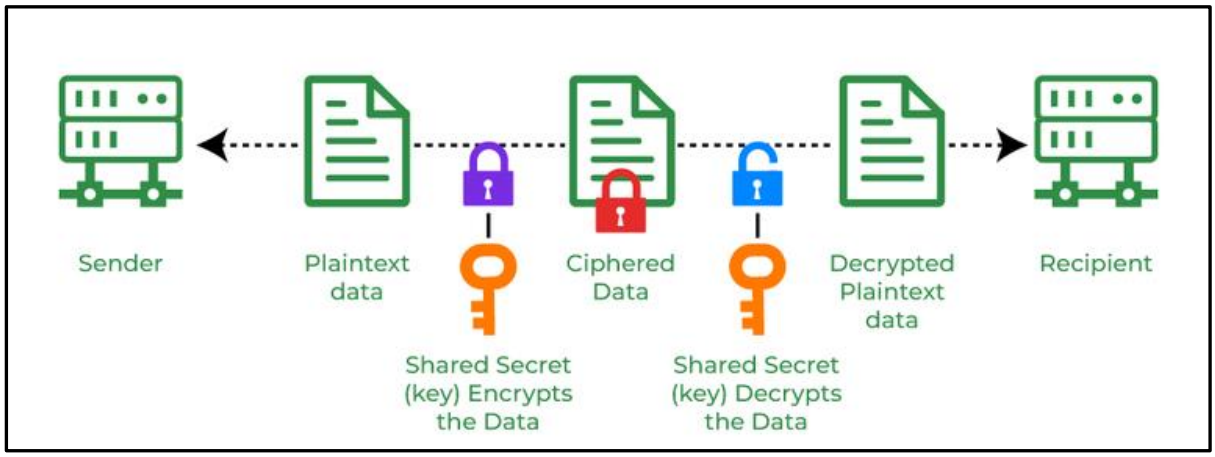

Fig 4: Symmetric Key Encryption

2) **Asymmetric Key Cryptography**: In this, a pair of keys are used to encrypt and decrypt information. A sender's public key is used for encryption and a receiver's private key is used for decryption. Public keys and Private keys are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows his private key.

Most popular asymmetric key cryptography are

- RSA(Rivest-Shamir-Adleman) algorithm
- ECC (Elliptic Curve Cryptography)
- DSA (Digital Signature Algorithm)

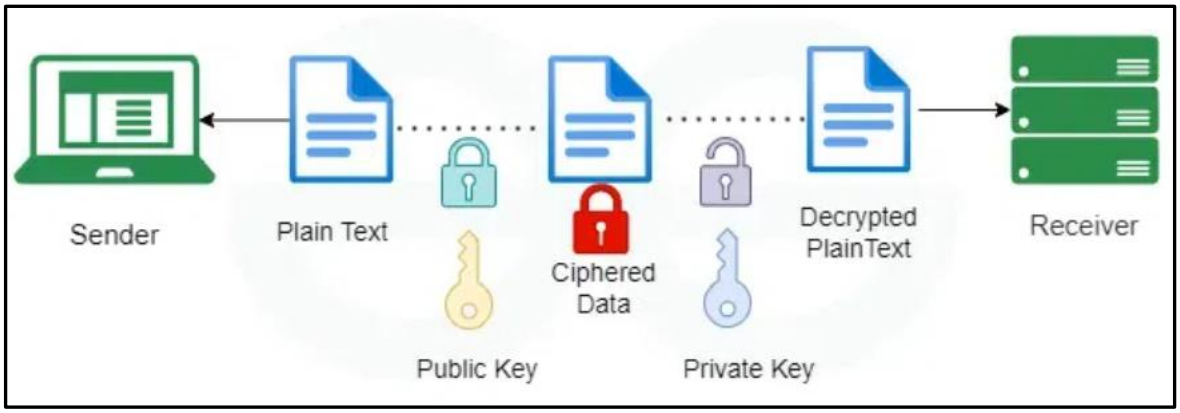

Fig 5: Asymmetric Key Encryption

3) **Hash Function**: A hash function is mathematical function that converts digital data into an output string with a fixed number of characters. Hashing is a one-way act of converting the data into the output called hash value.

Popular Hashing algorithms are:

- SHA2(Secure Hash Algorithm 2): SHA256, SHA384, SHA512 are the family of SHA2.
- SHA3(Secure Hash Algorithm 3): Latest member of SHA family.
- Argon 2: Primarily used for password hashing

The following are the best practices used for Cryptography.

4.1 Application should use the industry proven or government approved cryptographic algorithms and libraries, instead of custom coded cryptographic algorithms.

4.2 Application should use secure random number generation, encryption or hashing algorithms to ensure data integrity, confidentiality, and security. Key lengths, rounds, ciphers and modes should be configurable and upgradable at any time to protect against evolving threats and vulnerabilities.

4.3 Application should support TLS1.3 and old protocols like SSLV2, SSLV3, TLS1.0, TLS1.1, TLS1.2 etc. should be disabled. TLS1.2 can be used ONLY if TLS1.3 is not available. (NIST Standard.800-52R2).

4.4 For TLS1.3 ,0-RTT (Round Trip Time) should be disabled. If the application involves sensitive operations or non-idempotent actions, it is safer to disable 0-RTT to mitigate the risks (Replay attacks, Lack of Forward Secrecy) (NIST Standard.800-52 Revision 2).

4.5 Ciphers with RSA, SHA, CBC, RC4, CHACHA20, EXPORT should be disabled. Only ciphers suites with keys 128 bits or longer should be enabled. Disable RSA with small keys (e.g., 1024 bits). Use RSA with a minimum of 2048 bits for enhanced security. (NIST.SP 800-52R2, Section 3.3, SP 800-131A Revision 3, OWASP standard Transport Layer Security Cheat Sheet).

4.6 ECDHE less than 256 bit and DHE less than 2048 bit are disabled. Both ECDHE and DHE are used for secure key exchange during TLS handshake, ensuring forward secrecy. Disabling weaker variants protects against attacks like brute-force, key compromise, and cryptanalysis (NIST.SP 800-56A R3).

4.7 Disabling SSL/TLS renegotiation and secure client renegotiation is an important measure to mitigate certain vulnerabilities, such as renegotiation attacks or denial-of-service (DoS) attacks.

4.8 Server should not be vulnerable to OpenSSL Heartbleed attack. The OpenSSL Heartbleed vulnerability (CVE-2014-0160) is a serious flaw that could allow attackers to access sensitive data from the memory of affected servers. Ensure that the OpenSSL version used on server is at least 1.0.1g or later, as this version contains the patch for the Heartbleed vulnerability

4.9 SSL Certificate for the below points (NIST.SP.1800-16 section 2.3.1.2.7)

4.9.1 SSL certificate should be issued to correct domain name.
4.9.2 Application should not accept the wild card certificate.
4.9.3 SSL certificate should not exceed 395 days maximum validity lifetime.
4.9.4 Application should not use the self-signed certificate.

4.10 SSL(HTTPS) connections are used for all the pages including the login page. (NIST SP 800-52 R2)

4.11 Application should not accept HTTP connection by changing HTTPS to HTTP. (OWASP guidelines).

4.12 Weak ciphers and hashing algorithms (MD5, SHA1 etc.) should not be used by the application. (NIST SP 800-131A Revision 2).

4.13 Application should not use weak encoding methods used for sending/storing sensitive data such as

- Base 64
- HEX
- ASCII

- URL Encoding

4.14 Application should not use weak encryption methods such as

- Caesar Cipher/ Caesar Shift.
- Simple Substitution Cipher

4.15 Nonces (numbers used once), initialization vectors and other single use of numbers must not be used more than once with a given encryption key. The method of generation must be appropriate for the algorithm being used. (OWASP guidelines).

4.16 Encrypted data is authenticated via signatures, authenticated cipher modes, or HMAC to ensure that ciphertext is not altered by an unauthorized party. (OWASP guidelines).

4.17 All cryptographic modules fail securely, and errors should not disclose information regarding the encryption algorithms etc.

4.18 Application should not use same key for data in rest and data in transit. (OWASP guidelines).

4.19 Effective management of cryptographic keys is essential to maintaining security in any cryptographic system. A well-defined policy and adherence to a key management standard ensure that keys are securely generated, stored, distributed, used, and retired. (NIST SP 800-57 Part1 Revision 5).

4.20 All keys and passwords are replaceable and are part of a well-defined process to re-encrypt sensitive data. When keys are replaced, sensitive data encrypted with old keys must be re-encrypted using the new keys. This ensures that the data remains protected under the latest security measures. A clear and standardized procedure for key and password replacement helps minimize disruptions and ensures the transition is secure.

5. AUTHENTICATION

Authentication

Authentication is the act of establishing someone as authentic and that claims made by a person or about a device are correct, resistant to impersonation, and prevent recovery or interception of passwords. The use of Username and password were the most common forms of authentication outside of high security systems. While Multi-factor Authentication (MFA) is widely accepted in security circles, it is rarely required elsewhere. As the number of password breaches increased, the idea that usernames are somehow confidential and passwords unknown, has rendered many security controls untenable. As per NIST 800-63, passwords are called Memorized secrets, which includes PINs, passwords, unlock patterns, pick the correct kitten or another image element, and passphrases.

The following are the best practices used for Authentication.

5.1 User set passwords are at least 12 characters in length. The maximum password length should be at least 64 characters to allow passphrase. (NIST SP 800-63B, Section 10.2.1)

5.2 Application should allow the user to change their password. (NIST Standard 800-63B)

5.3 Password change functionality requires that user's current and new user password. (OWASP guidelines)

5.4 Password strength meter is provided to help users set a strong password (NIST Standard 800-63B Section:5.1.1.2).

5.5 Use of weak authenticators should be limited to secondary verification and not as a replacement for more secure authentication methods.

5.6 Application should implement extra authentication while performing at least below scenarios (NIST SP800-63B Section 4)

- Personal Information (add/ update/ delete)
- PCI information (add/updates/delete)
- Any monetary transactions (add/ update/ delete)
- Credentials reset
- Adding users (add/update/delete)

5.7 Application should send notification to the user while performing at least below scenarios such as

- Credentials reset.
- PCI information addition/updates/deletion
- Personal Information (add/modification/deletion).
- Critical Transactions like payment activity.
- Concurrent login activity.
- Email notification to both old and new email address when email address is modified.

5.8 Credential Service provider (CSP) and application verifying authentication are separated, mutual authenticated TLS is in place between the end points.

5.9 Login form and other screens containing sensitive fields like username, SSN, Credit/ Debit Cards, CVV, Account numbers, PAN, Email, Phone, Tax ID or any other sensitive information field should have auto complete=off at field or at form level. (OWASP guidelines)

5.10 The Application should update the password only after the server validates the current password.

5.11 The password change screen should never allow one user's password to be replaced with another user's credentials. Ensure each user's credentials remain isolated and cannot be modified by another account.

5.12 Username length should not be less than 10 characters. (NIST standard)

5.13 Application should not have "remember me" functionality that pre-fills the username on login screen.

5.14 Application/ web service should not support following authentication methods:

- Basic authentication
- NTLM V1 authentication

5.15 Application should lock the user account after 3 invalid attempts for at least 30 minutes or adjusting the duration based on business requirements on the following pages.

- Login page
- MFA verification page

5.16 Check for missing or weak Anti Automation or CAPTCHA or rate limit wherever features may be abused for. It ensures strong anti-automation, CAPTCHA, rate limiting to prevent abuse in applications.

Example:

- Publicly availability pages like contact us, Feedback, Signup/Registration or any other public pages. (seems incomplete sentence)
- Internal features like account lockout/CAPTCHA cannot be implemented rate limiting is an option.

5.17 The user should not bypass CAPTCHA using the following steps

- The CAPTCHA value cannot be reused and must be random
- The CAPTCHA step cannot be bypassed (seems repeat sentence of headline)
- Submit the form without CAPTCHA parameter/ value
- CAPTCHA value should be generated and verified at server side

5.18 Application (username enumeration) should display generic error message on the login page for the following scenarios: (OWASP guidelines)

- Valid user ID and invalid password
- Invalid user ID and valid password
- Invalid user ID and password

5.19 Application should display generic error messages for the following pages where it accepts username as input. (OWASP guidelines)

- Forgot my password
- Signup/ Registration page
- Supervisor / override password

5.20 User cannot create an account with duplicate username on Self-Signup / Registration Page or by admin user. (OWASP guidelines)

5.21 Following scenarios should be implemented, if application implements SSO (Single Sign On):

- SSO token should be unique, and it should not be replayable
- Check for user identifiers in the requests and it should not have any impersonations by changing the values
- Verify the message level confidentiality and integrity should be in place
- SSO token expiration time should be 5 minutes. Invalidating the SSO token after usage adds an additional security layer by preventing replay attacks.

5.22 Multi Factor Authentication (MFA) should be implemented in the application after the user login into the application.

5.23 The following conditions should be checked for MFA implementation:

5.23.1 Same Factor MFA

5.23.1.1 Application presents OTP page and shows error only on that page

5.23.1.2 Application presents OTP page and shows error only on that page regardless of correct/incorrect password on login

5.23.2 Bypassing MFA

5.23.2.1 Application should not access internal URLs of the application without entering MFA details

5.23.2.2 Application should not accept blank or invalid MFA value

5.23.2.3 Application should not accept the old OTPs

5.23.2.4 Application should not respond when the user login and stay at MFA page in one tab and access internal URLs in another tab

5.23.2.5 Analyse and tamper the response parameter to bypass the MFA wherever it is implemented. (E.g., if the application simply checks for a success: true in the response body to proceed, an attacker could change “success”: false to “success“: true in the intercepted response, to bypass the MFA.

5.24 OTP-based MFA should have the following controls:

5.24.1 OTP is characters in length and may be entirely numeric (NIST SP600-63B Section 5.1.5.1)

5.24.2 OTP should be random and non-predictable

5.24.3 The expiry of OTPs should indeed be within a short time span to minimize security risks. Expiry time can depend on the business requirements.

5.24.4 OTPs should not be reusable

5.24.5 Accounts should be locked for 30 minutes on entering invalid OTPs 3 times

5.24.6 Rate limiting on generating OTPs (Maximum 5 OTPs can be generated)

5.24.7 OTP should be unique to a specific user and their authentication session

5.25 Application can use a security question and answer to serve as the third or fourth layer in MFA, enhancing the authentication process. However, it is crucial to select questions with answers that are difficult to guess and not easily accessible through public information.

5.26 Time-based One-Time Passwords (TOTPs) have a defined lifetime (60 seconds) before expiring. TOTP is a type of dynamic password that changes after a set period of time. It is generated using a hashing algorithm that combines a secret key (shared between the user and the service) with the current time to produce a unique code. It is widely used in two-factor authentication(2FA). Applications like Google Authenticator or Microsoft Authenticator can generate these TOTPs.

5.27 Time-based OTP (TOTP) can be used only once within the validity period.

5.28 Following best practices should be implemented by the application during Forgot password.

5.24.8 Forgot password steps should not be bypassed.

5.24.9 Application should implement MFA before sending a password reset link, which enhances security.

5.24.10 Application should send password reset link to the registered email.

5.24.11 The application should not allow password setup on the screen after clicking on password reset link but should use different verification method such as OTP via SMS/email, answers to security questions, or biometric authentication (e.g., fingerprint or facial recognition), creates multiple barriers for attackers.

5.24.12 Temporary password reset link should not be reused. Once the reset link is used, invalidate it immediately.

5.24.13 Application should not display mobile number or email ID in clear text format in forgot password scenario.

5.24.14 Temporary password reset link should expire within 15 minutes time.

5.24.15 Temporary password reset link must be unpredictable to ensure security.

5.24.16 Restrict the number of passwords reset attempts to prevent brute force attacks.

5.24.17 Implement captcha or similar mechanism to differentiate real users from the bots.

5.24.18 5.28.11 Log the user activity in the log file for forgot scenario. Application should notify (Email/SMS) the user for forgot password activity as well as any suspicious activity in forgot password scenario.

5.29 Applications should send notifications to both the old and new email/mobile number, when the user changes their email ID/ mobile number. This approach minimizes the risks associated with it. The user can confirm the update was intentional on both ends.

5.30 OAuth2.0(Open Authorization 2.0) is an industry standard framework that allows third-party applications to access user data on behalf of a user without requiring the user to share their credentials. OAuth uses tokens to represent user's access/permission.

Below are the best practices for OAuth2.0

5.30.1 Validate Redirect URIs: Allow redirect URIs that are pre-registered and explicitly whitelisted by the authorization server, which ensures that only trusted URIs are used in the OAuth process.

5.30.2 Secure communication: Always use HTTPS for all requests to protect the data in transit.

5.30.3 State and Nonce Parameters: Use state parameters to prevent CSRF attack. Implement the nonce parameter to prevent replay attack.

5.30.4 Secure Token Storage: Avoid storing tokens insecure location such as browser's local storage or session storage. Use secure cookies with HTTP Only and Secure flags to store tokens if required.

5.30.5 Issue short lived access tokens (15 minutes or as per the business requirement) with a limited lifespan. Implement refresh token rotation to mitigate risks if a token is compromised.

5.30.6 Avoid token exposure in URLs: Never include access tokens in query parameters. Send the tokens in the Authorization Header.

5.30.7 Use PKCE (Proof Key for Code Exchange) to secure the authorization code flow, particularly for public clients like mobile applications or single page applications, where client secrets cannot be securely stored.

5.30.8 The implicit grant flow is considered insecure because it exposes access tokens directly in the browser's URL fragment, making it vulnerable to interception.

5.30.9 Monitor OAuth flows and maintains logs to detect and respond to suspicious activities.

5.30.10 Limit token scope and permissions and follow the principle of least privilege by requesting only the permissions necessary for the application.

5.31 JSON Web Tokens (JWT) are widely used for secure authentication and data exchange. The following are the security standards for JWTs:

Most programming languages have libraries to generate JWT tokens such as jsonwebtoken in Node.js, java-jwt in java, System.dentityModel.Tokens.jwt in C#. Also jwt.io tool can be used for generating and debugging JWT.

There are 3 parts in JWT separated by a period.
The Header, the payload and signature. Each section is base 64 encoded.

- Header: The header typically contains information like the algorithm used for signing (e.g., HS256, RS256) and the type (JWT).

```
{
    “typ”: “JWT”,
    “alg”:  “HS256”
}
```

- Payload: The payload consists of claims. Claims are statements about user and additional information.

```
{
   "iss": "example_issuer",
   "sub": "user_1000",
   "exp": 1644768000,
   "iat":    1644944000
}
```

- Signature: Signature ensures the integrity of the token and verifies that the token has not been tampered.

  Signature = HMAC-SHA256(base64urlEncode(header) + "." + base64urlEncode(payload), secret_salt)

5.31.1 Avoid storing JWTs in local or session storage, as they are vulnerable to XSS attacks, instead use secure cookies.

5.31.2 The JWT token should not accept NONE Hash algorithm.

5.31.3 JWT Token should not contain any sensitive information like PCI/PII information.

5.31.4 The JWT token should be expired after 15 minutes of idle time or depending upon business requirement.

5.31.5 Always verify the JWTs signature to ensure it is not tampered with. Check critical claims like issuer (to confirm it was issued by trusted entity) and expiration (to ensure token is not expired). Also validate the additional claims like audience and subject depending on

the application requirement. Server should properly validate the algorithm specified in the JWT header.

5.31.6 Application should invalidate the JWT token after logout of the application.

5.31.7 Applications should use cryptographic algorithms like RS256 or HS256.

5.31.8 Keep track of authentication events to detect and respond to suspicious activities.

5.31.9 Applications should not use the old or previously used JWT tokens to prevent the replay attacks.

5.31.10 If application utilize the access token (JWT token) for a long time, a hacker may steal it and misuse it. As a result, using the access token for an extended period is not recommended. Typically, refresh tokens are used to obtain new access tokens (JWT token). When a client uses a refresh token to get a new access token, the server issues a new refresh token along with the access token. The old refresh token is immediately invalidated.

6. PASSWORD SECURITY

Password Security

Passwords are referred to as 'Memorized Secrets' and include passwords, PINs, unlock patterns, and passphrases. They are classified as something you know and are often used as single-factor authenticators. However, there are significant challenges associated with the continued use of single-factor authentication. These include the exposure of billions of valid usernames and passwords on the Internet, the prevalence of default or weak passwords, and the risks posed by techniques like rainbow tables and ordered dictionaries of commonly used passwords.

The following are the best practices used for Password policies.

6.1 Application should accept the password whose length of 12 characters. Password should contain at least one of the characters below:

- Upper Case
- Lower Case
- Numbers
- Special characters Allowed are @, $, #, %

6.2 The password change functionality requires user's current and new password. Application should validate the current password. (OWASP guidelines)

6.3 Application should not contain username in the password. (NIST SP 800-63B, Section 5.1.1.2)

6.4 Application should not accept 'password' in the password field.

6.5 Application should force the user to change the temporary password during initial login. (NIST SP 800-63B, Section 5.1.1.2)

6.6 The application should not accept previous passwords.

6.7 Application should force the user to change the password after 90 days. Again, this should depend on the business requirement.

7. CREDENTIALS STORAGE

Credentials Storage

Credential storage refers to the methods used to securely store authentication information, such as usernames, passwords, and other sensitive data. Proper credential storage is crucial for preventing unauthorized access and protecting sensitive information.

The following are the best practices used for Credentials Storage.

7.1 Application should not use following:

a) Insecure Cryptographic Storage
b) Weak Encryption Algorithm
c) Weak Cipher Functions
d) Custom Encryption Methods

7.2 Passwords are not stored in clear text format. Instead, passwords should be salted with hash values while storing them in the database.

7.3 Salt value used for password hashing should be a minimum of 128 bits. Salts help protect against attacks like rainbow table lookups by introducing randomness to the hashing process. A 128-bit salt ensures a high degree of uniqueness, even for a large number of hashed values, making it computationally expensive for attackers to crack.

7.4 Application password should be hashed and not encrypted. Recommended algorithms like BCRYPT, Argon2, and PBKDF2 support proper use of salts with sufficient length and randomness.

7.5 If the application uses password based key derivative function (PBKDF2) for hashing of the password, the iteration count should be large as verification server performance will allow, typically at least 100,000 iterations.

7.6 If BCRYPT is used for hashing of the password, the work factor should be as large as verification server performance will allow, with a minimum of 10.

7.7 The Card Holder Data should be encrypted with a strong encryption algorithm in the database. Hashing of this information is preferred over encryption.

- Primary Account number,
- Credit Card number,
- Debit Card Number,
- Card Holders Name,
- Expiration Date

7.8 Sensitive and Personally Identifiable Information (PII) is in encrypted form in the database. Hashing of sensitive information is preferred over encryption.

7.9 Salt and Hash value should be stored in separate tables

8. AUTHORIZATION AND ACCESS CONTROL

Authorization and Access Control

Authorization is the process of determining an entity's access rights and permissions. A user who has been authenticated is not necessarily authorized to access all resources or perform all actions within a system. Ensure that analysts understand the various roles and permissions that the application supports.

Following matrix gives clarity regarding user roles and functions, mdules that user can access.

S no.	Tenant/ Bank/ FI Name or Code	User Role	Modules/ Feature Names accessible to user role	Action Allowed(Create/ Read/ Update/ Delete)

The following are the best practices used for Authorization and Access Control:

8.1 Application should evaluate the user's permissions before granting accessing to privileged data or fucntions and only authorized activities are permitted in the application.

8.2 Implement the principle of least privilege - Users should only have access to functions, features, data files, URLs , services and other resources for which they have specific authorization.

8.3 A user should not be able to access another user's information within the same role (preventing horizontal privilege escalation) by manipulating any parameters (Below are some of the examples):

- Cookies(Exploiting session cookies to impersonate users or gain elevated access)
- Request headers(Modifying HTTP headers to bypass security checks)
- Request parameters (Altering parameters in API calls or web requests)
- URL manipulation(Changing URL paths to access unauthorized areas of an application)
- Session ID (Manipulating session identifiers to gain access to privileged accounts)
- Token value (Exploiting authentication or authorization tokens to escalate privileges)
- Response manipulation
- Sequential value generared for the user(Manipulation of sequential value of one user with other user)
- Unique value generated for the users (Manipulation of unique value of one user with other user).

8.4 Low privileged user can't access high privileged user's information (Vertical privilege escalation should not exist) by manipulating any parameter (Below are some of the examples):

- Cookie values
- Request headers
- Request parameters
- URL manipulation

- Session ID
- Response manipulation
- Token value
- Sequential value generared for the user
- Unique value generated for the users

8.5 **One Tenant/FI/Bank/Merchant user can not access other user of Tenant/FI/Bank/Merchant** by manipulating following parameters:

- Cookie values
- Request headers
- Request parameters
- URL manipulation
- Session IDs
- Response manipulation
- Sequential value generared for the user
- Unique value generated for the users
- Token values

8.6 The application should prevent forceful browsing of files by restricting direct access to files for which the user does not have authorization.

8.7 User should not get excessive data in the response for which user is not granted by manipulating any parameter value.

8.8 The application should prevent modification of parameter values, especially when user-controllable parameters are used for logging user activities, as this can lead to log injection attacks.

8.9 The application should not disclose a user's access controls in the event of an exception.

8.10 Elevation of user privileages should be notified to the admin / application owner.

8.11 Application should have facility (report/ read only access) to review and the roles and privileges of the users in the application.

9. SESSION MANAGEMENT

Session Management

A Session is a sequence of HTTP requests and responses that share a common context. This context is often established through a unique session ID, usually stored in cookies. A session allows the server to remember stateful information about a client across multiple requests.

Session Management changes a stateless protocol to stateful protocol, which is critical for differentiating different users. The core component of any web-based application or stateful API is the mechanism by which it controls and maintains the state for a user interacting with it.

The following best practices are used for session management.

9.1 The application should never reveal session tokens in URLs.

9.2 The application should generate a new session value/token upon user authentication. If a session value/token is generated prior to authentication, the application must ensure that a new session value/token is generated after authentication.

9.3 Session values or tokens should possess at least 128 bits of entropy to ensure security. This can be verified using VAPT tools like Burp Suite. Application should store the session IDs/ tokens on server side instead of client side.

9.4 Session values or tokens should be generated using approved cryptographic algorithms to ensure security and compliance.

9.5 Application should have the logout functionality. Verify that session is destroyed on the server when the user logs out from the application.

9.6 Ensure that the application provides an option to terminate all other active sessions following a successful password change.

9.7 If authenticators permit users to remain logged in, verify that re-authentication occurs periodically both when actively used or after idle period.

9.8 Cookie-based sessions should have the following
Attributes: (OWASP guidelines)

- Secure (Secure flag ensures cookies are transmitted over HTTPS)
- HTTP only (Prevents client-side scripts from accessing cookies)
- Path (The cookie path should be set to the application root, rather than the server root, to ensure proper scope and security)
- Same site (Setting the cookie attribute Strict by default is a strong security measure)

9.9 Application should not permit concurrent logins. (OWASP guidelines).

9.10 Application should log the concurrent login user activities in the log file.

9.11 Application should display the last login time stamp of the user on top of the application. Users can quickly identify if there has been unauthorized access to their account by checking the last login time.

9.12 Applications are protected from cross site request forgery via the use of at least one or more of the following: (OWASP guidelines)

- Double submit cookie pattern
- Anti CSRF token

9.13 Application should assign new value to session cookie after authentication to avoid session fixation issue.

9.14 Application should not maintain the persistence cookies, which contain any sensitive information.

9.15 Application should not be vulnerable to session Hijacking. Session hijacking occurs when an attacker gains unauthorized access to a user's session, often by stealing session tokens or cookies. This can lead to severe consequences like data breaches or identity theft.

9.16 Session ID should be bind with the User ID, Client IP address, User Agent (OWASP guidelines).

9.17 Application should be redirected to login page after 15 minutes (or as per business requirement) idle session time. Redirecting users to the login page after idle time is a security measure to protect against unauthorized access due to unattended sessions.

9.18 User session should be disabled automatically when admin disables the user account. It ensures that the user cannot continue to access the application after their account has been disabled.

10. INPUT VALIDATION

Input Validation

The most common web application security weakness is where input validation is not properly implemented or using it without any output encoding, this weakness can lead to many injection issues such as XSS, SQL injection, file system attacks, buffer overflows etc. Output encoding is critical to the security of any application. Typically, output encoding is used to render the output safe in the appropriate output context for immediate use.

Input validation is checking the user supplied data against predefined rules. The validation can include:

- **Data Type Validation**: Ensure input matches the expected data type (integer, string, date etc.).
- **Length Validation**: Limiting the input to a specific

The following are the best practices used for Input Validation:

10.1 All user inputs on the client-side and server-side using a Whitelist approach, ensuring only expected data types, length, range and formats are accepted. (OWASP guidelines)

10.2 Application should use prepared statements or parameterized queries when interacting with database to separate SQL code from user input.

```
public class PreparedStatementDemo {
    public static void main(String[] args) throws Exception
    {
        Connection con = DriverManager.getConnection();
         String query
            = "Select Student_id, Student_name from students where age> ? and name = ?";

        PreparedStatement myStmt
            = con.prepareStatement(query);
         myStmt.setInt(1, 20);
        myStmt.setString(2, "Student_1");

        // Close the connection
        con.close();
    }
}
```

Fig 6: Example Prepared statement sample code

10.3 Validate when a web application receives untrusted input data that could lead to redirecting or forwarding the request to a URL specified in the untrusted input, potentially pointing to unknown websites.

10.4 All validation failures should result in input rejection. Do not store invalid data or display it in error message or logs. (OWASP guidelines)

10.5 Use a centralized input validation library or framework for the whole application.

10.6 Allow URL redirects and forwards only to destinations that are explicitly listed in an allow list (or whitelist).

10.7 All untrusted HTML input is properly sanitized with an HTML sanitizer library or framework feature. (OWASP guidelines)

10.8 Application sanitizes the user input before passing to mail systems to protect against SMTP or IMAP injection attacks. (OWASP guidelines)

10.9 Avoid the use of eval () and other dynamic code execution features. These features can be highly risky, especially when user input is involved, as they can open the door to code injection attacks.

10.10 The application should protect against template injection attack by ensuring any user input being included is sanitized.

10.11 Applications should implement output encoding where any untrusted data is displayed on the web page. Output encoding is the process of transforming data into a safe format that does not interfere with the intended functionality or appearance of the web page. (OWASP guidelines)

ESAPI (The OWASP Enterprise Security API) is an open-source library that provides functions for encoding the special characters of the input strings, which helps to prevent the various injection issues.

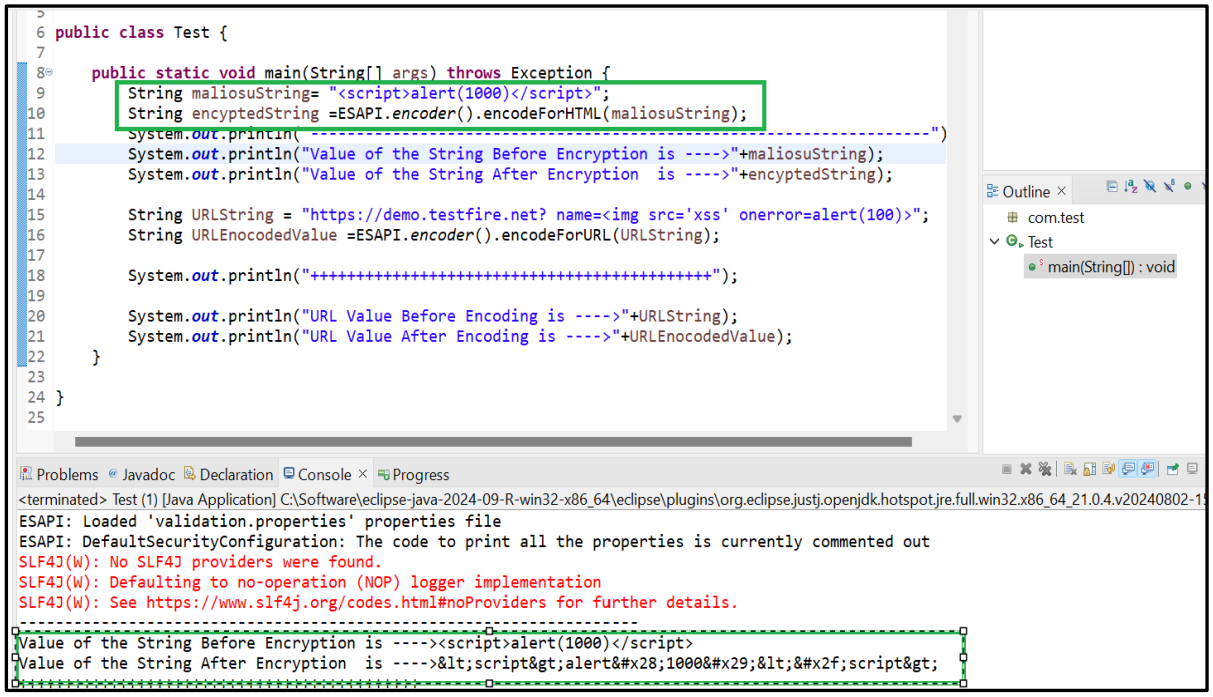

Fig 7: Sample code for ESAPI API for encoding the HTML special characters.

The encodeForHTMLAttribute() method encodes it to make it safe to use with HTML attributes.

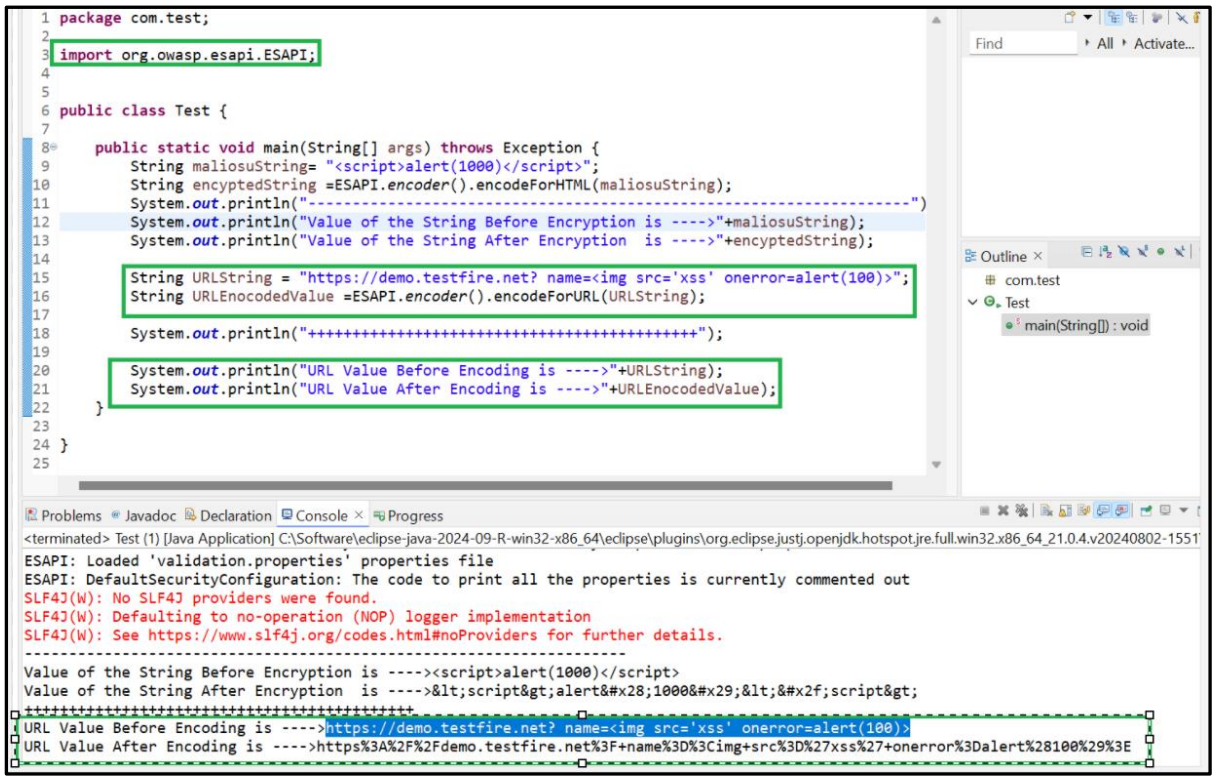

Fig 8: Sample code for ESAPI API for encoding the special characters in URL.

The unsafeURL contains potentially dangerous characters. The encodeForURL method encodes it to make it safe for use as a URL parameter.

2nd Method:

For encoding the Development team can use the following option org.apache. commons. text. StringEscapeUtils

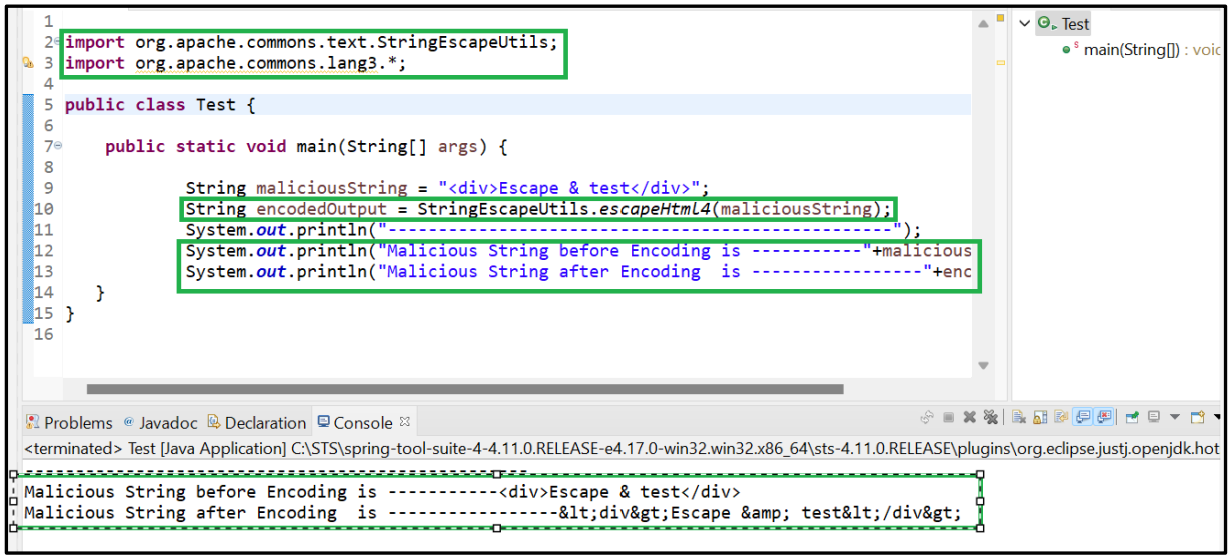

Fig 9: Sample code for encoding the HTML special characters using Apache Commons Lang's StringEscapeUtils.

10.12 Application should protect against OS command injection, where it executes arbitrary commands on the host operating system via vulnerable application (OWASP guidelines).

10.13 Application should protect against Local File Inclusion (LFI) or Remote File Inclusion attacks. (OWASP guidelines)

10.14 The application should protect against XPath injection or XML injection attacks (OWASP guidelines).

10.15 The application should validate the Host header against an allow list of trusted domains or expected hosts to avoid redirection to unauthorized or malicious URLs.

10.16 Check for Path Traversal/ Directory Traversal attack allows to access the files and directories that are stored outside of web root folder by manipulating variables that reference files with. For example: ../../../../../../etc/passwd or ../../../../../../boot.ini based on OS.

10.17 Check for CSV injection (Formula Injection) by exporting untrusted user input to CSV files and later open in excel. If the input starts with =, +, -, @, it may be interpreted as formula, potentially leading to malicious actions.

10.18 10.18 The application should accept allowable characters (Whitelisting) for each input field as per the business requirement, it should block the remaining characters to prevent different types of XSS (Stored XSS, Reflected XSS, DOM XSS, HTML Injection, Link Injection, Frame Injection) issues. (OWASP guidelines)

10.19 The application should respond with valid error message/ generic error message on entering long values for request parameters. Application should not display detailed errors. Application should not be vulnerable to DoS attack (OWASP guidelines)

10.20 Application should respond with valid error message/ generic error message on entering null value for request parameters. Application should not display the detailed errors on entering null values. (OWASP guidelines)

10.21 Application should not accept and respond with generic error message in the following scenarios (OWASP guidelines).

10.21.1 Enter duplicate parameters in a request. (HTTP Parameter Pollution)
10.21.2 Enter multiple parameters in a request
10.21.3 Enter custom parameters in a request.

10.22 Application should not process the request on sending POST request as GET request.

10.23 The application should not treat HEAD requests for internal pages differently from GET requests in terms of response handling, as this could expose inconsistencies or vulnerabilities. A HEAD request is meant to return the same headers as a GET request without the response body.

11. SERVER SIDE REQUEST FORGERY(SSRF) PROTECTION

Server Side Request Forgery(SSRF) Protection

Server-side request Forgery (SSRF) is web security vulnerability that allows an attacker to cause server-side application to make requests to an unintended location.

The attacker tricks the server into sending requests to internal or external resources by supplying a malicious URL. These requests can target internal systems (e.g., databases, cloud metadata services) or external systems, potentially exposing sensitive data.

The following are the best practices used for Server-Side Request Forgery:

11.1 Web or application server is configured with an allow list of resources or systems to which the server can send requests or load data/files from. It strictly limits the destinations the server can access, reducing the risk of unauthorized or malicious requests. Specify domains, IP addresses, or subnets that the server is permitted to access.

11.2 Ensure that server-side requests cannot be controlled by the end user. Allow the server to interact with pre-approved, trusted endpoints or systems. Reject requests for destinations that are not explicitly included in the allowlist.

11.3 Make sure that all user supplied data including URLs are validated and sanitized to prevent malicious input from being processed. Below the best practices can be used.

- Ensure all user supplied URLs are checked against a strict validation process. This includes verifying the format, protocols and domain.
- Reject URLs with suspicious patterns such as localhost, 127.0.0.1 or private IP ranges.
- Use URL parsing libraries to validate components like hostname, port and path.
- Disable automatic redirects to prevent attackers from chaining requests.
- Secure Libraries.

11.4 Limit the application access to range of allowed IP addresses to minimize the attack surface and prevent unauthorized access to internal resources.

11.5 Enforce a firewall policy that specifies what hosts and applications are allowed to connect to is an effective strategy for mitigating Server-Side Request Forgery (SSRF) attacks.

11.6 Disallow protocols like file://, gopher://, or ftp:// that could be exploited. These protocols can be exploited by attackers to gain unauthorized access or extract sensitive information from servers.

12. FILE UPLOAD SECURITY

File Upload Security

File uploads are an integral feature for modern applications, but they do come with inherent risks, especially from malicious or corrupt files

The following are the best practices used for File Upload features.

12.1 The application should not accept large files that can cause the denial of storage. Applications should accept the specific file size that they can accept.

12.2 Applications should restrict the types of files they allow for upload, blocking potentially dangerous executables such as .exe and .bat and any other executable files.

12.3 The application should check the compressed files (ex: zip, gz, docx, odt) against their maximum allowed uncompressed size and maximum number of files before extraction. Some compressed files, like "zip bombs" are designed to expand into enormous sizes, potentially exhausting system resources and causing a denial of service. Verifying the uncompressed size helps to mitigate risk.

12.4 The file size quota and maximum number of files per user is enforced to ensure that a single user cannot fill up the storage with too many files or excessively large files.

12.5 Applications should reject files with double extensions (Ex: TestFile.jpg.exe) and null byte extensions (Ex: TestFile.php%00.jpg), as they can be used to bypass validation and execute malicious code.

12.6 Application should implement antivirus/CDR scanning to detect malicious content before processing the files.

12.7 Application should prevent Cross Site Request Forgery during the file upload scenario.

12.8 Application should validate the Content Type/ MIME header of the file.

12.9 Ensure protection against XSS, it is vital to safeguarding users and the application from malicious scripts. Prevent uploaded files from being executed or accessed.

12.10 Verify the content of uploaded files for malicious characters or code, this is a vital security measure. File should not contain any malicious characters or patterns.

12.11 Sanitize the filenames to prevent injection attacks or directory traversal.

12.12 Implement rate limiting to restrict the number of uploads a user or IP address can make within a specific time frame.

12.13 Implement logging mechanisms to track file upload activities and monitor for suspicious patterns.

13. DESERIALIZATION PREVENTION

Deserialization Prevention

Serialization is the process of converting an object into a format that can be easily stored, transmitted or reconstructed later. This is used when user need to save the state of an object to a file, send data over a network or share information between systems. JSON (JavaScript Object Notation) and XML are widely used formats for serialization in many programming languages.

In Java, Serializable interface helps to achieve this. In Python, pickle module is used for serialization. The JsonSerializer.Serialize method in .NET is used to convert .NET objects into JSON format. Deserialization is the reverse of this process; it takes the serialized data and rebuilds it. Attacks against deserialization are referred to as Insecure Deserialization, they occur when untrusted or manipulated data is deserialized without proper validation, which can lead to Remote Code Execution (RCE), Denial of Service (DoS), Privilege Escalation and Data Tampering.

The following are the best practices used for Deserialization Prevention:

13.1 Never trust user supplied data for deserialization unless it has been strictly validated or sanitized.

13.2 Ensure the serialized data comes from a trusted and verified source.

13.3 Prefer secure data formats like JSON or XML for communication instead of native serialization mechanisms.

13.4 Validate all input data before processing it. Reject unexpected or malformed data during deserialization. Apply strict validation rules, such as whitelisting acceptable data types or structures.

13.5 Utilize serialization libraries or frameworks that incorporate built-in security mechanisms. Avoid libraries with known vulnerabilities and always keep application dependencies up to date.

13.6 Avoid using features like reflection or dynamic code execution during deserialization. Ensure that the deserialization code does not allow execution of arbitrary code.

13.7 Implement Integrity checks such as hashing or digital signature to verify serialized data has not been tampered with. Reject data that fails these checks.

13.8 Use strict access controls to limit the types of objects that can be deserialized. Explicitly restrict deserialization to expected and safe object types only.

13.9 Monitor deserialization related operations and log the suspicious activity, which helps in detecting and mitigating attacks.

13.10 Regularly update the software, libraries, and frameworks to ensure known vulnerabilities are patched.

13.11 Web application Firewalls (WAF) can be used in mitigating the insecure deserialization attacks.

14. DATA PROTECTION AND PRIVACY

Data Protection and Privacy

If the application is storing and/or processing personal data, then compliance with the Digital Personal Data Protection (DPDP) Act must also be ensured. Personal data or personally identifiable information (PII) means any information connected to a specific individual that can be used to uncover that individual's identity. The safeguarding of PII is of the utmost importance to ensure compliance with the DPDP Act. Examples of PII are:

- Name
- Date of birth
- Phone number
- Aadhar Number
- PAN
- Fingerprints
- Retina scan
- Residential address
- Credit card number
- Bank account number
- Medical records
- Passport details

Please note that the above list is not exhaustive.

Some definitions in the DPDP Act - “Data Fiduciary” refers any person, company, organization responsible for deciding the purpose and methods of processing personal data, either individually or in collaboration with others. “Data Principal” means the individual to whom the personal data relates. “Data Processor” means any person who processes personal data on behalf of a Data Fiduciary.

The following are the best practices used for Data Protection and Privacy.

14.1 Ensure that there is a mechanism to provide a notice to the Data Principals to enable them to give their consent for the processing of their personal data.

The notice should meet the following requirements.

14.1.1 Notice must be in clear and plain language.

14.1.2 The notice must give an account of the personal data that is being captured and the purpose as well.

14.1.3 The notice should have a mechanism to allow the Data Principals to withdraw their consent and this should be as easy as it was for them to give their consent.

14.1.4 Provide details on the manner in which the Data Principal may make a complaint to the Data Protection Board (Rule 3 of the draft DPDP Rules, 2025 and Section 5(1) of the DPDP Act, 2023).

14.2 Ensure that reasonable security measures are implemented to safeguard the personal data. These measures include:

14.2.1 Securing personal data through encryption, obfuscation or masking or the use of virtual tokens mapped to that personal data.

14.2.2 Implement appropriate access control measures.

14.2.3 Enable log monitoring and review for detection of unauthorized access, its investigation and remediation to prevent recurrence.

(Rule 6 of the draft DPDP Rules, 2025)

14.3 There should be a mechanism to inform the Data Principals in case of a breach of personal data. (Rule 7 of the draft DPDP Rules, 2025).

14.4 There is a mechanism to inform the Data Principals that their personal data will be deleted once the processing of the personal data has been served. (Rule 8 of the draft DPDP Rules, 2025).

14.5 Ensure that the contact details of the Data Protection Officer, if applicable, or a person who can answer on behalf of the Data Fiduciary the questions of the Data Principal about the processing of their personal data are easily available/accessible. (Rule 9 of the draft DPDP Rules, 2025).

14.6 Ensure that there is a mechanism to ensure verifiable consent for processing of personal data of child or of person with disability who has a parent or a lawful guardian. The identity of the parent or lawful guardian shall be verified by:

14.6.1 Reliable details of identity and age available with the Data Fiduciary; or

14.6.2 Voluntarily provided details of identity and age or a virtual token mapped to the same, which is issued by the Central Government or a State Government and includes such details or token verified and made available by a Digital Locker service provider.

14.6.3 (Rule 10 of the draft DPDP Rules, 2025).

14.7 Application should protect sensitive data from being cached in server components such as load balancers and application caches.

14.8 All cached or temporary copies of sensitive data stored on the server are protected from unauthorized access or purged/invalidated after the authorized user accesses the sensitive data.

14.9 Application or underlying infrastructure should facilitate to perform the backups of important data, and test restoration of data.

14.10 Application or underlying infrastructure should detect and alert on abnormal number of requests, such as by IP, user, total per hour or day.

14.11 The data stored in local Storage, session Storage, Cookies on end points should not contain sensitive data.

14.12 The authenticated data such as username, password, Card numbers and anything else used for identification, authentication or authorization should not be stored on the client in any form.

14.13 Sensitive information contained in memory is overwritten as soon as it is no longer required to mitigate memory dumping attacks, using zeroes or random data.

14.14 Personal information of the user is subject to data retention, relevant to the classification, such that old or out of date data is deleted automatically as per the relevant regulations.

15. SECRETS MANAGEMENT

Secrets Management

Secrets are being used everywhere nowadays.

The most well-known examples of secrets are:

- Database Credentials
- LDAP Credentials
- Application Secrets (API Keys, Sensitive data required by the application)
- SSH Keys
- Encryption Keys
- Privileged account credentials.
- Database connection strings

Many organizations have them hardcoded within the source code in plaintext, littered throughout configuration files and configuration management tools. You must standardize and centralize the secrets management solution with care. Standardizing and centralizing can mean that you use multiple secret management solutions. Secrets follow a life cycle. The stages of the lifecycle are as follows:

- Creation
- Rotation
- Revocation
- Expiration

Further guidelines are as below:

- Create a policy that should set strict rules governing the life cycle of secrets while prohibiting the use of default or hardcoded credentials.
- Remove the human element from the secrets management process and rely on automation platforms to create, manage, distribute and maintain secrets.
- Secrets should be changed from time to time to minimize the potential compromise.
- It is the best practice to keep secrets separate from the data to limit the potential of breach.
- Training the development team in the best practices of secret management
- Logging, Auditing and monitoring of secrets access should be maintained.

The following best practices are used for Secrets Management:

15.1 Manage the entire life cycle of secrets, including creation, usage, rotation, and decommissioning.

15.2 Key material is not exposed to the application but instead uses an isolated security module like HSM (Hardware security Module) for cryptographic operations. Hardware security Module (HSM) is a physical device designed to securely manage cryptographic keys and perform encryption, decryption, and other cryptographic operations. It provides a high level of security for sensitive data and is often used in industries requiring strict compliance, such as finance and healthcare.

HashiCorp Vault is a tool for managing secrets and protecting sensitive data. It integrates with HSMs to enhance security by leveraging their cryptographic capabilities.

15.3 Never hardcode secrets in source code or configuration files. Use environment variables or secrets management tools such as HashiCorp vault instead.

15.4 Encrypt secrets both at rest and in transit using strong encryption standards like AES-256.

15.5 Periodically rotate secrets to minimize the risk of compromise.

15.6 Ensure secrets are backed up securely and can be restored in case of an incident.

15.7 Implement the principle of least privilege (PoLP) and role-based access control (RBAC) to restrict access to secrets.

16. BUSINESS LOGIC SECURITY

Business Logic Security

Business logic vulnerabilities arise because design and development teams make flawed assumptions about how users interact with the application. These bad assumptions can lead to inadequate validation of user input.

Application should satisfy the following requirements:

- Business logic flow is sequential, processed, in order and cannot be bypassed.
- Business logic includes limits to detect and prevent automated attacks such as fund transfers, adding million users at a time etc.

The following are the best practices used for Business Logic Security.

16.1 The application will only process business logic flows for the same user in sequential step order without skipping steps.

16.2 The application has appropriate limits for specific business actions or transactions which are correctly enforced per user basis.

16.3 The application has to process business logic flows with all steps being processed in realistic time.

16.4 Implement anti-automation controls (Rate Limiting, CAPTCHA, Throttling) to safeguard applications against abuse from excessive calls which can lead to denial of service.

16.5 Implement mutex locks or synchronization tools to control access to shared resources Ensure that only one process or thread can modify the resource at any given time to prevent the Race condition issue.

16.6 Identify trust boundaries where data comes from or sent to other systems. Verify that inbound data is always validated before being used, no matter where it comes from.

16.7 Application should check for Insufficient process validation by checking for:

16.7.1 Negative payments or funds transfers.
16.7.2 Repeated fund transfers.
16.7.3 Repetition of request where single submission was needed like Vote button or receive payment.
16.7.4 Try to bypass validations or restrictions in early steps by changing values in later steps or bypass early steps.
16.7.5 Money transferred to or from unregistered or other user's bank account.

17. RANDOM VALUES

Random Values

For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between it and a truly random value. In general, if a PRNG algorithm is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts.

The following are the best practices used for Random Values:

17.1 All random numbers, random file names, random GUIDs, and random strings are generated using cryptographically secure random number generator, when these random values are intended to be not guessable by an attacker.

17.2 Random GUIDs are created using the GUID v4 algorithm, and a Cryptographically secure Pseudo-random Number Generator (CSPRNG). GUIDs created using other pseudo-random number generators may be predictable.

17.3 Random numbers are created with proper entropy even when the application is under heavy load, or that the application degrades gracefully in such circumstances.

18. RESTFUL WEB SERVICES

Restful Web Services

A RESTful API is an application programming interface (API) that adheres to the principles of Representational State Transfer (REST). It's a way for applications to communicate with each other, typically over the web, using standard HTTP methods to manipulate resources. This approach allows for more efficient and scalable web services. REST APIs mostly uses JavaScript Object Notation (JSON) files to compress and transfer data from one web application to another. JSON files are small, and therefore easier to transfer. REST is an architectural style designed to be lightweight, scalable, and easy to use. Resources are represented using formats like JSON or XML, making data exchange simple and human readable formats

REST is resource-centric, meaning every resource (like a user, product, or file) is identified by a unique URI (Uniform Resource Identifier).

Ex: https://api.example.com/users/123, : https://api.example.com/product/789

REST uses standard HTTP methods for communication

- GET: Retrieves data from a resource, typically used to fetch information.
- POST: Sends data to create new resources or submit data for processing.
- PUT: Updates or replaces an existing resource with new data
- PATCH: Applies partial updates to a resource.
- DELETE: To remove the resource.
- OPTIONS: Returns the HTTP methods supported by the server for a given resource.
- HEAD: Similar to GET but retrieves only the headers and not the body of the response.

The following are the best practices used for Restful services.

18.1 API URLs do not expose any sensitive information, such as API Key, Session tokens etc. (OWASP, guidelines).

18.2 Restful Services should implement strong authentication mechanisms like OAuth 2.0 or JSON Web Tokens (JWT) for stateless authentication. Please refer to 5.30 and 5.31 for more information regarding JWT and OAuth 2.0.

18.3 Restful web services should use role-based access control (RBAC) to limit access to sensitive resources.

E.g., an "Admin" role might have permissions for all CRUD (Create, Read, Update, Delete) operations on all resources. A “Viewer” role might only have permission to perform GET requests to view resources but not modify them.

18.4 Restful web services should validate/sanitize all the inputs to prevent various injection attacks like cross site scripting (XSS), SQL injections etc. Please refer 10.11 and 10.18 for input validation and output encoding for more information.

18.5 RESTful web services should use the Rate limiting and throttling mechanisms to enhance their security and performance.

All APIs operate on finite resources, and rate limiting is essential to improve the availability of API service for as many users as possible by avoiding excessive resource usages. This helps to prevent malicious activities like Denial of Service (DoS) attacks or resource exhaustion caused by excessive requests.

API throttling is the process of limiting the number of API requests a user can make in a certain period. The API throttling logic checks if the current request exceeds the allowed number of API calls. If the request is within limits, the API performs as usual and completes the user's task. If the request exceeds the limit, the API returns an error response to the user.

18.6 Restful web services should log the API activities, and these API activities can be monitored for the suspicious behaviour to detect and respond to potential threats.

18.7 Ensure that only authorized clients can access the API end points by using IP whitelisting or API gateways.

18.8 Any HTTP method not explicitly allowed should be rejected with appropriate status codes like 405 Method Not Allowed. Restricting HTTP methods ensures that only the intended actions are allowed on specific API endpoints. Apply allowlist of permitted HTTP methods (ex: GET, POST, PUT). Make sure that caller is authorized to use the incoming HTTP method on the resource collection, action, and record.

18.9 Restful Services should use HTTPS to encrypt data in transit safeguarding against eavesdropping and man in the middle attacks by ensuring secure communication between client and servers.

18.10 REST services should validate the Content-Type of incoming requests to ensure they match the expected format. The requests containing unexpected or missing content types are rejected with appropriate headers (HTTP response status 406 Not Acceptable or 415 Unsupported Media Type) (NIST Standard 800-204).

Example:

- 415 Unsupported Media Type is returned when server does not support the format specified in the content-type header of the request. If the client sends text/plain, but server only accepts application/json, then the request is rejected with a 415 error.
- 406 Not Acceptable is used when the server cannot provide a response in a format that matches the client's specified in the Accept header. If client requests application/xml, but the server only supports application/json, the server may return 406 Not Acceptable.

18.11 Restful Services should implement authorization decisions commonly made at two levels

a. Field-Level Authorization: Once access to a resource is granted, this level controls what parts of the resource are visible or editable.

Example: A regular user might only view non-sensitive fields, while sensitive data like financial or personal details is restricted. An admin might have access to all fields, including sensitive ones.

Regular User/Normal User

```
Json
{
 "name": "Mohan Krishan",
 "email": "mohan.krishan@example.com",
 "financial_details": "Restricted"
}
```

Admin User

```
Json
{
 "name": "Shiv Sagar",
 "email": "shiv.sagar@example.com",
 "financial_details": "Bank Account: ****1234"
}
```

b. Resource-Level Authorization: This ensures that only authenticated and authorized users can access specific resources.

Example: A regular user might only have GET access to their own data. An admin user might have access with POST, PUT, DELETE permissions across all resources.

18.12 API versioning is the process of managing changes to an API over time while ensuring compatibility for its consumers. This helps in tracking, managing, and auditing APIs effectively. Key components are:

- Version Tracking: Keep a record of all API versions and their status (active, deprecated, retired)
- Access Control: Monitor who has access to which APIs to ensure security
- Usage Metric: Track API usage to identify popular endpoints and optimize performance

18.13 Application should use DDoS mitigation services to protect critical APIs and other digital assets from distributed denial-of-service (DDoS) attacks.

19. SOAP WEB SERVICES

SOAP Web Services

SOAP (Simple Object Access Protocol) web services are a type of communication protocol used for exchanging structured information in a decentralized, distributed environment. SOAP is protocol-driven, designed to enable communication between applications over the internet or within a network. SOAP uses XML for its message format, ensuring that data is platform-agnostic and can be understood across different systems. SOAP defines strict message formats and schemas, making it robust but less flexible compared to REST.

The following are the best practices used for SOAP based services:

19.1 SOAP based services should implement robust **authentication** to ensure only authorized entities can access and interact with the service.

Below are the authentication standards used in SOAP based services. (NIST SP 800-95, Section 5)

19.1.1 **Username/password** Authentication.

19.1.2 **Token Based** Authentication (Tokens such as OAuth or JWT). Please refer to 5.32 and 5.33 for more information.

19.1.3 **Mutual TLS (mTLS)**/ Certificate Based Authentication.

19.1.4 **Web Services Security (WS-Security)** a standard designed to secure SOAP-based web services. It provides mechanisms for ensuring confidentiality, integrity, and authentication of messages exchanged between services. (NIST SP 800-95, Section 3.1.2)

- Message integrity: Ensure that message has not been tampered by using digital signature.
- Message confidentiality: Protect sensitive data by encrypting parts of SOAP message.
- Authentication: Verifies the identity of the sender using security tokens like X.509 certificates, Kerberos tickets or SAML assertions.

19.2 SOAP-based web services should use **HTTPS** to encrypt data in transit (NIST SP 800-95, Appendix A)

19.3 The application should first check the **XML document** against the **XSD** to ensure it is **well-formed** and compliant with the schema. This step ensures that the document's structure, data types, and relationships align with the rules defined in the XSD.

19.4 The application should not accept **user defined XML tags**. Allowing user-defined XML tags could potentially pose security risks, such as **XML Injection** attacks or unexpected behaviour in the application.

19.5 Web services need to **validate input before consuming** it. Following content validation for XML input should be included. (NIST SP 800-95, Section 3.6.4)

19.5.1 Validation against **malformed XML entities**.

19.5.2 Validation against **XML Bomb attacks**.

19.5.3 Validated against external entity (**XXE)attacks**.

19.5.4 Input uses **strong allowlist**.

19.6 Web services must provide the following validations to protect from DoS attacks. (NIST SP 800-95, Section 3.6.4)

19.6.1 Against recursive payloads (nested XML tags).
19.6.2 Against huge XML size (large XML file sizes).
19.6.3 Against overlong element names.
19.6.4 Rate Limiting: Restrict the number of requests a user or system can send within a given time frame to avoid abuse. When a request exceeds the limit, the server should respond with an appropriate status code, such as 429 Too Many Requests.
19.6.5 Resource Throttling: Limit the consumption of server resources (e.g., CPU, memory) for individual requests to mitigate the impact of resource-exhaustion attacks.

19.7 Verify that all application components use the same encodings and parsers to avoid parsing attacks that exploit different URI or file parsing behaviour that could be used in **SSRF and RFI** (Remote File Inclusions) attacks.

19.8 Make sure SOAP based API URLs do not expose **sensitive information**, such as API keys, session tokens etc.

19.9 Verify the SOAP based request headers should check the **Content-Type**, which ensures that conformity to expected standards and prevents any bypass of security controls. By checking the Content-Type header (OWASP guidelines)

(e.g., application/soap+xml for SOAP messages), it confirms that the incoming request adheres to the protocol's standards.

20. MOBILE APPLICATION STANDARDS

Mobile Application Standards

Mobile application security standards are essential for protecting sensitive user data and ensuring the integrity of mobile applications. These standards are designed to guide developers, security professionals, and organizations in building and maintaining secure mobile applications.

The following are the best practices used for Mobile Applications:

20.1 Mobile applications should not store sensitive information (PCI/PII Information) in their directory structure or files after installation. This is a critical security measure to protect user data, application integrity, encryption keys and server-side resources from being exposed to unauthorized access.

20.2 After running a mobile application, the directory structure and files should never contain any sensitive information, such as PCI (Payment Card Information), PII (Personally Identifiable Information), developer names, encryption keys, or other sensitive content. Storing such data in directories or files could lead to unauthorized access or data breaches. Avoid storing sensitive information locally. Ensure that any temporary files or data generated during runtime are securely deleted after use.

20.3 Application should implement SSL pinning, allowing secure communication between the mobile application and its server. It ensures that the application only communicates with a trusted server by embedding a copy of the server's certificate (or public key) within the mobile application's code. SSL pinning prevents the man-in-the-middle (MITM) attacks. (NIST Special Publication 800-163 Revision 1)

20.4 Application should not hardcode the sensitive information such as passwords, file paths, and encryption keys in APK or IPA files poses a significant security risk. This can expose application to vulnerabilities, making it easier for attackers to reverse-engineer the files and gain unauthorized access to the systems or user data.

20.5 Use code obfuscation tools (E.g., ProGuard, DexGuard, iXGuard) to make it harder for attackers to reverse-engineer the application. Code obfuscation makes the source code harder to understand, which can help to protect it from being reverse engineered by attackers.

20.6 Application should not use the vulnerable libraries, third party components or frameworks, these vulnerable components can be exploited if they are not properly vetted or updated. The vulnerabilities in third party software libraries, SDKs allows gaining unauthorized access to mobile applications or backend servers.

20.7 Applications should not use inadequate or missing authorization schemes which can lead to security vulnerabilities. If low-privileged users can access high-privileged user information by manipulating session tokens, this can result in unauthorized data access and potential data breaches.

20.8 Weak authentication schemes can indeed pose significant security risks. Applications should avoid using authentication methods that allow anonymous users to access the mobile application or backend servers. Such weaknesses can arise from mobile device limitations, which sometimes lead to the use of short passwords or PINs. These methods are generally easier for attackers to compromise.

20.9 Ensuring proper input validation and sanitization in mobile applications is crucial to preventing a variety of security threats, including command injection, SQL injection, and cross-site scripting (XSS) attacks. Input validation ensures that input provided by the users meets the expected format, length, and type. It also removes or neutralizes any malicious characters or code from the user input to prevent malicious actions before it is processed by the application. Encoding the output can prevent injection attacks by transforming potentially harmful characters into a safe format. Refer to 10.11 for more information.

20.10 Application should use SSL/TLS for data transmission to backend services. It should implement strong, industry-standard cipher suites with appropriate key lengths. Application should implement SSL pinning, SSL chain verification. Application should use certificates signed by trusted certificate authorities (CA). Refer section 4 for more information.

20.11 Application should disable the debugging feature for android applications.

20.12 Application should disable backup mode on Android to prevent sensitive data from being included in the device backups.

20.13 Mobile applications should export only necessary activities, content providers and services.

20.14 Mobile applications should provide only the necessary permissions to ensure the application functions correctly, while minimizing the risk of unauthorized access or modifications. No unauthorized user or application on the device can access (read) or modify (write) the data.

20.15 Mobile applications should prevent snapshot caching by implementing measures that obscure sensitive content whenever the application moves to the background, such as when a user presses the home or lock button. This ensures that sensitive content does not appear in system-generated screenshots or thumbnails visible in the app-switcher or multitasking view.

20.16 Application should not log the sensitive information such as user credentials, account numbers (PCI/PII data), requests/ responses, detailed error messages in the device logs. Unauthorized access to these logs could lead to data breaches or misuse.

20.17 Mobile applications should disable keyboard cache files and user dictionary content, as these can pose security risks if sensitive information is inadvertently stored or exposed. Ensure sensitive data entered in text fields cannot be added to the user dictionary for autocorrection or predictive text.

For sensitive input fields use secure settings for IOS and Android applications

- On IOS,
    - Enable secureTextEntry, ensures that sensitive information, such as passwords, is obscured in the text field.
    - autocorrectionType=UITextAutocorrectionTypeNo,

disables autocorrection for the text field.

- On Android,
    - Use inputType= “textPassword”, the field behaves like a password field. It ensures that characters typed into the field are obscured.
    - android:importantForAutofill="no".

This disables the autofill functionality for sensitive fields.

20.18 Mobile applications should disable the "Cut" and "Copy" functionality for sensitive fields in mobile applications. This prevents sensitive information, such as passwords, PINs, or credit card details, from being inadvertently copied to the clipboard, where it can be accessed by other applications or processes.

- On Android
    - Use Android:importantForAutofill=”no” This disables autofill functionality, preventing sensitive data in these fields from being stored or accessed during autofill operations.
    - Use android:longClickable="false" prevents actions like copy, paste, or sharing of sensitive content via the long-click functionality.

    - Even if copy-paste functionality is disabled, ensure that sensitive information isn't stored or left on the clipboard. On Android, clearing clipboard content for sensitive fields is an additional safeguard.

20.19 Memory dumps can pose significant security risks for mobile applications, especially if sensitive data like user credentials or application content is exposed. Avoid storing sensitive data (e.g., passwords, tokens) in memory for longer than necessary. Encrypt sensitive data before storing it in memory to ensure it remains protected, even if a memory dump occurs.

20.20 Mobile applications should overwrite sensitive data in memory as soon as it is no longer needed. Use platform-specific secure memory management techniques.

20.21 Mobile applications should not store sensitive information (e.g., user credentials, tokens) in temporary files, especially on external storage like SD cards, as it may not be secure. If temporary files are necessary, ensure they're encrypted before storage.

20.22 Mobile applications should give a warning message if the application is installed on the rooted device.

20.23 Mobile applications that handle sensitive information should restrict third-party keyboards to reduce the risk of data leakage. Some third-party keyboards might collect keystrokes, posing a security risk.

- On iOS, use textInputMode.primaryLanguage to detect and restrict third-party keyboards for specific input fields.

- On Android, it is more challenging, but applications can restrict custom keyboards for certain fields by implementing secure text input behaviours.

20.24 Mobile applications should not be installed on outdated device versions. Setting minimum requirements for the operating system (OS) and hardware is essential to enhance security by preventing usage on devices that lack critical updates or patches.

20.25 Use the following attributes to ensure that the content in the field is not editable, which can be useful for sensitive fields where user input is not required.

- android:focusable="false": Prevents the field from gaining the focus.
- android:clickable="false": Ensures the field does not respond to click events.
- android:inputType="none": Disable the input method, such as keyboard.

In addition to the above, use TextView instead of EditText for displaying the non-editable content.

21. APPLICATION HOSTING

Application Hosting

Application Hosting involves ensuring that hosted applications are protected against potential threats and vulnerabilities throughout their life cycle. Third party software components need to be updated from time to time. Failure to keep up to date with outdated or insecure dependencies is the root cause of the largest and most expensive attacks to date. It is recommended that applications should use the latest versions of software components. (OWASP guidelines)

The following are the best practices used for Application Hosting:

21.1 The application should not use vulnerable versions of the open-source components. Verify that a Software Bill of Materials (SBOM) is maintained for all third-party libraries in use. Verify, all components are up to date, preferably using Software Composition Analysis (SCA) tools during build or compile time. (OWASP guidelines)

21.2 Verify that third party components come from pre-defined, trusted and continually maintained repositories.

21.3 Verify that the attack surface is reduced by sandboxing or encapsulating third party libraries to expose only the required behaviour into the application. Exposing only required behaviour ensures that unused or risky functionalities of the libraries are not accessible, further mitigating potential risks.

21.4 Application should not use the unsupported versions of licensed software components, which can pose significant security risks, as these versions no longer receive updates, including critical security patches. (OWASP guidelines).

21.5 Licensed software components should be patched at regular intervals in accordance with the organization's policy to mitigate security risks and ensure software compliance. (OWASP guidelines).

21.6 Application should disable the default content, such as sample files, test pages, or configurations, may unintentionally expose sensitive information about the application server or software, making it vulnerable to attacks.

21.7 Default account credentials are often publicly available or easily guessed. Application should disable these default credentials.

21.8 Application should disable the following server-side files.

21.8.1 Common configuration files such as

- web.xml,
- robots.txt
- global.asax
- web.config

- .htaccess
- httpd.conf
- applicationHost.config

21.8.2 Alternate versions of the files by changing the file extensions. Configure the servers explicitly allow only known and trusted file extensions.

21.8.3 Application Test script files such as test.aspx.

21.9 Application files and directories should not be accessible directly.

21.10 The application should not reveal any hidden directories of the application. Application should respond with 404 error code or redirect to login page while accessing hidden directories. This approach prevents attackers from distinguishing between valid and invalid directories.

21.11 The following conditions should be checked for web server.

21.11.1 server-status should not be enabled. module can expose sensitive information about server performance, uptime, and active requests, which could be valuable to an attacker.

21.11.2 server-info should not be enabled. It can reveal details about the server's configuration, such as loaded modules and their settings

21.12 Applications should prevent the use of the following insecure HTTP methods. Verify that only secure and appropriate methods, such as GET, POST, or other valid choices, are enabled for specific users or actions. The following methods should generally be disabled as they may pose security risks.

- OPTIONS
- TRACE
- TRACK
- PATCH
- PUT
- DELETE
- CONNECT
- PROPFIND
- COPY
- DEBUG

22. LOGGING AND AUDITING

Logging and Auditing

The primary objective of logging is to provide useful information for the privileged user / administrators, and incident response teams. The objective is not to create massive amounts of logs, but high-quality logs, with more signal than discarded noise. Logs must be protected from tampering and unauthorized access.

The following are the best practices used for Logging and Auditing:

22.1 Application should not log any sensitive information PCI/PII data, Session tokens, password, security question and answers etc.

22.2 User authentication details are logged in the log file, without storing sensitive session tokens or passwords.

22.3 Application should use common logging format, and approach is used across the system.

22.4 Application should log all critical activities performed by the user. Some of the information are: (OWASP guidelines)

- Source IP address
- Time stamp
- Login and logout activity
- User identity (username)
- User agent
- Authentication success/ failures
- Authorization access control failures
- Session Management Failures
- Application errors and system events
- Application and related systems start-ups/ shutdowns
- Input validation failures
- Output validation failures such as invalid data encoding etc.
- Critical activities performed by the user

22.5 High risk functionalities should be logged in audit trail and log file. This practice is essential for accountability, detecting unauthorized activities, troubleshooting issues and complying with regulatory requirements. Audit trails serve as a tool for administrators to view and review the activities performed within the system. This helps ensure transparency and accountability. Below are some of the examples. (NIST SP 800-12, Chapter 18).

- User admin functionalities such as addition, modification and deletion of users, changes to privileges, assign users to tokens, adding / deleting of tokens
- Access to Sensitive data such as payment cardholder data
- Use of default or shared accounts
- Encryption-related activities, such as the use, generation, or rotation of cryptographic keys
- Data import and export including screen-based reports
- Submission and processing of file uploads

- Network connections and associated failures
- Important privileged activities such as administrative tasks like modifying configurations, managing user accounts, accessing restricted files, or executing commands at a high system level
- User profile information addition/ modification

22.6 Application logs are securely transmitted to a preferably remote system for analysis, detection, alerting and escalation. This approach ensures logs are protected from tampering or loss due to local system failures or breaches. Implementing encryption during transmission and access control for remote log storage adds extra layer security.

22.7 Application logs are securely protected from unauthorized access and modification by implementing Restrict access, monitor access and implement immutability.

22.8 Configure the application to send logs to the SIEM. SIEM tools allow organizations to collect, analyse, and correlate logs from various sources to detect threats and ensure compliance.

22.9 Review Logs and security events for all system components to identify anomalies or suspicious activity. Perform critical log reviews at least daily.

22.10 Retain audit trail history as per the organization policy. This is essential for compliance, accountability and security.

23. ERROR HANDLING

Error Handling

The purpose of error handling is to allow the application to provide security relevant events for monitoring, triage and escalation. When logging security related events, ensure that there is a purpose to the log. One common security problem caused by improper error handling is the fail-open security check.

Error handling should not focus solely on input provided by the user but should also include any errors that can be generated by internal components such as system calls, database queries, or any other internal functions.

The following are the best practices used for Error Handling:

23.1 Generic error message is shown to user when an error occurs, potentially with a unique ID which support personnel can use to investigate. (OWASP Application Security Verification Standard (ASVS))

23.2 Exception handling is used across the code base to account for expected and un excepted error conditions. Detailed error messages should be logged in log file with a unique ID and Timestamp. (OWASP guidelines)

24. QUANTUM SAFE CRYPTOGRAPHY

Quantum Safe Cryptography

Quantum computing is rapidly evolving. Quantum Computers pose a threat to current encryption methods. NIST has released Post Quantum Cryptography (PQC) standards designed to secure electronic information against post quantum computer attacks.

The following are the best practices used for Quantum Safe Cryptography. NIST is working on and periodically releasing algorithms related to quantum computing, specifically focusing on developing post-quantum cryptography (PQC) standards to protect against potential attacks from quantum computers.

24.1 Application should use Stateless Hash-Based Digital Signature Standard (SLH-DSA) to detect unauthorized modifications to authenticate the identity of the signatory.

OR

Application should use Module Lattice Based Digital Signature Standard (ML-DSA), which generates and verify digital signatures.

24.2 Application should use Module Lattice Based Key Encapsulation Standard (ML-KEM), which transmits cryptographic keys using lattice-based algorithms.

ANNEXURE A: ABBREVIATION

API	Application Programming Interface
CDR	Content Disarm and Reconstruction
CSRF	Cross Site Request Forgery
CVE	Common Vulnerabilities and Exposures
DPDP	Digital Personal Data Protection
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
HSM	Hardware security Module
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
MFA	Multi Factor Authentication
NIST	National Institute of Standards and Technology
NTLM	NT LAN Manager
OWASP	Open Worldwide Application Security Project
PAN	Permanent Account Number
PCI	Payment Card Information
PII	Personally Identifiable Information
PKCE	Proof Key for Code Exchange
PBKDF2	Password-Based Key Derivation Function 2
RBAC	Role Based Access Control
REST	Representational State Transfer
SIEM	Security Information and Event Management
SSL	Secure Sockets Layer
SSO	Single Sign On
SSRF	Server-side request Forgery
TLS	Transport Layer Security
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
VAPT	Vulnerable Application Penetration Testing
WADL	Web Application Description Language
WAF	Web Application Firewall
WSDL	Web Services Description Language
XSS	Cross Site Scripting

ANNEXURE B: REFERENCES URLS

1) https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf
2) https://pages.nist.gov/800-63-3/sp800-63b.html
3) https://owasp.org/www-project-application-security-verification-standard/
4) https://owasp.org/www-project-enterprise-security-api/
5) Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations
6) OWASP Cheat Sheet Series(https://cheatsheetseries.owasp.org/)
7) OWASP Application Security Verification Standard 4.0.3

ANNEXURE C: APPLICATION TYPES

SNO	Application Classification Type	Classification Description	Chapters Applicable (Control Numbers)
1	Class 1	Internet Based Application. Handles PCI/ PII Data	All controls are applicable
2	Class 2	Internet Based Application. Does Not handle any PCI/PII Data	2.1, 2.6 To 2.11, 2.13 To 2.15, 3.1 To 3.5 , 4.1 To 4.20, 5.1 To 5.8, 5.10 To 5.31, 6.1 To 6.7, 7.1 To 7.6, 7.9 8.1 To 8.11, 9.1 To 9.18, 10.1 To 10.22, 11.1 To 11.6, 12.1 To 12.13, 13.1 To 13.11, 15.1 To 15.7, 16.1 To 16.7, 17.1 To 17.3, 18.1 To 18.12, 19.1 To 19.10, 20.1 To 20.26, 21.1 To 21.11, 22.1 To 22.10, 23.1 To 23.2, 24.1 To 24.2
3	Class 3	Intranet Based Application. Handles PII Data.	2.1 To 2.15, 3.1 To 3.5, 4.1 To 4.20, 5.1 To 5.33, 6.1 To 6.7, 7.1 To 7.9, 8.1 To 8.11, 9.1 To 9.18, 10.1 To 10.22, 11.1 To 11.6, 12.1 To 12.13, 13.1 To 13.11, 14.1 To 14.15, 15.1 To 15.7, 16.1 To 16.7, 17.1 To 17.3, 18.1 To 18.12, 19.1 To 19.10, 20.1 To 20.24, 21.1 To 21.12, 22.1 To 22.10, 23.1 To 23.2, 24.1 To 24.2.
4	Class 4	Intranet Based Application. Does not handle any PII Data.	2.6, 2.10 To 2.13, 4.10, 5.1 To 5.3, 5.20, 6.1 To 6.5, 8.3 To 8.5, 9.5, 10.1 To 10.3, 10.6, 10.15, 10.18, 11.1 To 11.6, 12.1 To 12.13, 13.1 To 13.11, 16.1 To 16.7, 18.1 To 18.12, 19.1 To 19.6, 21.1 To 21.7, 22.2

ANNEXURE D: APPLICATION SECURITY TOOLS

In the context of Secure Software Development Life Cycle (SSDLC), the Shift Left approach emphasizes incorporating security practices from the beginning of the development process. This includes activities like static code analysis, security testing, and continuous security validation throughout the development stages.

As part of shift left approach implementation SSDLC checklist, SAST, DAST, IAST SCA tools can be incorporated to find the vulnerabilities before the application goes into UAT environment. Only tests specially written to detect security vulnerabilities or unexpected behaviours will reveal potential flaws, this is where SAST and DAST tools come into play. This testing method plays a critical role in identifying and mitigating the potential security vulnerabilities before these security vulnerabilities are exploited.

1) SAST (Static Application Security Testing)

This tool analyses the source code of the program to identify the security issues. SAST tool will analyse the source code, bytecode, or binary code without running the program. This tool allows developers to identify the security issues in the code early in development cycle, before code deployment happens. SAST tools can detect many of the vulnerabilities listed in the OWASP TOP 10, SANS Top 25.

SAST should be run as soon as the code is committed, it means code can be detected and highlighted with to the developer who committed the code. SAST tools can be integrated into development workflow to proactively address security concerns, improve the code quality and reduce the security breaches.

2) DAST (Dynamic Application Security Testing)

Dynamic Application Security Testing (DAST) is a method used to identify vulnerabilities in web applications by simulating real-world attacks. Unlike static testing, which examines source code, DAST evaluates applications in their runtime environment, making it a "black box" testing approach. It mimics the behaviour of malicious users to uncover issues like SQL injection, cross-site scripting (XSS), and authentication flaws.

DAST tools are particularly useful because they don't require access to the application's source code, making them versatile for testing various applications. They are often integrated into the software development lifecycle (SDLC) to catch vulnerabilities early, reducing the cost and effort of fixing them later.

If you're considering implementing DAST, it can be a valuable addition to security toolkit, especially when combined with other methods like Static Application Security Testing (SAST) for a comprehensive approach.

3) Software Bill of Materials (SBOM)

SBOM is a comprehensive list of all the components, libraries, and dependencies that make up a software application. It provides detailed information about the software's ingredients, including their versions, licenses, and relationships. SBOMs are essential for ensuring transparency, security, and compliance in the software supply chain.

The primary goal of an SBOM is to offer transparency into the software supply chain. It helps organizations understand what components are used, their origins, and any associated risks. This transparency is crucial for managing security, compliance, and operational risks.

The integration of SBOM (Software Bill of Materials) with VEX (Vulnerability Exploitability eXchange) is a powerful approach to managing vulnerabilities effectively.

SBOM provides a detailed inventory of all components in a software application, including dependencies and their versions.

VEX (Vulnerability Exploitability eXchange) complements SBOM by assessing whether identified vulnerabilities are exploitable in the specific context of the software. Not all vulnerabilities pose a risk, and VEX documents clarify this, reducing unnecessary patching efforts.

An SBOM typically includes the following

Component Name: The name of the software component.

Version Information: The specific version of the component.

Licenses: The licensing information for the component.

Dependencies: Details about other components that the given component depends on.

Hash Values/ Checksums: Used to verify the integrity of each component.

Author/Supplier Information: Details about the developer or vendor of each component.

Below are the key points about SBOM:

- SBOMs provides a clear view of all the open source and third-party components used in a software application.
- By listing all components, SBOM helps us to identify the address vulnerabilities more quickly.
- Assists in managing risks associated with the software supply chain by providing visibility into component provenance and dependencies.
- Helps ensure compliance with regulatory requirements and industry standards by providing detailed licensing information.

SCA tools (Software Composition Analysis tools) often use SBOMs (Software Bill of Materials) as input to perform vulnerability analysis for open-source and third-party components. Monitoring these components periodically for vulnerabilities is a vital security practice, ensuring that applications remain secure and up to date.