CompTIA SecurityX (CAS-005) Study Notes

Introduction

- **Introduction**
  - SecurityX Overview
    - Part of the CompTIA Xpert Series certifications
    - Previously known as CompTIA Advanced Security Practitioner (CASP+)
    - Other certifications in the Xpert Series include DataX and CloudNetX
  - Target Audience
    - Designed for senior security engineers and architects
    - Intended for professionals focused on technical work rather than managerial roles
    - Validates advanced-level competency in:
      - Risk management
      - Enterprise security operations and architecture
      - Research and collaboration
      - Integration of enterprise security in organizational networks
  - Skills Validated by SecurityX
    - Ability to conceptualize, engineer, integrate, and implement secure solutions across complex environments
    - Focus on supporting resilient enterprise systems
  - Experience Requirements

        - At least 10 years of IT experience
        - Five years of broad, hands-on security experience
    - Exam Recommendations
        - Security+, CySA+, and PenTest+ exams recommended but not required
    - Exam Content Overview
        - Builds on knowledge from lower-level certifications such as Security+ and Network+
        - Focuses on advanced topics in computer security, cybersecurity analysis, and penetration testing
    - Exam Structure
        - Four domains covered in the SecurityX exam:
            - Governance, Risk, and Compliance (20%)
                - Focus on managing organizational risk, ensuring regulatory compliance, and governance practices
                - Major topics include GRC strategies, threat modeling, and artificial intelligence challenges
        - Security Architecture (27%)
            - Design of resilient systems and secure infrastructures
            - Topics include firewalls, IDSs, IPSs, VPNs, secure architecture, enterprise cloud implementation, and zero trust architecture
        - Security Engineering (31%)
            - Focuses on troubleshooting and securing systems
            - Includes identity and access management, endpoint protection, network infrastructure security, and cryptographic techniques
        - Security Operations (22%)
            - Continuous monitoring, analysis, and response to security events

        - Topics include SIEM, vulnerability analysis, threat intelligence, incident response, malware analysis, and threat hunting
- Objectives and Exam Questions
    - 23 exam objectives under the four domains
    - The exam has a maximum of 90 questions
    - Questions include multiple choice, multiple select, and performance-based style questions
    - Expect 3 to 5 performance-based questions
- Scoring
    - Exam results display only as pass or fail
    - Practice exams with scores of 80%-85% or higher are recommended for success
- Exam Voucher
    - Can be purchased from CompTIA or at a discounted rate from diontraining.com/vouchers
    - Vouchers are valid for 11 months after purchase
- Instructor Introduction
    - Jeremiah Minner, instructor with Dion Training
    - Professional background in teaching certification courses and managing network and security systems
- Tips for Success
    - Closed Captions
        - Enable closed captions for better comprehension, especially for non-native English speakers
    - Playback Speed

        - Adjust playback speed to match learning preferences (faster or slower)
      - Downloadable Study Guide
        - Study guide available as a PDF in lesson 2 of the course
        - Download and print exam objectives as a checklist for offline study
      - Student Support Groups
        - Join Facebook or Discord groups for additional support
        - Participate in discussions and get quick answers from fellow students
  - Q&A Section
    - Post questions in the Q&A tab for a response from the Dion Training team within 24-48 hours
    - Facebook and Discord may provide faster responses due to a larger student presence
  - Course Structure
    - Objectives covered in each section are clearly labeled with their domain and objective number
    - Example:
      - "Change and Configuration Management (OBJ 1.1)" indicates content from Domain 1, Objective 1.1

- **Exam Tips**
  - No Trick Questions
    - All questions are precisely worded. No "gotcha" questions, so focus on reading carefully to understand what's being asked

- Read the Questions Carefully
    - Read each question multiple times to ensure you understand exactly what is being asked before selecting an answer.
    - Pay attention to every detail and focus on answering the question being asked
- Beware of Distractors (Red Herrings)
    - At least one of the four answer choices is likely a distractor. Eliminate obvious distractors to increase your chances of selecting the correct answer
- Pay Attention to Highlighted Words
    - Words in bold, italics, or ALL CAPS are key to understanding the question's focus. Pay close attention to these, as they are often crucial in guiding you to the right answer
    - Examples:
        - ALL
        - AND
        - OR
        - NOT
        - BEST
        - SELECT TWO
        - SELECT THREE
- Rely on Your Course Knowledge
    - Answer based on the course material and exam objectives, not your personal work or school experiences. Workplace practices might differ from the exam's standard answers
- Select the Best Answer

    - Multiple answers may be technically correct, but always choose the one that is most correct in general situations. Avoid overthinking or considering fringe cases
- General vs. Specific Answers
    - If one answer is a general category and another is a specific example related to the question, the specific example is likely the better choice
- Identify the Key Concept
    - Focus on understanding the core concept of the question. Once you identify the key idea, it becomes easier to eliminate wrong choices and find the correct one
- No Need for Exact Definitions
    - You don’t need to memorize definitions verbatim. You just need to recognize terms and concepts in the answer choices
- Hands-On and Scenario-Based Focus
    - 74% of the objectives in SecurityX are scenario-based and emphasize tool implementation over simple identification. Be ready to apply your knowledge in real-world scenarios.
- Multiple-Choice, No Fill-in-the-Blank
    - The exam is all multiple-choice or multiple-selection. You are selecting from pre-written answers, not filling in blanks or writing essays
- Recognize, Don’t Regurgitate
    - Unlike academic tests, certification exams focus on recognizing the correct answer, not on recalling exact definitions or lengthy explanations

Governance

Objective 1.1: Implement appropriate governance controls

- **Security Program Documentation**
    - Security Program Documentation
        - Outlines the foundational approach to managing and protecting information systems
    - Administrative Controls
        - Designed to regulate the behavior of employees and ensure adherence to security and operational standards
        - Further focus on managing human and operational risks
        - Policies, procedures, standards, and guidelines are all considered administrative controls and are essential to an organization because they provide a clear framework for consistent decision-making and best practices to meet regulatory requirements
            - Policies
                - Establish high-level principles and expectations for security
                - Define what is allowed and expected behavior
                - Required to be followed policies and there are often consequences for not following them
                - Examples:
                    - Separation of Duties
                        - Helps prevent fraud by ensuring no single person has control over all aspects of a process

- Split Knowledge
    - A variant of separation of duties in which two individuals each hold half of the information needed for a task
    - Includes breaking a cryptographic key into two parts and assigning each part of the key to different administrators

■ Job Rotation

- Involves training multiple employees to perform the same tasks
    - Helps identify fraud
    - Provides backup in emergencies
    - Allows cross-training to enhance resilience

■ Mandatory Vacation

- Policies that require employees to take time off
    - Job duties are temporarily assigned to others
    - Helps uncover any suspicious activity

○ Key Benefit

■ Establishes clear, mandatory principles and expectations, ensuring consistent behavior and decision-making

- Procedures

    - Provide detailed, step-by-step instructions for implementing policies
    - Examples
        - Incident Response Procedures
            - Notifying the incident response team
            - Documenting the details of the breach
            - Following a specific communication plan to inform relevant stakeholders
        - Backup and Recovery Procedures
    - Key Benefit
        - Eliminates of ambiguity, providing clear instructions that reduce the likelihood of errors or misinterpretation
- Standards
    - Define specific requirements that should be met, but they don't have the enforcement that laws and regulations do
    - Often followed as best practices
    - Some standards, though, do have penalties associated with noncompliance
    - Examples
        - Payment Card Industry Data Security Standard (PCI-DSS)
            - Contractual agreement that any organization handling credit card information must follow
        - International Organization for Standardization (ISO)

- Provides a series of standards for industries
  - ISO 27000 Series
    - Focuses on information security management

- Capability Maturity Model Integration (CMMI)
  - Process improvement model used in the development of software, products, and services
  - Assesses organizational maturity level from one to five
    - Higher levels indicate more refined and optimized processes
    - Government contracts
      - Often require a specific maturity level, such as three, four, or five
- National Institute of Standards and Technology (NIST)
  - Provides over 1,300 standards for industries
    - NIST 800 Series
      - Provides important standards for information security
  - Not legally binding like PCI DSS, but critical for ensuring security and operational efficiency in various sectors

- Key Benefit

            - Provides clear, uniform requirements and best practices, ensuring consistency, quality, and compliance
    - Guidelines
        - Offer recommended best practices to help achieve compliance and alignment with organizational policies and standards
        - Flexible and not mandatory
        - Examples
            - Firewall configuration guidelines
                - Enhance network security
            - Suggested encryption protocols
                - Secure sensitive data
            - Password management
                - Suggestions for multi-factor authentication and complexity
        - Key Benefit
            - Flexibility allows organizations to tailor practices without the strictness of policies or procedures

- **Awareness and Training Considerations**
    - Awareness and Training Considerations
        - Part of security program management focused on educating employees about security risks like phishing, social engineering, physical security, and privacy practices.

        - Through initial and ongoing training, employees build situational awareness and support operational security (OPSEC) by recognizing and avoiding security threats
    - Components of Awareness and Training Considerations
        - Operational Security (OPSEC)
            - Definition
                - Processes and practices designed to protect sensitive information from being inadvertently exposed
            - Purpose
                - Helps mitigate risks by identifying critical data and controlling its dissemination
            - Example
                - Limiting access to confidential data only to authorized personnel to prevent accidental leaks
            - Mitigation Strategies
                - Implement data classification and access control policies
                - Educate employees on recognizing and safeguarding critical information
        - Physical Security
            - Definition
                - Protecting an organization's assets, facilities, and personnel through measures like surveillance, secure entry points, and access control
            - Purpose
                - Ensures the safety of infrastructure and personnel by preventing unauthorized access or harm

- Example
  - Training employees to recognize physical security threats, such as tailgating or unauthorized access
- Mitigation Strategies
  - Install surveillance and access control systems
  - Conduct regular physical security training for employees

▪ Privacy
- Definition
  - Safeguarding personal and sensitive information from unauthorized access and ensuring secure data handling
- Purpose
  - Prevents unauthorized disclosure of personal information
- Example
  - Protecting customer data from unauthorized access to comply with legal standards
- Mitigation Strategies
  - Train employees on privacy best practices and data handling policies
  - Enforce data protection regulations, such as GDPR or HIPAA, within the organization

▪ Social Engineering
- Definition
  - Manipulative tactics used by attackers to trick individuals into revealing confidential information, often exploiting human psychology
- Purpose

        - Highlights the importance of awareness and critical thinking in interactions that may be manipulated by attackers
    - Example
        - Teaching employees to recognize pretexting, where attackers pose as trusted sources to gather information
    - Mitigation Strategies
        - Conduct social engineering training sessions with examples of common tactics
        - Simulate social engineering attacks, like phishing exercises, to reinforce learning
- Phishing
    - Definition
        - A type of manipulation where attackers use fraudulent communications, typically emails, to deceive individuals into taking unsafe actions
    - Purpose
        - Increases vigilance as phishing is a common attack vector
    - Example
        - Showing employees examples of phishing emails and how to verify suspicious communications
    - Mitigation Strategies
        - Implement email filtering and spam detection
        - Train employees on recognizing and avoiding phishing attempts
- Situational Awareness

        - Definition
            - Understanding the current threat landscape and maintaining vigilance to detect and respond to security incidents
        - Purpose
            - Enhances the organization's ability to detect and address security threats proactively
        - Example
            - Encouraging employees to report unusual activities, such as unexpected email links or unfamiliar login prompts
        - Mitigation Strategies
            - Provide regular threat intelligence updates to employees
            - Encourage a security-first culture where employees are actively involved in threat detection
- Using the Social-Engineer Toolkit (SET) for Training
    - Social-Engineer Toolkit (SET)
        - An open-source tool that simulates social engineering attacks, such as phishing and credential harvesting, for realistic training scenarios
    - Credential Harvesting
        - Simulates phishing pages to teach employees about credential theft risks
    - Phishing Tactics
        - Trains employees on recognizing phishing attempts in a controlled environment
    - OPSEC and Privacy

            - Demonstrates the ease with which sensitive information can be compromised, emphasizing the need for privacy
        - Situational Awareness
            - Realistic attack scenarios build awareness, helping employees detect threats in real life
    - Summary
        - Awareness and training are essential components of security management focused on educating employees about potential threats
        - Key areas include
            - Operational Security (OPSEC)
                - Protecting sensitive information by managing access and exposure
            - Physical Security
                - Preventing unauthorized physical access through training and protective measures
            - Privacy
                - Ensuring sensitive data is securely handled and compliant with regulations
            - Social Engineering
                - Identifying manipulation tactics used by attackers to gain unauthorized access
            - Phishing
                - Recognizing fraudulent communications to prevent credential theft
            - Situational Awareness
                - Building vigilance to stay prepared for evolving threats

        - Through tools like the Social-Engineer Toolkit (SET), organizations can simulate real-world attack scenarios, improving employees' ability to recognize and avoid security threats
        - Regular security training and awareness initiatives strengthen the organization's overall security posture, equipping employees to handle a dynamic threat landscape

- **Governance Frameworks**
    - Governance Frameworks
        - Establish structured guidelines and best practices for managing and aligning IT operations with business goals
        - Ensure proper risk management and regulatory compliance
    - Governance Frameworks Covered
        - COBIT (Control Objectives for Information and Related Technologies)
            - Definition
                - IT governance and management framework focusing on aligning IT processes with business objectives
                - Emphasizes risk management and regulatory compliance
            - Domains
                - Divides IT into four domains
                    - Plan and Organize
                        - Includes processes such as defining a strategic plan
                    - Acquire and Implement
                        - Ensures systems security during acquisition
                    - Deliver and Support

            - Involves supporting and delivering IT services
          - Monitor and Evaluate
            - Continuous monitoring for effectiveness
    - Total Processes
      - 34 processes spread across four domains
    - Implementation Steps
      - Define strategic goals and key business objectives
      - Evaluate current IT practices to identify gaps
      - Implement COBIT's recommended processes to improve IT governance
      - Continuous monitoring using COBIT’s metrics for ongoing improvements
  - ITIL (Information Technology Infrastructure Library)
    - Definition
      - IT service management framework providing best practices for delivering IT services
      - Focuses on meeting business needs with consistent and reliable service
    - Adoption
      - Widely adopted by IT operations teams to improve service quality
      - Effective in Agile, DevOps, and DevSecOps environments
      - Standard in large enterprises, including Fortune 500 companies
    - Information Security Management

            - Included as one of ITIL's 34 practices to integrate security with IT services
    - Comparison of COBIT and ITIL
        - COBIT
            - Focus
                - Align IT processes with business objectives
            - Emphasis
                - IT governance, risk management, compliance
        - ITIL
            - Focus
                - Delivering and managing IT services to meet business needs
            - Emphasis
                - IT service management, integration with Agile, DevOps, and security teams
    - Key Points on Governance Frameworks
        - COBIT helps manage IT governance, aligning IT with business objectives, managing risks, and maintaining compliance
        - ITIL provides best practices for IT service management, improving service quality, aligning IT with business needs, and integrating security
        - Both frameworks are crucial for ensuring IT operations align with business goals

- **Governance, Risk, and Compliance (GRC) Tools**
    - Governance, Risk, and Compliance (GRC) Tools
        - Tools used to integrate governance, risk, and compliance management

        - Automate processes, track compliance, and maintain documentation
    - Features of GRC Tools
        - Documentation
            - Definition
                - Provides a centralized place to store and organize policies, procedures, and regulations
            - Example
                - RSA Archer
                    - Helps businesses store documentation in one place
        - Mapping
            - Definition
                - Links internal processes to external regulations and standards
                - Ensures alignment with external regulatory requirements
            - Example
                - SAP GRC
                    - Maps company activities to rules like the GDPR
        - Compliance Tracking
            - Definition
                - Monitors whether a company adheres to necessary regulations and policies
            - Example
                - ServiceNow GRC
                    - Tracks compliance and sends alerts when audits are due
        - Automation

                - Definition
                    - Automates repetitive tasks such as control testing and report generation
                - Example
                    - SAP GRC
                        - Automates control testing for efficient compliance.
            - Continuous Monitoring
                - Definition
                    - Monitors risks and compliance in real-time
                - Example
                    - Qualys
                        - Continuously monitors a network for security risks
    - Key Points on GRC Tools
        - Documentation
            - Organizes policies and procedures for easy access during audits
        - Mapping
            - Aligns internal processes with external regulations
        - Compliance Tracking
            - Monitors adherence to regulations and sends alerts for reviews
        - Automation
            - Reduces manual work by automating tasks like control testing
        - Continuous Monitoring
            - Provides real-time visibility into risks and compliance issues

- **Management Involvement**
    - Management Involvement

        - Effective governance relies on management involvement to assign and manage roles and responsibilities
        - Management's participation and commitment are key to setting business strategy and implementing governance structures
    - RACI Matrix
        - Definition
            - Tool for defining and clarifying roles in projects
            - Outlines roles as Responsible, Accountable, Consulted, and Informed
        - Structure
            - Typically presented as a table listing tasks (rows) and roles (columns)
            - Specifies role involvement at the intersection of each task and role
        - Purpose
            - Assign clear roles, prevent confusion or overlaps
            - Ensure efficient communication, accountability, and project management
    - RACI Matrix Categories
        - Responsible (R)
            - Person or team directly in charge of task completion
            - Involves executing plans, meeting deadlines, and progressing the task
        - Accountable (A)
            - Person with final authority over the task's outcome
            - Ensures the task meets goals and objectives; typically, only one person is designated

        - Consulted (C)
            - Stakeholders or experts providing opinions before decisions are made
            - Their role is advisory to guide the task towards quality standards
        - Informed (I)
            - Individuals who need progress updates but do not participate in task execution
            - Ensures transparency and alignment across the organization
    - Example of RACI in Action
        - Scenario
            - Company-wide software rollout
                - Responsible
                    - IT department – handles technical setup.
                - Accountable
                    - Chief Information Officer (CIO) – oversees the project success
                - Consulted
                    - HR department – ensures system meets employee needs
                - Informed
                    - Senior leadership – kept updated on progress
    - Key Points on Management Commitment
        - Active management involvement is crucial for successful governance and project execution
        - Management ensures proper role assignment, accountability, and alignment with business goals

        - Without management commitment, projects risk delays, resource issues, and inefficiencies

- **Change and Configuration Management**
    - Change and Configuration Management
        - Structured processes to manage system modifications efficiently, ensuring changes are approved and tracked accurately
        - Critical aspects include inventory, asset management lifecycle, and Configuration Management Database (CMDB)
    - Change and Configuration Management Elements
        - Inventory
            - Definition
                - Complete and accurate record of all hardware, software, and other assets within an organization
            - Importance
                - Helps assess the impact of proposed changes and prevents disruptions during updates or maintenance
                - Assists in troubleshooting by identifying and isolating issues
            - Example Tool
                - Microsoft System Center Configuration Manager (SCCM)
                    - Manages hardware and software assets, tracks configurations, and pushes updates
        - Asset Management Lifecycle
            - Definition

        - Process ensuring all assets are tracked, maintained, and updated throughout their lifespan
    - Five Steps
        - Planning and Procurement
            - Acquire assets that align with organizational needs
        - Deployment
            - Integrate assets into the environment
        - Utilization and Maintenance
            - Apply regular updates and patches
        - Asset Tracking
            - Document and manage changes
        - Decommissioning and Disposal
            - Retire outdated assets smoothly
    - Example Tool
        - ServiceNow IT Asset Management
            - Manages the asset lifecycle
- Configuration Management Database (CMDB)
    - Definition
        - Centralized database tracking Configuration Items (CIs) within an organization
        - Configuration Items
            - Assets, components, or services managed to deliver IT services (e.g., servers, software, databases)
        - Functions

            - Maintains information on each CI’s setup, usage, and relationships with other CIs
            - Tracks changes to ensure network stability and effective IT service management
        - Example Tool
            - ServiceNow CMDB
                - Tracks and manages CIs and their relationships
    - Key Points on Change and Configuration Management
        - Inventory
            - Provides an accurate record of assets, ensuring no critical systems are missed during updates
        - Asset Management Lifecycle
            - Manages assets from acquisition through disposal, focusing efforts on active assets
        - CMDB
            - Tracks configuration items and their relationships, preventing disruptions by understanding system interdependencies

- **The Data Life Cycle**
    - Data Lifecycle
        - Describes managing data through six stages
            - Creation
            - Use
            - Sharing
            - Storage

            - Archival
            - Destruction
        - Staging is an intermediate phase for development, testing, and quality assurance before production
- Data Lifecycle Stages
    - Creation
        - Data is acquired, entered, or captured (e.g., receiving an email or generating logs)
    - Use
        - Data is accessed, processed, or modified. Includes an audit trail (e.g., document edits)
    - Sharing
        - Data is made available to others (e.g., sharing financial reports via email)
    - Storage
        - Data is maintained for future use (e.g., saving financial records for trend analysis)
    - Archival
        - Data is moved to long-term storage for later recovery (e.g., archiving customer records)
    - Destruction
        - Data is securely destroyed when no longer valuable (e.g., deleting financial records after seven years)
- Development, Testing, Quality Assurance, and Production
    - Development

            - Writing code, designing system architecture, and configuring software
            - Lays the groundwork for system operations
        - Testing
            - Evaluates system functionality through various scenarios to identify bugs or errors
        - Quality Assurance (QA)
            - Reviews and validates the entire system to ensure standards are met before release
        - Production
            - The system is fully deployed and used in a live environment (e.g., e-commerce site launch)
    - Key Points on Data Lifecycle
        - The data lifecycle ensures efficient and secure management of data from creation to destruction
        - Staging activities (Development, Testing, QA) prepare and validate data before it is fully deployed in Production

- **Communication Considerations**
    - Communication Considerations
        - Involves effective communication and reporting within security program management
        - Ensures crucial information about security incidents, compliance status, and risk assessments is shared with internal and external stakeholders
    - Importance of Communication

        - Enables clear, timely sharing of information about risks, incidents, and responses throughout an organization
        - Supports coordinated actions during security events
    - Importance of Reporting
        - Documents and shares essential details about security incidents, risks, and compliance status
        - Helps decision-makers act swiftly, address potential threats, and maintain compliance
    - Key Points on Communication and Reporting
        - Communication
            - Ensures information is shared effectively, enabling coordinated responses
        - Reporting
            - Provides accurate documentation for decision-makers, ensuring timely action and regulatory compliance
        - Prioritizing both communication and reporting minimizes damage, maintains trust, and supports effective security management

Risk Management

Objective 1.2: Perform risk management activities

- **Confidentiality Risk Considerations**
  - Confidentiality Risk
    - The potential for unauthorized access, disclosure, or misuse of sensitive information
    - Considerations:
      - Incident Response Testing
      - Encryption
      - Sensitive Data Breach
      - Privileged Data Breach
      - Data Leak Response
      - Reporting
  - Incident Response Testing
    - Regular testing of incident response capabilities to ensure quick, effective handling of security incidents
    - Purpose:
      - Identifies weaknesses in response procedures and strengthens the ability to contain and mitigate information breaches
    - Example:
      - Riverbend swiftly implemented its incident response plan after detecting suspicious network activity
  - Encryption

        - The process of making data unreadable without a decryption key, protecting data both in transit and at rest
        - Consideration:
            - Attackers may compromise privileged accounts or access decryption keys, rendering encryption ineffective
        - Example:
            - Riverbend used encryption, but the attack escalated because the attacker gained access to the decryption key
    - Sensitive Data Breach
        - Unauthorized access to personal data, such as Personally Identifiable Information (PII) or driver's license details
        - Risk:
            - Exposed data may be misused for identity theft and fraud
        - Example:
            - Riverbend's patient contact information was compromised, constituting a sensitive data breach
    - Privileged Data Breach
        - Unauthorized access to highly sensitive data, such as financial records or health information
        - Risk:
            - Higher confidentiality risk due to the critical nature of the data
        - Mitigation:
            - Strict access controls, multi-factor authentication, and Data Loss Prevention (DLP) tools
        - Example:

            - Riverbend's privileged data breach allowed attackers access to encryption keys, heightening the impact of the breach
    - Data Leak Response
        - A plan designed to reduce the impact of a confidentiality breach
        - Components:
            - Quick identification of the source, containment of exposure, and corrective actions to prevent further data loss
        - Example:
            - Riverbend activated its data leak response plan immediately, preventing further delays in addressing the breach
    - Reporting
        - Ensures relevant internal and external stakeholders are informed of the breach in a timely manner
        - Importance:
            - Facilitates decision-making
            - Compliance with legal obligations
            - Coordination of response efforts
        - Example:
                - Riverbend's comprehensive reporting allowed for timely communication with regulatory bodies and customers

- **Integrity Risk Considerations**
    - Integrity
        - The assurance that data remains accurate and unaltered, with any unauthorized changes being detectable

        - Integrity risk considerations involve protective actions to prevent unauthorized modifications that could compromise data accuracy and trustworthiness
    - Interference
        - Definition
            - Any unauthorized modification or disruption of data or systems that compromise accuracy and integrity
        - Purpose
            - Prevents data inaccuracy due to malicious attacks, accidental changes, or system errors
        - Example
            - An attacker modifies financial records in a database, altering transaction accuracy
        - Mitigation Strategies
            - Implement strict access controls and user authentication
            - Enable system monitoring and logging to detect unauthorized changes
            - Use backup and failover strategies to restore systems if interference occurs
    - Hashing
        - Definition
            - A cryptographic technique that generates a fixed-size "fingerprint" or hash value from an input of any size to ensure data integrity
        - Purpose

        - Detects unauthorized data changes by comparing original and current hash values, as hash outputs cannot be reverse-engineered to discover the original input
    - Example
        - SHA-256 hashing of a document ensures any alterations produce a different hash, signaling an integrity breach
    - Mitigation Strategies
        - Use secure hashing algorithms like SHA-256 or SHA-3 to prevent hash collisions
        - Regularly compare stored hash values with newly generated hashes to verify data integrity
- Remote Journaling
    - Definition
        - The continuous transmission of transaction logs to a remote location to securely record system activities in real-time
    - Purpose
        - Maintains a secure, real-time record of critical system changes and transactions to reconstruct activities after a failure or breach
    - Example
        - Transaction logs transmitted offsite every few minutes to capture recent system activities in case of a local data center outage
    - Mitigation Strategies
        - Enable continuous or scheduled log transmission to a remote, secure location
        - Monitor log integrity and access to ensure they remain accurate and untampered

- Anti Tampering
  - Definition
    - Measures designed to detect or prevent unauthorized modifications to hardware or software
  - Purpose
    - Ensures that systems remain secure and trustworthy, with any tampering being detectable
  - Example
    - Tamper-evident seals on hardware to alert administrators if a physical breach occurs
  - Mitigation Strategies
    - Use checksums or tamper-evident logs to monitor for unauthorized software modifications
    - Implement physical protections like seals, locks, or alarms on hardware components
- Summary
  - Integrity risk considerations focus on safeguarding data and systems from unauthorized modifications that could compromise their accuracy and reliability
  - The components include
    - Interference
      - Unauthorized disruptions that compromise data accuracy, mitigated by access controls and monitoring
    - Hashing

            - Cryptographic technique producing unique digital fingerprints for data, enabling the detection of unauthorized changes
        - Remote Journaling
            - Offsite log storage to maintain transaction integrity and facilitate activity reconstruction after failures
        - Anti Tampering
            - Measures to prevent or detect modifications to hardware or software, ensuring systems remain unaltered and secure
    - By implementing these integrity controls, organizations can detect and prevent unauthorized changes, ensuring that their data remains accurate, reliable, and trustworthy

- **Availability Risk Considerations**
    - Availability Risk
        - Ensures critical systems and data remain accessible during and after disruptive events
        - Considerations:
            - Business Continuity and Disaster Recovery (BC/DR) Planning
            - Connected Backups
            - Disconnected Backups
            - BC/DR Testing
    - Business Continuity and Disaster Recovery (BC/DR) Planning
        - A process ensuring that critical assets and Mission Essential Functions (MEFs) are maintained during a disruption

- Business Impact Analysis (BIA):
  - Identifies and evaluates the effects of not maintaining Mission Essential Functions (MEFs)
  - *MEFs*
    - Core activities necessary to fulfill a business's strategic mission
  - Critical Assets:
    - Systems
    - Data
    - Processes supporting MEFs
  - Disaster Examples:
    - Natural disasters
    - Cyberattacks
    - Power outages
    - Pandemics
  - Backup Site Types:
    - Hot Site:
      - Fully operational, real-time backup with continuous mirroring of data
    - Warm Site:
      - Technical infrastructure in place, but data must be restored
    - Cold Site:
      - Basic infrastructure, requiring complete rebuild and setup
    - Cloud Site:
      - Uses cloud-based services for flexibility, combining elements of hot and warm sites
- Connected Backups

        - Backups connected to the production network, providing continuous data synchronization
        - Types:
            - Synchronous Backups
                - Real-time data mirroring; supports hot sites
            - Asynchronous Backups
                - Transmits data with a delay, supporting warm sites
            - Example:
                - Minimizes data loss by mirroring real-time transaction data across multiple data centers
    - Disconnected Backups
        - Backups stored offline or off-site, not connected to the production network, unaffected by network attacks
        - 3-2-1 Backup Rule:
            - Three Copies
                - Keep three copies of data
            - Two Media Types
                - Store data on two different storage media types
            - One Off-Site
                - Store one backup copy off-site for additional security
        - Example:
            - Provides assurance that data being restored is unaltered by cyberattacks
    - Business Continuity and Disaster Recovery (BC/DR) Testing
        - Regular simulations of real-world disruptions to verify the effectiveness of recovery strategies

        - Testing Includes:
            - Shifting operations to a backup data center or alternate office location
            - Identifying gaps in the recovery plan
                - Communication breakdowns
                - Equipment failures
            - Ensuring employees know their roles during a crisis
            - Example:
                - Minimizes downtime and ensures the availability of Mission Essential Functions during real disruptions

- **Privacy Risk Considerations**
    - Privacy Risk
        - Potential for unauthorized access, misuse, or disclosure of personal or sensitive information
        - Privacy Risk Considerations:
            - Biometric data
            - Data subject rights
            - Data sovereignty
    - Biometric Data Considerations
        - Biometrics:
            - Technologies such as fingerprint scans, facial recognition, iris scans, and voice recognition
        - Privacy Risks:
            - Due to the sensitive and permanent nature of biometric data
        - False Acceptance Rate (FAR):

            - Measures how often unauthorized individuals are mistakenly identified as legitimate users
        - False Rejection Rate (FRR):
            - Tracks how often legitimate users are wrongly denied access
        - Crossover Error Rate (CER):
            - The point where FAR and FRR are equal; system sensitivity should be set near this point for optimal security and user convenience
        - Adjusting Sensitivity:
            - Increased Sensitivity: Higher FRR, lower FAR
            - Lowered Sensitivity: Lower FRR, higher FAR
    - Data Subject Rights
        - Legal protections allowing individuals control over their personal data.
        - Relevant Laws:
            - *GDPR (General Data Protection Regulation)*
                - Allows individuals to access, correct, delete, and transfer their personal data
                - Requires organizations to inform individuals of how their data is used
            - *CCPA (California Consumer Privacy Act)*
                - Grants California residents the right to know what data is collected
                - Allows requests for data deletion and opting out of data sales
            - *NY SHIELD Act (Stop Hacks and Improve Electronic Data Security)*
                - Ensures organizations take steps to protect data and notify individuals of breaches

    - Empowerment of Data Subjects
        - Provides tools and rights to control how personal information is collected, used, and shared
- Data Sovereignty
    - Data is subject to the laws of the country where it is collected or stored
    - Implication:
        - Organizations must comply with the legal frameworks of both the country where data originates and where it is stored or processed
    - GDPR Example:
        - Personal data collected from EU citizens must comply with GDPR regulations, even if stored outside the EU
    - Complexity for Multinational Companies:
        - Ensures compliance with multiple legal standards across different jurisdictions
- Case Study: 2019 BioStar 2 Data Breach
    - Incident:
        - Exposed 27.8 million biometric records, including fingerprints and facial recognition data
    - Privacy Risk:
        - Long-term risks due to exposure of biometric data (a permanent identifier)
    - Regulatory Impact:
        - GDPR classifies biometric data as special category data, requiring stronger safeguards
        - Breach raised compliance questions about GDPR and data handling by Suprema

        - Data Sovereignty:
            - Data exposed across multiple countries, implicating various privacy laws

- **Risk Assessment Frameworks**
    - Risk Assessment Framework
        - A toolkit for identifying, assessing, and managing risks. Helps organizations address threats, ensure compliance, and protect assets and operations
    - Common Risk Assessment Frameworks:
        - *NIST RMF (National Institute of Standards and Technology Risk Management Framework)*
            - Designed for critical sectors: defense, healthcare, and federal agencies
            - Focuses on cybersecurity risk management and compliance with standards like FISMA
            - Provides a step-by-step process for selecting, implementing, and monitoring security controls
            - Key Feature:
                - Continuous monitoring and ongoing risk assessment to ensure proactive management
    - ISO/IEC 27005
        - Structured, standardized framework for information security risk management
        - Helps organizations identify, assess, and mitigate security risks throughout the information lifecycle

    - Works in conjunction with ISO/IEC 27001 for implementing Information Security Management Systems (ISMS)
    - Emphasizes documentation and evaluation of risks
- *COSO ERM (Committee of Sponsoring Organizations of the Treadway Commission Enterprise Risk Management)*
    - Aligns risk management with business strategy
    - Addresses risks across the entire organization, including financial, operational, and cybersecurity risks
    - Emphasizes strong governance and integrates risk management into corporate strategy and daily decisions
- *OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)*
    - Hands-on framework for assessing risks, threats, and vulnerabilities specific to organizational assets
    - Focuses on IT infrastructure and emphasizes identifying risks from an internal perspective
    - Helps create tailored risk management plans based on operational needs and business priorities
- *FAIR (Factor Analysis of Information Risk)*
    - Focuses on quantifying risk in financial terms
    - Helps decision-makers understand the monetary impact of security threats
    - Useful for non-technical stakeholders to view cyber risk as a financial issue
    - Supports clear decision-making for resource allocation in risk mitigation

- **Risk Assessment**
    - Risk Assessment
        - Risk assessment identifies, analyzes, and evaluates the potential impact of risks and guides the implementation of mitigation strategies.
        - Key Concepts:
            - Quantitative Analysis
            - Qualitative Analysis
            - Risk Prioritization
            - Risk Appetite
            - Risk Tolerance
    - Risk Management Lifecycle
        - Steps:
            - Identification of risks
            - Assessment of risks
            - Control of risks
            - Review of risks
    - Risk Management
        - Risk management helps identify risks and implements controls to bring risk levels to an acceptable threshold
        - Choices:
            - *Risk Acceptance*
                - Choosing to accept a risk due to high mitigation costs, while monitoring it for changes
            - *Risk Avoidance*
                - Identifying a risk and stopping the risky activity entirely to avoid it

            - *Risk Mitigation*
                - Implementing controls to reduce risk to an acceptable level, acknowledging residual risk
    - Residual Risk
        - The remaining risk after mitigation efforts have been applied
    - Risk Analysis
        - The process of determining how to handle identified risks, involving both quantitative and qualitative methods
    - Qualitative Risk Analysis
        - Assigns non-numeric values (e.g., low, medium, high) to risks based on intuition, experience, and best practices
        - Best Practices:
            - Brainstorming
            - Focus Groups
            - Surveys
            - Delphi Method (asking experts for consensus)
        - Example:
            - Using terms like "high risk" or "low risk" to describe a risk's potential impact
        - Downside:
            - Lacks numeric data, making cost/benefit analysis and budget forecasting difficult
    - Quantitative Risk Analysis
        - Uses numeric and monetary values to assess risk, including assets, threat frequency, vulnerability severity, and impact.
        - Approach:

            - Treats risk assessment as a mathematical problem, providing direct cost estimates for risks
    - Hybrid Approach
        - Combines both quantitative and qualitative methods, often used when not enough data is available for a purely quantitative analysis
    - *Risk Prioritization*
        - Ranking risks based on their likelihood and potential impact
        - Purpose:
            - Helps organizations allocate resources effectively, addressing high-priority risks first
    - *Risk Appetite*
        - The overall amount of risk an organization is comfortable taking on to achieve its goals
    - *Risk Tolerance*
        - The level of variation an organization is willing to accept before corrective action is necessary

- **Risk Response**
    - Risk Response
        - Risk response is the implementation of controls to mitigate identified risk
        - Key Concepts:
            - Validation
            - Severity Impact
            - Remediation
    - Risk Validation

    - Risk validation is the process of confirming whether an identified risk is real or a false alarm
    - Example:
        - A team receives an alert about a possible security breach. They investigate to confirm if the breach is a real threat or a benign event, such as an employee logging in from a different location
    - Purpose:
        - Ensures that resources are allocated to address real risks rather than false alarms
- Risk Severity Impact
    - Risk severity impact assesses the potential consequences or disruption a validated risk may cause
    - Example:
        - After validating a supplier delay, the company determines the severity of its impact on production. A small delay may have minor effects, while a major delay could cause significant revenue loss and missed deadlines
    - Purpose:
        - Helps prioritize risk response based on how critical the impact might be
- Risk Remediation
    - Risk remediation involves taking actionable steps to reduce or eliminate the impact of a confirmed risk
    - Example:
        - Following a phishing attack, the company blocks affected accounts, enforces password resets, installs multi-factor

authentication, and provides employee training to mitigate the risk

        - Purpose:
            - Protects the organization by minimizing or eliminating the impact of a risk

- **Impact Analysis**
    - Impact Analysis
        - The process of evaluating the potential consequences of identified risks on an organization's operations, assets, and objectives
        - Quantifies the effect of adverse events on business continuity, financial performance, and compliance to prioritize risk mitigation
    - Five Steps of Impact Analysis
        - Identify and Analyze Events
            - Purpose
                - Identify extreme scenarios with severe consequences that could realistically happen
            - Examples
                - Cyberattack
                    - A ransomware attack that locks down critical organizational data
                - Natural Disaster
                    - A massive hurricane that damages the company's main data center
        - Evaluate Impact
            - Purpose

            - Assess the key business assets or processes that would be impacted by the identified scenarios
        - Examples
            - Ransomware Attack
                - Impacts data, customer information, and operational systems, halting processes like billing, customer service, and order fulfillment
            - Natural Disaster
                - Damages essential hardware and backup systems, disrupting access to critical applications and data
    - Develop Scenarios
        - Purpose
            - Create realistic scenarios based on the identified threats and impacts
        - Examples
            - Ransomware Attack
                - Hackers infiltrate the system through phishing, encrypt critical data, forcing the company to decide whether to pay the ransom or lose access to vital data
            - Natural Disaster
                - A hurricane floods the data center, leading to widespread hardware failure and delays in restoring backups due to damaged infrastructure
    - Assess the Outcomes
        - Purpose

- Evaluate the ripple effects of each scenario across the organization
- Examples
  - Ransomware Attack
    - Results in operational downtime, financial losses, damaged customer trust, and potential regulatory fines for compliance failures
  - Natural Disaster
    - Leads to delayed operations, costly hardware replacements, missed deadlines, loss of customer confidence, and relocation expenses
- Implement Mitigation Strategies
  - Purpose
    - Recommend controls or strategies to reduce the damage from the scenarios
  - Examples
    - Ransomware Attack
      - Implement backup protocols, regular employee phishing training, and advanced threat detection software to restore systems without paying ransom or prevent the attack
    - Natural Disaster
      - Establish an offsite backup data center, use cloud-based data redundancy, and develop a disaster recovery plan for quick service restoration
- Summary

      - Impact Analysis
         - Helps organizations assess the consequences of major risks to operations, assets, and objectives by identifying extreme scenarios, evaluating impact, developing realistic scenarios, assessing outcomes, and implementing mitigation strategies
      - Examples
         - Ransomware Attack
            - Disrupts business operations, causing downtime, financial losses, and damaged customer trust
         - Natural Disaster
            - Causes delays, infrastructure damage, and recovery costs
      - Mitigation
         - Implementing backup systems, training, cloud redundancy, and recovery plans ensures resilience against extreme risks

- **Third-Party Risk Management**
   - Third-Party Risk Management
      - The process of assessing and mitigating risks associated with external parties that provide commodities or services to an organization
      - Involves risks such as Vendor Risk, Subprocessor Risk, and Supply Chain Risk
   - Key Components of Third-Party Risk Management
      - Vendor Risk
         - Definition
            - The risk posed by external vendors who provide goods or services directly to the organization

        - Examples
            - Software Vendor Breach
                - If a vendor with poor security suffers a breach, sensitive organizational data could be exposed
        - Management Techniques
            - Due Diligence
                - The process of evaluating a vendor's reliability, risks, and integrity before entering a partnership
            - Due Care
                - Ongoing efforts to mitigate risks by maintaining a secure relationship with the vendor
            - Product Support Lifecycle
                - Ensuring long-term updates and security patches from vendors to reduce vulnerabilities (e.g., Microsoft vs. less stable vendors)
    - Subprocessor Risk
        - Definition
            - The risk that arises when vendors outsource part of their services to third parties known as subprocessors
        - Examples
            - Cloud Provider Outsourcing
                - A cloud service provider outsourcing data management to another company
        - Management Techniques
            - Disclosure

            - Vendors must disclose their subprocessors and their roles
        - Verification
            - Ensure that subprocessors follow strong security practices and comply with expected standards
    - Supply Chain Risk
        - Definition
            - The risk associated with all stages of the supply chain, from parts sourcing to final product delivery
        - Examples
            - Hardware Tampering
                - A router with tampered parts could compromise network security
            - Source Authenticity
                - Purchasing equipment from unauthorized vendors could increase the risk of counterfeit or compromised devices
        - Management Techniques
            - Supply Chain Assessment
                - Evaluating the reliability of suppliers to ensure secure components
            - Source Authenticity Verification
                - Purchasing from trusted sources (e.g., Cisco) to avoid malware or hidden threats in products
- Summary
    - Third-Party Risk Management

    - Focuses on identifying and mitigating risks from external vendors, subprocessors, and the entire supply chain
  - Key Aspects
    - Vendor Risk
      - Involves due diligence, ensuring product support, and managing risks from direct suppliers
    - Subprocessor Risk
      - Requires vendors to disclose subprocessors and verify their security practices
    - Supply Chain Risk
      - Includes conducting supply chain assessments and ensuring source authenticity
  - Goal
    - To maintain security, integrity, and reliability throughout external business relationships

Compliance

Objective 1.3: Explain how compliance affects information security strategies

- **Industry Compliance**
    - Industry Compliance
        - Requires organizations to meet established laws, regulations, guidelines, and specifications
        - Aims to protect sensitive data, maintain customer trust, and avoid regulatory penalties
        - Specific industries with notable compliance requirements include Government, Healthcare, Financial, and Utilities
    - Government Compliance
        - Federal Information Security Management Act (FISMA)
            - Requires federal agencies to develop and implement security programs to protect their information
        - National Institute of Standards and Technology Risk Management Framework (NIST RMF)
            - Provides a method for managing risk, including categorizing information systems, selecting security controls, and monitoring their effectiveness
        - Cybersecurity Maturity Model Certification (CMMC)
            - Essential for government contractors to safeguard controlled unclassified information (CUI)
            - CUI refers to sensitive information that is not classified but requires protection

        - Divided into five levels of certification
            - Level 1
                - Basic cybersecurity hygiene (e.g., password protection, backups).
            - Level 2
                - Intermediate practices (e.g., access control, incident response)
            - Level 3
                - Full implementation of NIST SP 800-171 guidelines; advanced threat detection
            - Level 4
                - Proactive security (e.g., threat hunting, proactive risk management)
            - Level 5
                - Optimized defenses (e.g., adaptive security controls, supply chain security)
    - Healthcare Compliance
        - Health Insurance Portability and Accountability Act (HIPAA)
            - Focuses on protecting patients' sensitive health information and privacy
            - Sets strict rules on how medical data is stored, shared, and accessed
        - Health Information Technology for Economic and Clinical Health Act (HITECH)
            - Encourages healthcare providers to adopt electronic health records

        - Strengthens security and privacy measures for electronic data.
        - Enforces tougher penalties for improper safeguarding of patient information
- Financial Compliance
    - Gramm-Leach-Bliley Act (GLBA)
        - Ensures that banks and financial institutions protect customers' personal financial information
        - Requires third-party service providers to maintain adequate security measures
    - Sarbanes-Oxley Act (SOX)
        - Ensures public companies maintain accurate financial records and implement fraud prevention controls
    - Payment Card Industry Data Security Standard (PCI DSS)
        - Applies to businesses handling credit card transactions
        - Enforces six key security goals
            - Maintain a secure network
            - Encrypt cardholder data
            - Manage vulnerabilities
            - Enforce strict access controls
            - Continuously monitor systems
            - Have a robust information security policy
- Utility Compliance
    - North American Electric Reliability Corporation Critical Infrastructure Protection (NERC-CIP)
        - Focuses on securing systems that keep power grids operational

            - Ensures power companies have security practices to prevent disruptions
            - Non-compliance can result in fines of up to $1 million per day, per violation
        - Federal Energy Regulatory Commission (FERC)
            - Oversees and enforces compliance in the energy industry
            - Ensures utility companies conduct risk assessments and implement cybersecurity measures
    - Compliance Drivers by Industry
        - Government
            - Securing federal information through frameworks like FISMA, NIST RMF, and CMMC
        - Healthcare
            - Ensuring privacy and security of patient data through regulations like HIPAA and HITECH
        - Financial
            - Protecting financial information and preventing fraud through GLBA, SOX, and PCI DSS
        - Utilities
            - Securing critical infrastructure (e.g., power grids) through NERC-CIP and FERC oversight

- **Industry Standards**
    - Industry Standards
        - Established guidelines and practices that organizations within a specific industry are expected to follow.

        - These are often developed by industry bodies or regulatory organizations and serve as benchmarks for compliance and best practices
    - Payment Card Industry Data Security Standard (PCI DSS)
        - Applies to
            - Any business handling credit card transactions
        - Primary Goal
            - To protect credit card data and ensure transaction security
        - Six Key Security Goals
            - Build and maintain a secure network
            - Protect cardholder data through encryption
            - Maintain a vulnerability management program
            - Enforce strong access control measures
            - Regularly monitor and test networks
            - Establish an information security policy
        - Merchant Levels
            - Level 1
                - Processes over 6 million transactions/year
                - Requires an annual on-site audit by a Qualified Security Assessor and an annual Report on Compliance (ROC)
            - Level 2
                - Processes 1-6 million transactions/year
            - Level 3
                - Processes between 20,000 and 1 million transactions/year
            - Level 4
                - Processes fewer than 20,000 e-commerce transactions or fewer than 1 million overall transactions/year

        - Compliance Requirements for Levels 2, 3, and 4
            - Typically complete a Self-Assessment Questionnaire (SAQ)
            - May require quarterly network scans by an Approved Scanning Vendor
    - ISO/IEC 27000 Series
        - Purpose
            - A series of international standards for managing information security across industries
        - Core Components
            - ISO 27001
                - Focuses on establishing, implementing, maintaining, and improving an Information Security Management System (ISMS)
            - ISO 27002
                - Provides best practices for implementing security controls defined in ISO 27001
            - ISO 27005
                - Addresses information security risk management
            - ISO 27017
                - Focuses on security in cloud services
            - ISO 27018
                - Deals with protecting personal data in the cloud
        - Adoption
            - Not legally enforced but widely adopted to demonstrate data protection strategies to clients and regulators
    - Digital Markets Act (DMA)

        - Introduction
            - A European Union regulation to ensure fair competition in the digital market
        - Targets
            - Large tech companies known as "gatekeepers" that control key digital services (e.g., Google, Apple, Meta)
        - Key Requirements for Gatekeepers
            - Ensure interoperability for messaging apps from smaller providers with major platforms
            - Prevent favoring of their own products in search results or app stores
        - Enforcement
            - Managed by the European Commission
        - Penalties for Non-Compliance
            - Includes fines up to 10% of a company's global revenue or even break-up orders

- **Security Frameworks**
    - Security Frameworks
        - Sets of guidelines, best practices, and standards designed to help organizations manage and reduce cybersecurity risks
    - Foundational Best Practices
        - Purpose
            - Core principles that guide the implementation of security measures across industries
        - Examples of Foundational Best Practices

            - ISO 27000 Series
                - Sets international standards for managing information security
            - NIST Cybersecurity Framework
                - Helps businesses assess and improve their ability to protect information systems
            - COBIT
                - Essential for governing and managing IT systems
            - PCI DSS
                - Focuses on securing credit card transactions
            - COSO Framework
                - Emphasizes enterprise risk management
            - GDPR
                - Sets strict data privacy rules in the EU and for EU citizens
            - ITIL
                - Crucial for managing IT services
            - CMMC
                - Outlines security requirements for defense contractors
            - HIPAA
                - Ensures protection of patient data in healthcare
        - Goal
            - Implementing controls, reducing security risks, and ensuring compliance with relevant regulations
    - Benchmarks
        - Baseline

            - The minimum set of security configurations or standards set internally by an organization
        - Benchmark
            - An external standard that provides a reference point for evaluating how well internal baselines align with industry best practices
        - Types of Baselines
            - Configuration Baseline
                - Defines internal security configurations and provides a starting point for measuring security
            - Operational Baseline
                - Establishes normal operational traffic patterns within the organization
        - Continuous Monitoring
            - Purpose
                - Identifies abnormal events in organizational security
            - Example
                - Detecting a new user account logging in outside of normal hours, which could indicate potential foul play
        - Usage
            - Helps organizations identify and adjust abnormal patterns, ensuring ongoing security and alignment with industry standards
    - Center for Internet Security (CIS) Benchmarks
        - Purpose
            - Provides globally recognized, best-practice security configurations to secure systems and data against cyber threats

        - Application
            - Once an internal baseline is established, CIS benchmarks are used to compare it against industry best practices
        - Example
            - The CIS benchmark for Windows Server 2022 provides step-by-step security configuration guidelines
        - Goal
            - Aligns organizational security configurations with proven, widely tested setups to strengthen defenses and support compliance

- **Security Organization Control Type 2 (SOC 2)**
    - SOC 2 Framework
        - An auditing process that evaluates an organization's information security controls at a specific time or over a 6-to-12-month period
        - Trust Service Principles
            - Security
                - Ensures the system is protected from unauthorized access
            - Availability
                - Confirms that the system is reliable and accessible when needed
            - Processing Integrity
                - Verifies that data is processed accurately and completely
            - Confidentiality
                - Protects sensitive information from unauthorized access
            - Privacy

                ○ Covers how personal data is collected, stored, and managed

- Types of SOC 2 Reports
    - SOC 2 Type I
        - Focus
            - Assesses the design of an organization's controls at a specific point in time
        - Example
            - Evaluates whether a company has encryption and access control systems in place to protect data
        - Scope
            - Examines if security measures meet the trust service criteria but does not assess ongoing effectiveness
    - SOC 2 Type II
        - Focus
            - Evaluates both the design and operational effectiveness of controls over a period (typically 6 to 12 months)
        - Example
            - Checks if the encryption and monitoring systems functioned correctly and consistently protected data during the evaluation period
        - Scope
            - Verifies that the security controls were consistently effective over time
- SOC 3 Report
    - Purpose

            - A simplified, public version of the SOC 2 report
        - Audience
            - Intended for clients, customers, or the public who need reassurance of data protection practices
        - Content
            - Provides a high-level overview of controls related to security, availability, processing integrity, confidentiality, and privacy
        - Usage
            - Demonstrates a company's commitment to data security in an accessible format without technical details

- **NIST Cybersecurity Framework (CSF)**
    - NIST Cybersecurity Framework (CSF)
        - Developed by the National Institute of Standards and Technology (NIST) to help organizations manage and reduce cybersecurity risk
        - Designed to be flexible and adaptable across various industries
    - Five Core Functions of the NIST CSF
        - Identify
            - Purpose
                - Understand and manage risks to critical assets, systems, and data
            - Tools
                - Asset Management
                    - SolarWinds, Lansweeper
                - Risk Assessment
                    - RSA Archer, RiskLens

- Example
  - Mapping out critical systems and identifying vulnerabilities using asset management and risk assessment platforms
- Protect
  - Purpose
    - Implement protective measures to prevent unauthorized access and safeguard data
  - Tools
    - Firewalls
      - Cisco Adaptive Security Appliance, Palo Alto
    - Encryption
      - BitLocker, VeraCrypt
    - Identity and Access Management (IAM)
      - Okta
  - Example
    - Securing sensitive data with firewalls, encryption, and access controls
- Detect
  - Purpose
    - Monitor networks for potential threats and suspicious activities
  - Tools
    - Intrusion Detection Systems (IDS)
      - Snort
    - Security Information and Event Management (SIEM)
      - Splunk

            - Example
                - Using SIEM tools to detect signs of breaches by continuously monitoring network traffic
        - Respond
            - Purpose
                - Take action to contain, investigate, and respond to cybersecurity incidents
            - Tools
                - Incident Response Platforms
                    - IBM Resilient, CrowdStrike
            - Example
                - Responding to an incident by isolating affected systems and notifying personnel to investigate and mitigate the situation
        - Recover
            - Purpose
                - Restore data and services after a security incident while improving resilience for future incidents
            - Tools
                - Backup and Recovery
                    - Veeam, Acronis
            - Example
                - Using backup tools to restore lost data and analyzing the incident to strengthen defenses
    - Summary
        - NIST CSF

        - Helps organizations manage and reduce cybersecurity risks by organizing efforts into five core functions
            - Identify, Protect, Detect, Respond, and Recover
    - Functions
        - Work together to assess risks, implement protective measures, detect threats, respond to incidents, and recover from damage
    - Benefits
        - By following the NIST CSF, organizations can improve their overall cybersecurity posture and better safeguard their critical assets

- **Cloud Security Alliance (CSA)**
    - Cloud Security Alliance (CSA)
        - Set of guidelines and best practices for securing cloud computing environments
        - Provides a comprehensive approach to managing and mitigating risks in cloud services
        - Addresses areas such as:
            - Data protection
            - Security management
            - Compliance
    - CSA Security, Trust, Assurance, and Risk (CSA STAR) Program
        - Certification process for assessing and validating cloud service providers' security practices
        - Ensures transparency, trust, and compliance within cloud environments
        - Two certification levels:
            - Level 1: Self-assessment

            - Cloud providers voluntarily submit their own security and privacy assessments
            - Evaluates against the Cloud Controls Matrix or GDPR Code of Conduct

        - Level 2: Third-party audit
            - Conducted by an independent organization
            - Verifies compliance with security standards for added customer confidence
    - CSA STAR Program Details
        - Includes a publicly accessible registry for cloud providers' security and privacy controls
        - Standardizes security measures through frameworks like the Cloud Controls Matrix
        - Supports organizations in comparing cloud providers based on verified security practices
    - Example Providers in CSA STAR
        - Major cloud providers such as AWS and Google Cloud participate
        - Providers like Salesforce and Dropbox use CSA STAR to demonstrate security compliance

- **Privacy Regulations**
    - Privacy Regulations
        - Legal requirements designed to protect individuals' personal information by setting standards for how organizations collect, store, use, and share data

        - These regulations vary by region and jurisdiction
    - Children’s Online Privacy Protection Act (COPPA)
        - Applies to
            - Websites and online services directed toward children under 13, or any service that knowingly collects personal data from children under 13 in the U.S.
        - Requirements
            - Obtain parental consent before collecting data from children under 13
            - Protect the privacy and security of children's personal information
        - Enforcement
            - Managed by the Federal Trade Commission (FTC)
        - Fines
            - Up to $40,000 per violation
        - Example
            - Even if children lie about their age, websites that collect data from them are still subject to COPPA requirements
    - Brazil’s General Data Protection Law (LGPD)
        - Applies to
            - Any company processing personal data from Brazilian residents, regardless of the company’s location
        - Requirements
            - Obtain clear consent before collecting personal information
            - Allow users to access, correct, or delete their data
        - Fines

        - Up to 2% of company revenue in Brazil, capped at 50 million reais per violation
    - Example
        - A non-Brazilian company that processes Brazilian customer data must comply with LGPD, including obtaining clear consent
- California Consumer Privacy Act (CCPA)
    - Applies to
        - Businesses meeting criteria such as earning over $25 million in revenue, or those that handle personal data of 50,000 or more California residents
    - Requirements
        - Inform consumers about what personal data is being collected
        - Allow consumers to request deletion of their data
        - Provide an option to opt out of data sales to third parties
    - Fines
        - Up to $7,500 per intentional violation
    - Example
        - An online retail site collecting data from California residents must allow users to delete their data and opt out of data sales
- General Data Protection Regulation (GDPR)
    - Applies to
        - Organizations processing personal data of EU citizens
    - Requirements
        - Obtain informed consent for data collection and processing
        - Provide the "right to be forgotten," allowing users to request the deletion of their data

            - Allow users to inspect, amend, or erase any data held about them
        - Enforcement
            - Managed by Data Protection Authorities in each EU member state
        - Example
            - A company selling products in the EU must clearly explain how personal data will be used, and users must provide consent for any marketing material
    - Summary
        - COPPA
            - Protects children's data by requiring strict guidelines for websites and services targeting or collecting data from users under 13 in the U.S.
        - LGPD
            - Sets strict data protection standards for Brazilian residents, applying to companies worldwide that process Brazilian data
        - CCPA
            - Provides California residents control over their personal data, allowing them to request its deletion and opt out of sales, applying to large businesses and those handling significant consumer data
        - GDPR
            - Provides strong data privacy protections for EU citizens, requiring informed consent, data access, and deletion rights, with strict penalties for non-compliance

- **Security Reviews**
  - Security Reviews
    - Evaluations of an organization's security policies, controls, and practices to ensure they are effective, compliant with regulations, and aligned with industry best practices
  - Internal Audits
    - Purpose
      - Evaluate adherence to internal security policies and procedures
    - Example
      - A company checks if employee access to sensitive data is restricted and that patches for vulnerabilities are applied promptly
    - Focus
      - Internal processes, risk identification, and prevention of security breaches
    - Sharing of Results
      - Not normally required, though they can be shared internally
  - External Audits
    - Purpose
      - Assess compliance with external regulations, standards, or contractual obligations
    - Example
      - A financial services company undergoes a PCI DSS audit to prove it complies with credit card data handling standards
    - Focus
      - Compliance with regulations
    - Sharing of Results

      - Typically shared with regulators, clients, or stakeholders
  - Internal Assessments
    - Purpose
      - Broader evaluations aimed at understanding the organization's overall cybersecurity posture
    - Example
      - Conducting an internal assessment before migrating systems to the cloud to identify potential risks like insecure APIs
    - Focus
      - Risk evaluation and cybersecurity readiness
    - Goal
      - Take corrective action to address identified vulnerabilities
  - External Assessments
    - Purpose
      - Impartial evaluations conducted by third-party experts or consultants to review security from an outside perspective
    - Example
      - A healthcare company hires an external firm to conduct a penetration test to ensure HIPAA compliance
    - Focus
      - Identifying unseen weaknesses and ensuring compliance with external standards (e.g., HIPAA)
    - Results
      - Provide recommendations for security improvements
  - Certifications
    - Purpose

        - Formal recognition that an organization meets a specific set of cybersecurity standards
    - Example
        - Cloud service providers (e.g., AWS, Microsoft Azure) holding ISO 27001 certifications for security compliance
    - Process
        - Certifications are typically granted after successful audits or assessments by an accredited certifying body
    - Use Case
        - Demonstrating compliance with regulations in industries like healthcare (HIPAA) and finance (PCI DSS)
- Summary
    - Internal Audits
        - Conducted by the organization's own teams to evaluate adherence to internal policies
    - External Audits
        - Conducted by third parties to ensure compliance with external regulations like PCI DSS
    - Internal Assessments
        - Broader evaluations initiated by the organization to assess overall cybersecurity readiness
    - External Assessments
        - Objective reviews conducted by third-party experts to identify vulnerabilities and ensure compliance
    - Certifications

            - Formal recognition of compliance with industry standards like ISO 27001, which is crucial for industries requiring stringent data protection, such as healthcare and finance

- **Cross-Jurisdictional Compliance**
    - Cross-Jurisdictional Compliance
        - The process of ensuring an organization's practices comply with local, national, and international laws, covering areas such as due diligence, due care, contractual obligations, and export controls
    - Due Diligence
        - Definition
            - The preparation and investigation a company performs before taking an action
        - Example
            - Before adopting a new software solution, a company researches potential risks and implements necessary security measures
        - Purpose
            - To identify and mitigate risks ahead of time
    - Due Care
        - Definition
            - The ongoing actions a company takes to maintain security after identifying risks through due diligence
        - Example
            - Regularly monitoring systems, applying patches, and updating security measures to protect data
        - Purpose

            - To ensure continued protection against risks
    - Contractual Obligations
        - Definition
            - Specific legal requirements included in contracts, often involving compliance with different regional laws
        - Example
            - A company in the U.S. handling data for European clients must comply with both U.S. laws and the European GDPR
        - Purpose
            - To ensure legal compliance across multiple regions, especially when dealing with international clients or operations
    - Legal Hold and E-discovery
        - Legal Hold
            - Definition
                - The requirement to preserve data that may be relevant in legal proceedings
            - Purpose
                - To prevent the deletion or alteration of information that could be used as evidence in court
        - E-discovery
            - Definition
                - The process of searching through preserved data to find relevant information for a court case
            - Example
                - Legal teams and IT professionals work together to search through emails, documents, and files to gather evidence

- Export Controls
  - Definition
    - Regulations governing the international transfer of technologies, especially dual-use items with both civilian and military applications
  - Example
    - Exporting Checkpoint Firewall and VPN software requires compliance with the Wassenaar Arrangement, and may involve applying for export licenses
  - Purpose
    - To ensure that dual-use technologies are not exported without proper authorization, avoiding legal and financial penalties
- Summary
  - Due Diligence
    - Involves research and risk identification before implementing new technologies or services
  - Due Care
    - Ensures ongoing security maintenance and protection after risks are identified
  - Contractual Obligations
    - Require companies to comply with different regional laws, such as GDPR in Europe or local privacy regulations
  - Legal Hold and E-discovery
    - Legal hold preserves data for potential legal use, and e-discovery searches through this data to find relevant information for court cases

- Export Controls
    - Regulate the transfer of dual-use technologies, requiring compliance with agreements like the Wassenaar Arrangement to avoid penalties

Resilient System Design

Objective 2.1: Analyze requirements to design resilient systems

- **Security Devices**
  - Security Devices
    - Hardware or software tools designed to protect networks and data by enforcing policies and monitoring for malicious activity
  - Firewalls
    - Purpose
      - Manage and filter traffic based on rules defined in Access Control Lists (ACLs)
    - Types
      - Packet Filtering Firewalls
        - Inspect packet headers to allow or deny traffic based on IP addresses and port numbers
        - Limited against advanced attacks
      - Stateful Firewalls
        - Track active connections, allowing return traffic for outgoing requests but blocking unsolicited traffic
      - Proxy Firewalls
        - Act as intermediaries, inspecting traffic at various layers (e.g., session layer for circuit-level proxies, application layer for application-level proxies)
      - Next Generation Firewalls (NGFWs)

            - Application-aware firewalls capable of deep packet inspection and integrating with intrusion prevention systems
        - Web Application Firewalls (WAFs)
            - Protect web applications by filtering and monitoring HTTP traffic to prevent attacks like SQL injection and cross-site scripting
- Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)
    - Intrusion Detection Systems (IDS)
        - Purpose
            - Detect, log, and alert on malicious activity without taking action
        - Types
            - Network IDS (NIDS)
                - Monitors network traffic for threats such as port scans and malicious payloads
            - Host-based IDS (HIDS)
                - Detects suspicious activity on individual servers or endpoints
            - Wireless IDS (WIDS)
                - Monitors wireless networks for threats like denial-of-service attacks
        - Detection Methods
            - Signature-based
                - Identifies known attack patterns
            - Anomaly-based

            - Detects abnormal behavior, identifying new threats but may result in false positives
    - Intrusion Prevention Systems (IPS)
        - Purpose
            - Detect, log, alert, and block threats
        - Types
            - Network IPS (NIPS)
                - Positioned in line with traffic to stop real-time attacks
            - Host-based IPS (HIPS)
                - Responds to threats on individual systems
            - Wireless IPS (WIPS)
                - Prevents wireless attacks like unauthorized access attempts
- Virtual Private Network (VPN)
    - Purpose
        - Create encrypted tunnels over untrusted networks (e.g., the internet) to securely connect to enterprise resources
    - Types
        - Remote Access VPN
            - Enables secure connections for remote workers using encrypted tunnels
        - Site-to-Site VPN
            - Connects two office locations securely over the internet using encryption keys
- Network Access Control (NAC)

- Purpose
  - Ensure that only authorized and compliant devices can access the network, scanning devices for security compliance (e.g., updated antivirus, security patches)
- Types
  - Persistent Agents
    - Installed on devices, ideal for corporate environments
  - Non-persistent Agents
    - Temporarily installed for scanning and deleted afterward, common in environments like college campuses
  - Agentless NAC
    - Scans devices without installing software, ideal for bring-your-own-device (BYOD) environments

- Summary
  - Firewalls
    - Enforce rules using ACLs to filter traffic
    - Types include packet filtering, stateful, proxy, next-generation firewalls, and WAFs, each with different levels of inspection and control
  - IDS vs. IPS
    - IDS detects and alerts on malicious activity, while IPS takes active steps to block threats
  - VPNs
    - Provide secure, encrypted communication over untrusted networks, with remote access and site-to-site setups
  - NAC

        - Controls network access by verifying that devices meet security requirements, with different types of agents for various environments

- **Monitoring and Detection**
    - Monitoring and Detection
        - Continuous observation of network activities to identify and respond to security incidents, anomalies, and vulnerabilities
    - Test Access Points (TAPs)
        - Definition
            - Hardware that provides full visibility into network traffic by creating a copy of the data flowing between two points in the network
        - Purpose
            - Allows real-time traffic analysis and monitoring without disrupting data flow, ideal for deploying network-based IDS
        - Types
            - Passive TAPs
                - Split signals without altering traffic, require no power, but can't capture data if a connection fails
            - Active TAPs
                - Regenerate signals, require power, and can continue capturing data during link issues if a backup power supply is available
        - Example Use

            - For permanent installations needing real-time monitoring, such as a network-based IDS
        - Alternative for Virtual Environments
            - Virtual Private Cloud solutions, as physical TAPs are not compatible with virtualized settings
    - Collectors
        - Definition
            - Devices or software that gather network traffic or log data from various sources for analysis
        - Purpose
            - Collects data from servers, endpoints, firewalls, etc., to detect and alert on suspicious activity
        - Examples
            - Wireshark
                - Captures and analyzes packets in real-time for network protocol analysis
            - Splunk
                - Collects log data from multiple sources, providing insights and allowing detection of anomalies
            - SolarWinds Network Performance Monitor
                - Monitors network traffic and alerts on unusual occurrences like bandwidth spikes
    - Vulnerability Scanners
        - Definition
            - Tools that identify security weaknesses within a network or system, providing detailed reports for remediation

        - Purpose
            - Finds vulnerabilities like outdated software and misconfigurations before they can be exploited
        - Examples
            - OpenVAS
                - Open-source network vulnerability scanner, initially a fork of Nessus, suitable for various systems
            - Nessus
                - Commercially popular for ease of use and comprehensive plugin library, scanning for a range of vulnerabilities with actionable recommendations
            - Qualys
                - Cloud-based scanner known for scalability, also offering compliance checking and continuous monitoring
    - Summary
        - Monitoring and Detection are critical for identifying security incidents and vulnerabilities within a network
        - Key tools include
            - TAPs
                - Hardware that provides network visibility for real-time monitoring without interrupting data flow
            - Collectors
                - Software and devices that gather data from multiple sources, enabling security teams to detect and analyze threats
            - Vulnerability Scanners

                - Identify and report on potential weaknesses in systems, offering guidance for risk mitigation
        - These tools work together to ensure that potential threats are detected early, enabling quick responses to protect network security

- **Network Traffic Management**
    - Network Traffic Management
        - The control, direction, and optimization of data flow across a network, involving tools like proxies and content delivery networks (CDNs)
    - Forward Proxies
        - Purpose
            - Serve as intermediaries between users and external servers, managing outgoing traffic from internal clients
        - Benefits
            - Web Caching
                - Stores copies of frequently accessed static websites to speed up load times and reduce bandwidth use
            - Content Filtering
                - Blocks access to harmful or restricted websites, enhancing network security
            - Auditing
                - Logs user activity, allowing administrators to track internet usage and access attempts
        - Example
            - A forward proxy can block employees from accessing inappropriate content and monitor their web usage

- Reverse Proxies
  - Purpose
    - Serve as intermediaries between external users and internal servers, managing incoming traffic to internal applications
  - Benefits
    - Content Caching
      - Stores cached copies of web content, reducing server load and enhancing the user experience
    - Traffic Scrubbing
      - Inspects and filters incoming traffic, helping prevent DDoS attacks and improving security
    - IP Masking
      - Hides the internal server's IP address from clients, enhancing privacy and security
    - Load Balancing
      - Distributes traffic across multiple backend servers, ensuring better performance and reliability
  - Example
    - A reverse proxy distributes traffic to multiple web servers to prevent overload and enhance availability
- Content Delivery Networks (CDNs)
  - Purpose
    - Distribute copies of content to servers in various geographic locations to ensure fast and efficient content delivery
  - Benefits
    - Reduced Latency

            - Delivers content from the server closest to the user, reducing load times
        - Global Scalability
            - Handles large volumes of traffic across multiple regions efficiently
        - Content Acceleration
            - Routes traffic through the most efficient paths for faster delivery of dynamic and static content
        - DDoS Protection
            - Provides built-in protection against DDoS attacks, ensuring continuous service availability
    - Example
        - A CDN delivers website content to users from servers closest to their location, reducing page load times
- Types of Proxies
    - Forward Proxies
        - Position
            - Located at the edge of the network, managing outbound traffic from internal users
        - Functions
            - Caching Static Content
                - Speeds up website load times for frequently accessed pages
            - Filtering Access
                - Blocks access to restricted websites, enhancing compliance with company policies

            - Monitoring User Activity
                - Tracks internet usage to ensure productivity and prevent misuse.
    - Reverse Proxies
        - Position
            - Located at the edge of the network, managing inbound traffic to internal servers
        - Functions
            - Caching Content
                - Speeds up content delivery by storing frequently accessed data
            - Filtering Malicious Traffic
                - Protects servers from attacks like DDoS by filtering incoming traffic
            - Balancing Load
                - Distributes incoming requests across multiple servers to prevent overload and improve performance
- Summary
    - Forward Proxies
        - Focus on managing outbound traffic by caching content, filtering access, and monitoring user activity
    - Reverse Proxies
        - Focus on managing inbound traffic by caching content, scrubbing traffic for security, masking server IPs, and balancing load across multiple servers

            - CDNs
                - Ensure efficient and fast content delivery by distributing content to servers across various geographic locations, reducing latency, and optimizing traffic routing

- **Application Layer Security**
    - Application Layer Security
        - Involves protecting applications and their data from unauthorized access and attacks
        - Operates at the highest layer of the OSI model and includes tools like API gateways and Web Application Firewalls (WAFs)
    - API Gateways
        - Purpose
            - Controls the flow of data between clients and backend services, ensuring requests are valid and secure
        - Functions
            - Reverse Proxy
                - Acts as an intermediary, accepting API calls and routing them to the appropriate backend services
            - Request Routing
                - Directs specific requests (e.g., account details or payment processing) to the correct service, ensuring efficient handling
            - Authentication and Authorization
                - Verifies user credentials to control access to certain services or data

                - Rate Limiting and Throttling
                    - Manages the flow of requests to prevent overloading the system with too many requests from one device or user
                - Security
                    - Blocks unauthorized access and filters data flow to protect backend services from malicious activity
            - Example
                - Netflix uses an API gateway to route requests based on user location and device, ensuring efficient delivery of content
        - Web Application Firewalls (WAFs)
            - Purpose
                - Protects web applications from common attacks by inspecting and filtering HTTP traffic
            - Functions
                - Blocking Attacks
                    - Detects and blocks harmful traffic, such as SQL injections or cross-site scripting, before it reaches the web server
                - Deployment Options
                    - Separate Appliance
                        - Functions independently within the network infrastructure to inspect traffic
                    - Software Plugin
                        - Integrates directly with the web server to provide protection
                - Inline Deployment

                - Actively blocks threats in real-time but may slightly slow traffic or occasionally block legitimate requests
            - Out of Band Deployment
                - Passively monitors traffic by analyzing a copy, detecting threats without blocking live traffic
        - Example
            - A WAF detects and blocks a SQL injection attempt that could compromise a web application's database
    - Summary
        - API Gateways
            - Act as a traffic controller, ensuring secure, authorized requests reach backend services, while managing load, routing traffic, and handling security tasks like authentication and rate limiting
        - Web Application Firewalls (WAFs)
            - Filter and block harmful HTTP traffic to protect web applications from attacks like cross-site scripting and SQL injections, with the ability to operate inline or out of band

- **Availability Considerations**
    - Availability Considerations
        - Techniques and strategies used to ensure systems, applications, and services remain accessible and functional during failures, high demand, or disruptions
    - Load Balancing
        - Purpose

- Distributes incoming traffic across multiple servers to prevent any single server from becoming overwhelmed and to maintain service availability

- Techniques
  - Round-Robin
    - Requests are sent to servers in a rotating order, without considering server load
  - Least Connections
    - Traffic is directed to the server with the fewest active connections, balancing loads based on current server activity
  - IP Hashing
    - Maps a client's IP address to a specific server to ensure the same user is always sent to the same server
- DDoS Defense
  - Load balancers can help mitigate Distributed Denial of Service (DDoS) attacks by spreading traffic across servers and dropping excessive requests

- Persistence vs. Nonpersistence
  - Persistence (Session Affinity)
    - Definition
      - Ensures that once a user connects to a specific server, all subsequent requests from that user during the same session are handled by the same server
    - Use Case

            - Online shopping, where maintaining a consistent connection ensures that the user's shopping cart data remains accurate throughout their session
    - Nonpersistence
        - Definition
            - Each request a user makes can be handled by a different server, improving efficiency when session continuity isn't necessary
        - Use Case
            - Browsing a news website, where each article request is independent and doesn't rely on previous interactions
    - Purpose
        - Persistence is essential for maintaining session-specific data, while nonpersistence maximizes system efficiency by distributing requests across servers
- Interoperability
    - Definition
        - The ability of different systems, applications, or devices to work together, exchange, and use information seamlessly
    - Purpose
        - Ensures that various systems can function together, improving availability and minimizing downtime by allowing systems to share resources and handle requests efficiently
    - Use Case

            - In the financial industry, interoperability enables seamless transactions across different banking systems, regardless of the institution
    - Summary
        - Load Balancing
            - Distributes traffic across multiple servers, using algorithms like round-robin, least connections, and IP hashing to prevent overload and maintain availability
            - It also helps protect against DDoS attacks
        - Persistence vs. Nonpersistence
            - Persistence keeps users connected to the same server to maintain session continuity, while nonpersistence distributes requests across servers for efficiency when session data isn't critical
        - Interoperability
            - Ensures that different systems and applications can work together smoothly, sharing resources and improving system availability by reducing bottlenecks and downtime

- **Scaling Considerations**
    - Scalability
        - The ability of a system or application to handle increasing demand by effectively managing and allocating resources
        - Scalability is measured by the number of requests a system can support simultaneously
    - Vertical Scaling (Scaling Up)
        - Definition

        - Adding more resources (e.g., memory, processing power, storage) to an existing machine to improve its capacity
    - Example
        - Upgrading a cloud-hosted blog from a $5/month plan with 1 GB of memory to a $20/month plan with 4 GB of memory and more disk space and processing power
    - Characteristics
        - Popular in traditional and cloud systems
        - Limited by the maximum capacity of the machine or cloud plan
        - Once the system reaches its maximum vertical scale, horizontal scaling may be required
- Horizontal Scaling (Scaling Out)
    - Definition
        - Adding more machines (servers) to handle increasing workloads by distributing tasks across multiple machines
    - Example
        - A blog storing thousands of articles could distribute the articles across multiple servers, with each server responsible for a different year of articles
    - Characteristics
        - Enables long-term scalability by adding multiple inexpensive machines
        - Provides elasticity, allowing for virtually limitless scaling
        - Reduces costs over time compared to vertical scaling, as machines are cheaper than upgrading a single machine

      - Requires re-architecting applications to handle distributed workloads
- Designing for Horizontal Scaling
  - Stateless Applications
    - Definition
      - Applications that treat each user request independently, with no reliance on previous interactions or session data stored on the server
    - Example
      - A stateless web application where user session data is stored in an external database, allowing any server to handle a user's request
    - Benefits
      - Easily distributed across multiple servers
      - Simplifies horizontal scaling, as no server needs to retain user session information
      - Enables seamless addition or removal of servers without impacting user experience
- Summary
  - Vertical Scaling (Scaling Up)
    - Involves adding resources like memory or processing power to a single machine
    - It's effective but limited by the capacity of the machine or service plan
- Horizontal Scaling (Scaling Out)

        - Distributes workloads across multiple machines, offering nearly limitless scalability and lower long-term costs
        - It requires applications to be designed in a stateless manner
    - Stateless Applications
        - Designed to treat each request independently, making horizontal scaling easier by allowing tasks to be distributed across multiple servers

- **Recovery Strategies**
    - Recovery Strategies
        - Methods and plans designed to restore systems and services to operational status after disruptions or failures, ensuring minimal downtime and data loss
    - Recoverability
        - Definition
            - The organization's ability to restore normal operations as quickly and effectively as possible after an incident
        - Example
            - A company’s disaster recovery plan outlines steps to restore operations when a domain controller goes down due to a cyberattack
        - Key Elements
            - Recovery Time Objectives (RTO)
                - Defines how quickly critical systems need to be restored after an incident
            - Recovery Point Objectives (RPO)

            - Determines how much data loss is acceptable during a disaster

- Backup Strategies
    - Purpose
        - Ensures that data can be restored after an incident, such as a ransomware attack, to prevent data loss
    - Types of Backups
        - Full Backup
            - Creates a complete copy of all data. Takes longer and uses more storage
        - Incremental Backup
            - Saves changes made since the last full or incremental backup, quicker but requires multiple restorations for a full recovery
        - Differential Backup
            - Captures all changes since the last full backup, allowing quicker recovery by only needing the last full backup and the most recent differential
        - Synthetic Full Backup
            - Combines full and incremental/differential backups to create a new, full backup without copying all data again, saving time and storage
- Failover Mechanisms
    - Definition
        - Ensures continued operations when a system fails by automatically switching to a backup system

        - Types of Failover
            - Active-Active
                - Both primary and backup systems run simultaneously
                - If one fails, the other takes over immediately
            - Active-Standby
                - Only the primary system is running, with the standby system ready to take over if the primary fails
        - Example
            - A website remains online despite a server crash because a failover mechanism directs traffic to another active server
    - Disaster Recovery Tests
        - Purpose
            - Regular testing ensures the effectiveness of disaster recovery strategies
        - Types of Tests
            - Tabletop Testing
                - A walkthrough where key team members review the disaster recovery plan step by step
            - Parallel Testing
                - Backup systems are activated alongside production systems to ensure they work without causing downtime
            - Simulation Testing
                - A mock disaster is created to test the response without affecting live systems
            - Full Interruption Testing

            - Systems are taken offline, and backups take over, testing the entire recovery process under real conditions
    - Summary
        - Recoverability
            - Focuses on restoring critical systems and data quickly, defined by RTOs and RPOs, which set recovery speed and acceptable data loss limits
        - Backup Strategies
            - Include different types such as full, incremental, differential, and synthetic backups to ensure data can be recovered efficiently and quickly
        - Failover Mechanisms
            - Active-active or active-standby setups allow seamless operations during system failures, keeping services running
        - Disaster Recovery Tests
            - Regular testing, from tabletop reviews to full interruption testing, validates the recovery plan's effectiveness in real-world scenarios

- **Deployment Strategies**
    - Deployment Strategies
        - Methods for distributing infrastructure to ensure optimal performance, redundancy, and compliance across multiple geographical locations
    - Optimizing Performance
        - Purpose
            - Ensures applications run efficiently for users worldwide by reducing latency and handling traffic spikes

        - Techniques
            - Geographical Server Distribution
                - Deploying servers in different regions to ensure users connect to the closest server, reducing latency
            - Content Delivery Networks (CDNs)
                - Caching content closer to users to speed up load times
                - Tools like Cloudflare or AWS CloudFront assist in this
            - Load Balancing
                - Distributes traffic evenly across servers to prevent overload
                - Examples include AWS Elastic Load Balancing and NGINX
            - Autoscaling
                - Automatically adjusts the number of servers based on demand to maintain performance during traffic spikes
                - AWS Auto Scaling is a commonly used tool
            - Caching
                - Storing frequently accessed data for quicker retrieval, reducing database strain
                - Tools like Redis and Memcached are used for caching
        - Example
            - A global e-commerce site uses CDNs to cache images near users, reducing the time it takes for product pages to load
    - Network Redundancy
        - Purpose
            - Ensures service continuity, even during system failures, by having backup infrastructure in place

        - Techniques
            - Multi-location Infrastructure
                - Deploying servers and data centers in multiple physical locations to act as backups
            - Traffic Rerouting
                - Tools like AWS Route 53 and Azure Traffic Manager reroute traffic to operational servers during failures
            - Multi-Region Failover Systems
                - If one region experiences a failure, traffic is redirected to another region, maintaining availability
        - Example
            - A company's website remains operational during a data center failure because a secondary data center immediately takes over
    - Compliance
        - Purpose
            - Ensures infrastructure and data handling comply with local and international laws, which vary by region
        - Techniques
            - Data Residency
                - Ensuring specific data (e.g., financial, health) remains stored in the country of origin to comply with local laws
            - Compliance Tools
                - Tools like AWS Compliance Center and Azure Policy help enforce rules on where data can be stored and how it must be handled
            - Regional Data Restrictions

            - Setting policies to ensure that data storage complies with local regulations, such as GDPR in Europe
        - Example
            - A healthcare provider stores patient data in specific regions to comply with local data residency laws and GDPR requirements
    - Summary
        - Optimizing Performance
            - Involves deploying infrastructure in various regions, using CDNs, load balancing, autoscaling, and caching to reduce latency and manage traffic efficiently
        - Network Redundancy
            - Ensures continuous service availability by deploying infrastructure across multiple locations, using tools like AWS Route 53 to reroute traffic in case of failure
        - Compliance
            - Focuses on adhering to local data storage and processing regulations, using tools like AWS Compliance Center to manage global infrastructure and ensure data residency rules are followed

Secure Architecture Design

Objective 2.3: Integrate controls in the design of a secure architecture.

- **Data States**
    - Data States
        - The different conditions in which data exists: at rest, in transit, or in use. Each state requires distinct protection methods
    - Data at Rest
        - Definition
            - Data stored on physical devices, such as servers, hard drives, or cloud storage, not actively being accessed
        - Analogy
            - Like books stored on library shelves
        - Protection Methods
            - Encryption
                - Tools
                    - BitLocker (Windows), FileVault (macOS)
                - Cryptographic Methods
                    - 256-bit AES (Advanced Encryption Standard), symmetric encryption
                - Transparent Data Encryption (TDE)
                    - Automatically encrypts data at the storage level, commonly used in databases
            - Access Control

                - Strict access policies and multi-factor authentication to prevent unauthorized access
        - Example
            - A hard drive is encrypted using BitLocker, ensuring the data cannot be read if stolen
    - Data in Transit
        - Definition
            - Data being transferred over networks, vulnerable to interception or tampering
        - Analogy
            - Like books checked out from the library and being transported home
        - Protection Methods
            - Encryption Protocols
                - Transport Layer Security (TLS)
                    - Secures data transmitted over the internet (e.g., HTTPS)
                - IPsec
                    - Used to encrypt data over Virtual Private Networks (VPNs)
                - Secure Shell (SSH)
                    - Encrypts file transfers and remote access
                - SFTP
                    - File Transfer Protocol over SSH, ensuring encrypted file transfers
        - Example

                - Visiting a secure website with HTTPS, where TLS ensures that data transmitted is encrypted
    - Data in Use
        - Definition
            - Data actively being processed, accessed, or handled by applications or users, making it more vulnerable
        - Analogy
            - Like a book being read and handled at home
        - Protection Methods
            - Memory Encryption
                - Encrypts data while it is temporarily stored in RAM during processing
            - Access Control
                - Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC) to limit who can interact with the data
            - Data Masking
                - Shows only partial data to authorized users, hiding sensitive parts (e.g., masking parts of a social security number)
            - Secure Enclaves
                - Tools
                    - Intel SGX (Software Guard Extensions), which create isolated environments for secure data processing
        - Example

            - A database administrator views only the last four digits of a customer’s social security number, with the rest masked for security
    - Summary
        - Data at Rest
            - Stored data that requires encryption (e.g., AES) and access control to ensure security, even if devices are stolen
        - Data in Transit
            - Data being transferred over networks, protected by encryption protocols like TLS and IPsec to prevent interception
        - Data in Use
            - Data actively being processed, requiring memory encryption, access control, and secure environments to prevent unauthorized access or manipulation

- **Data Classification**
    - Data Classification
        - The process of organizing data into categories based on its sensitivity, value, or regulatory requirements
        - This determines the level of security controls and handling procedures for the data
    - Purpose
        - Organizes Data
            - Based on sensitivity, value, or legal requirements
        - Dictates Security

                - The classification level determines the security controls and handling procedures
            - Applies During Data Creation
                - Labels for confidentiality and privacy are applied when data is created
    - Government/Military Classification Model
        - Unclassified
            - Definition
                - Information that presents no risk if disclosed; open to public access
            - Example
                - Army field manuals available on public websites
        - Confidential
            - Definition
                - Data that requires protection but presents minimal risk if disclosed
            - Example
                - The location of a Navy ship
        - Secret
            - Definition
                - More sensitive data that requires strict controls and can only be accessed on specific networks like SIPRNet
            - Example
                - Military strategy documents stored on a network isolated from the public internet
        - Top Secret

                - Definition
                    - The most sensitive data, whose disclosure could have grave consequences
                - Example
                    - Intelligence methods used to track malicious hackers
    - Commercial/Business Classification Model
        - Public
            - Definition
                - Information available to anyone, with no security controls required
            - Example
                - Marketing brochures on a public website
        - Private/Internal
            - Definition
                - Data meant for internal use only, with basic protection to prevent unauthorized access
            - Example
                - Internal employee contact lists
        - Confidential
            - Definition
                - Sensitive data requiring more controls, often protected by encryption and access controls
            - Example
                - Customer financial records in a bank
        - Restricted
            - Definition

            - The highest level of protection in many business models, applied to sensitive and regulated data
        - Example
            - Health records protected under HIPAA laws in the U.S.
- Methods of Protection
    - Encryption
        - Used for higher-classified data (e.g., confidential, secret, top secret) to prevent unauthorized access
    - Access Control
        - Limits who can view or modify certain types of data based on its classification
    - Physical Separation
        - In Government/Military settings, separate networks (e.g., SIPRNet for secret data) and multiple computers for different classification levels
- Summary
    - Data Classification
        - Organizes information based on sensitivity and regulatory requirements
        - It determines the security measures that should be applied, from basic access controls for public data to encryption and isolated networks for classified or confidential information
    - Government/Military Mode
        - Uses levels like unclassified, confidential, secret, and top secret, each requiring increasing levels of protection
    - Commercial/Business Model

            - Often includes labels like public, private/internal, confidential, and restricted, with corresponding security measures based on the sensitivity of the data
        - Legal Requirements
            - Some industries, such as healthcare (under HIPAA), have specific legal obligations for handling and protecting certain types of data

- **Data Labeling and Tagging**
    - Data Labeling and Tagging
        - Data Labeling
            - The process of assigning classification labels to data (e.g., Confidential, Secret, Top Secret) to indicate its sensitivity and handling requirements
        - Data Tagging
            - Adding detailed markers (tags) to specify handling instructions, even within classified data (e.g., PII, PHI, SPI)
    - Data Labels
        - Purpose
            - Assigns a classification level based on sensitivity (e.g., Confidential, Secret, Top Secret)
        - Application Methods
            - Manual Labeling
                - End users assign classification based on the content (e.g., labeling a document "Top Secret")
            - Automatic Labeling

            - Systems apply labels based on preset rules, such as identifying sensitive terms from a "Dirty Word List"
        - Example
            - An email containing the word "bazooka" is automatically labeled "Secret" if "bazooka" is on a classified word list
    - Data Tags
        - Purpose
            - Adds more detailed instructions on handling specific data
        - Examples
            - BIGOT
                - A tag used for documents related to Operation Overlord, requiring additional special handling
            - PII (Personally Identifiable Information)
                - Indicates data that requires protection, even if it is labeled as Unclassified
            - PHI (Personal Health Information)
                - Used in healthcare to identify sensitive patient data that needs special protection
    - Declassification
        - Definition
            - The process of downgrading classified data to a lower level or making it unclassified when it no longer requires protection
        - Example
            - Operation Overlord plans were labeled "Top Secret" during WWII, but after the war, the information was declassified and is now publicly accessible

        - Purpose
            - Frees up resources used to protect outdated data, allowing for focus on current information
    - Data Formats
        - Structured Data
            - Follows a predictable, predefined model
            - Example
                - CSV files, where data like names and addresses are organized in a standard format
        - Unstructured Data
            - Does not follow a specific model and is more flexible
            - Example
                - PowerPoint slides, Word documents, emails, and chat logs
        - Importance
            - Structured and unstructured data require different systems for classification and protection
    - Summary
        - Data Labels
            - Indicate the overall sensitivity of data (e.g., Confidential, Secret, Top Secret)
            - Labeling can be done manually or automatically using tools like Microsoft Data Loss Prevention (DLP)
        - Data Tags
            - Provide more specific handling instructions, such as PII or PHI, ensuring sensitive data is protected based on its unique requirements

        - Declassification
            - Allows classified data to be downgraded when it no longer requires the same level of protection, freeing up security resources
        - Structured vs. Unstructured Data
            - Both data types require different management systems, with structured data following a predefined format and unstructured data having no specific organization

- **Data Loss Prevention (DLP)**
    - Data Loss Prevention (DLP)
        - A system used to detect, prevent, and respond to unauthorized access, transmission, or use of sensitive data
        - It manages both data at rest and data in transit to ensure data security and compliance
    - Core Functions
        - Data Discovery
            - Identifies and labels sensitive data based on predefined rules (e.g., confidentiality, privacy, sensitivity)
            - This is the foundation of a DLP system
        - Data at Rest
            - Protects data stored in files, databases, or storage devices from unauthorized access
        - Data in Transit
            - Monitors and protects data as it moves across the network, between systems or storage devices

- Components of a DLP System
  - Policy Server
    - Configures rule sets to classify data
    - Logs incidents of rule violations and generates reports
  - Endpoint Agent
    - Enforces DLP policies on individual devices, even when offline (e.g., blocking a user from copying sensitive data to a USB drive)
  - Network Agent
    - Monitors data in transit by scanning web traffic, emails, and messaging platforms
    - Detects and enforces policies for both structured (e.g., JSON, CSV) and unstructured (e.g., Word docs, emails) data
- Policy Management
  - Allowlist
    - Blocks all data except for specific, allowed data
  - Denylist
    - Allows all data unless it is explicitly blocked
  - Automated Data Discovery
    - DLP systems can automate data discovery and apply classification labels (e.g., PII, PHI) based on rules
- DLP Actions in Response to Violations
  - Alert
    - Logs the incident and notifies administrators, but allows data transmission (detective control)
  - Block

      - Stops the data transfer and alerts administrators (preventive control)
    - Quarantine
      - Removes user access to the data by encrypting it or rendering it unreadable
    - Tombstone
      - Replaces the original file with a message indicating a policy violation and instructions for regaining access
- Additional DLP Features
    - External Media Blocking
      - Prevents data from being copied to USB drives or CDs
    - Print Blocking
      - Stops sensitive documents from being printed
    - RDP Blocking
      - Prevents data transfer between remote and local systems during remote access
    - Clipboard Privacy
      - Prevents users from copying sensitive data into unauthorized applications
    - VDI (Virtual Desktop Infrastructure) Security
      - Applies DLP agents to virtual environments just like physical desktops
- Summary
    - Data Loss Prevention (DLP)
      - Protects sensitive data by detecting, preventing, and responding to unauthorized access, use, or transmission

        - DLP manages data at rest (stored data) and data in transit (moving data), using tools like policy servers, endpoint agents, and network agents to enforce security policies.
        - It relies on data discovery to identify sensitive information and applies policies to restrict its use and prevent breaches
    - Actions in Case of Violations
        - DLP systems can alert administrators, block data transfers, quarantine data, or replace it with a tombstone message indicating a policy violation
    - Integration with Other Security Tools
        - DLP integrates with additional controls like blocking external media, preventing printing, and protecting virtual desktop environments, enhancing data security across different platforms

- **Hybrid Infrastructures**
  - Hybrid Infrastructure
    - An environment that combines on-premises systems with cloud-based resources, requiring integrated security measures to protect data across both environments
  - On-Premises Infrastructure
    - Definition
      - Physical hardware, software, and networking resources hosted and managed within an organization's own facilities
    - Control
      - Complete control over physical security, configurations, and customization

        - Costs
            - Requires significant investment in hardware, software, maintenance, and personnel
        - Advantages
            - Tailored security and compliance for sensitive data and critical workloads
        - Example
            - An organization running its data center with all hardware and software managed on-site
    - Cloud-Based Infrastructure
        - Definition
            - Computing resources (e.g., storage, virtual machines, databases) delivered by third-party providers (e.g., AWS, Azure) over the internet
        - Scalability
            - Resources can be scaled up or down as needed; pay only for what is used
        - Service Models
            - IaaS (Infrastructure as a Service)
                - Rent virtualized servers and storage for custom configurations
            - PaaS (Platform as a Service)
                - Development platforms for building and deploying applications without managing hardware
            - SaaS (Software as a Service)

            - Ready-to-use applications hosted in the cloud (e.g., email, CRM)
    - Advantages
        - Flexible, cost-effective, offers high availability, redundancy, and disaster recovery
- Hybrid Infrastructure
    - Definition
        - A combination of on-premises and cloud-based resources, allowing organizations to balance control and scalability
    - Advantages
        - Flexibility
            - Organizations can use on-premises systems for sensitive data while leveraging the cloud for dynamic needs
        - Scalability
            - Cloud resources can be scaled on-demand without large hardware investments
        - Resilience
            - Cloud-based redundancy and disaster recovery options improve operational resilience
    - Challenges
        - Consistent Security
            - Extending traditional security controls (e.g., firewalls, encryption) across both environments can be complex
        - Unified Access Management

- Identity and Access Management (IAM) must ensure consistent authentication and authorization across on-premises and cloud systems

- Security Considerations
  - Firewalls
    - Must be extended to protect both on-premises and cloud environments
  - Encryption
    - Applied to data at rest and in transit in both environments to ensure security and compliance
  - Identity and Access Management (IAM)
    - Critical for consistent authentication and authorization across both environments
  - Policy Enforcement
    - DLP and other security policies must be enforced consistently across hybrid environments to prevent data loss or breaches
- Tools for Securing Hybrid Infrastructures
  - IAM Solutions
    - Ensures unified access management across on-premises and cloud environments
  - Encryption Tools
    - Applied uniformly across both environments to protect sensitive data
  - Data Loss Prevention (DLP)
    - Ensures policies for blocking unauthorized data transfers are consistently enforced

        - Policy Enforcement Tools
            - Monitors and enforces security policies across all resources, both in the cloud and on-premises
    - Summary
        - Hybrid Infrastructure
            - Combines on-premises systems with cloud-based resources, allowing organizations to maintain control over sensitive data while taking advantage of the cloud's scalability and flexibility
        - Security Requirements
            - Extending traditional security measures like firewalls, encryption, and access management is essential for protecting both on-premises and cloud-based systems
            - IAM solutions, encryption, and consistent policy enforcement are key to securing hybrid environments
        - Challenges
            - Hybrid infrastructures require robust strategies to ensure consistent security across both environments, as differences in management and control between on-premises and cloud systems can create security gaps

- **Third-Party Integrations**
    - Third-party Integrations
        - The process of securely connecting external services or applications to an organization's internal systems, often facilitated through Application Programming Interfaces (APIs) for communication between systems
    - Data Encryption

        - Purpose
            - Protects the confidentiality and integrity of sensitive data during transmission and storage by making it unreadable to unauthorized users
        - Application
            - Data in Transit
                - Encrypt data while it moves between your organization and third-party services
            - Data at Rest
                - Encrypt stored data within third-party systems
        - Encryption Standards
            - Must meet regulatory requirements (e.g., PCI DSS) for customer data protection, ensuring that both the organization and third-party providers use robust and audited encryption methodologies
    - Data Protection
        - Definition
            - Ensuring data security beyond encryption by applying policies and controls to safeguard data across third-party environments
        - Components
            - Access Controls
                - Multi-factor authentication (MFA), Role-Based Access Control (RBAC), and restricting access to authorized users only
            - Regular Audits

- ○ Security assessments and audits to ensure compliance and discover vulnerabilities
- ● End-to-End Protection
  - ○ Extends across the entire data lifecycle (storage, processing, transmission, and archiving)

- ○ API Design
  - ■ Role
    - ● APIs are essential for integrating with external services, making secure API design critical for safeguarding data
  - ■ Best Practices
    - ● Authentication and Authorization
      - ○ Use secure protocols like OAuth 2.0 to verify access rights and restrict data access
    - ● Principle of Least Privilege
      - ○ Limit API exposure to only the necessary data for functionality
    - ● Rate Limiting
      - ○ Prevents abuse or Denial-of-Service (DoS) attacks by limiting API call rates
    - ● Input Validation
      - ○ Protects against common API vulnerabilities such as SQL injection and cross-site scripting (XSS)
- ○ Monitoring and Logging
  - ■ Monitoring
    - ● Tracks real-time interactions between internal systems and third-party services to detect anomalies or suspicious activity

        - Logging
            - Provides a historical record of third-party interactions, showing when data was accessed, by whom, and what actions were performed
            - Importance
                - Essential for security investigations, compliance validation (e.g., GDPR, HIPAA), and maintaining accountability for third-party interactions
            - Compliance
                - Logs help meet regulatory audit requirements and demonstrate data protection measures
    - Summary
        - Third-party Integrations
            - Involve securely connecting external services to an organization's internal systems using APIs, requiring stringent security measures
        - Security Measures
            - Encryption
                - Ensures data confidentiality during transmission and storage, meeting industry standards
            - Data Protection
                - Involves extending access control, encryption, and security policies to third-party systems, ensuring end-to-end data security
            - API Design
                - Requires strong authentication, limited data exposure, and protection against malicious activities

        - Monitoring and Logging
            - Enables real-time detection of suspicious activities and provides a comprehensive audit trail for compliance and investigations

- **Attack Surface Management**
    - Attack Surface Management
        - Identifying, reducing, and continuously monitoring all potential paths of attack within a system's architecture to minimize exposure to security threats
    - Hardening
        - Definition
            - The process of securing a system by reducing its attack surface through the removal of unnecessary services, applications, and applying security patches
        - Purpose
            - Reduces the potential entry points for attackers, thereby increasing the system's resilience to attacks
        - Example
            - Disabling unused services like the CUPS daemon on a Linux server if printing is not required
        - Key Actions
            - Disable unnecessary services and ports
            - Install and maintain antivirus, host-based firewalls, and log collection agents

        - Implement hardware security features (e.g., UEFI, TPM, HSM) for secure boot and cryptographic functions

- Defense-in-Depth
    - Definition: A layered approach to security where multiple security controls work together to protect a system.
    - Purpose: Ensures redundancy in security; if one defense mechanism fails, others are in place to protect the system.
    - Example Layers
        - Firewall
            - Filters network traffic to prevent unauthorized access
        - Intrusion Detection System (IDS)
            - Monitors network traffic for suspicious activity and alerts administrators
        - Data Encryption
            - Protects data in transit and at rest, ensuring it remains secure if intercepted
        - Endpoint Protection Software
            - Safeguards devices from malware and other threats
        - Physical Security
            - Protects servers and other hardware from unauthorized access
    - Benefit
        - Creates a resilient security posture by layering protections for multiple potential threat vectors
- Vulnerability Management
    - Definition

        - The process of regularly identifying, prioritizing, and addressing security vulnerabilities within a system
    - Purpose
        - Helps maintain system security by promptly addressing weaknesses before they are exploited
    - Examples
        - Running regular scans with tools like Nessus or CIS-CAT to detect vulnerabilities and ensure compliance
        - Applying patches for critical vulnerabilities, such as in Apache HTTP Server
    - Additional Actions
        - Remove unnecessary services, close unused ports, and disable default accounts
        - Use automated tools (e.g., SCAP compliance checkers) to continuously monitor system configurations for compliance
- Legacy Components
    - Definition
        - Systems or components approaching or past their End of Life (EOL) or End of Support (EOS), making them vulnerable due to a lack of security updates
    - Purpose
        - Legacy components often become security risks due to unpatched vulnerabilities that attackers can exploit indefinitely
    - Example
        - Microsoft will cease support for Windows Server 2019 in 2029, making it vulnerable to new threats post-EOS

        - Mitigation Strategy
            - Plan upgrades or replacements for legacy systems well in advance to maintain support and security
            - Consider isolating legacy systems if they must remain in use, reducing their exposure within the network
    - Summary
        - Attack Surface Management is essential for securing an organization's infrastructure
        - Key components include
            - Hardening
                - Secures systems by disabling unnecessary features, applying patches, and using security tools
            - Defense-in-Depth
                - Uses multiple layers of security, including firewalls, IDS, encryption, and endpoint protection, to safeguard systems even if one defense layer fails
            - Vulnerability Management
                - Regularly scans and addresses security weaknesses to keep systems updated and secure
            - Legacy Components
                - Manages or replaces unsupported systems to avoid unpatched vulnerabilities that could be exploited
        - Together, these strategies minimize entry points, strengthen defenses, and ensure ongoing protection against security threats across the network

- **Control Effectiveness**
  - Control Effectiveness
    - The degree to which security controls mitigate risks and protect organizational assets
  - Metrics
    - Definition
      - Quantitative measures that evaluate the performance and impact of security controls
    - Examples
      - Incident Response Time
        - Measures how long it takes to detect and resolve security threats
        - A shorter response time indicates an efficient incident management process
      - Vulnerabilities Mitigated
        - Tracks the number of vulnerabilities addressed over a specific period
        - A higher count reflects effective vulnerability management
      - Patch Compliance Rate
        - Measures the speed at which patches are applied after vulnerabilities are discovered
      - False Positive Rate
        - Indicates how often security devices incorrectly flag benign events, affecting the overall accuracy of security controls
  - Scanning
    - Definition

            - The process of using tools to detect vulnerabilities and misconfigurations within systems
        - Types of Scanners
            - Network Scanners
                - (e.g., OpenVAS, Tenable Nessus, Qualys) identify network-level issues like open ports or outdated protocols
            - Web Application Scanners
                - (e.g., Nikto) detect application-specific vulnerabilities like SQL injection and cross-site scripting (XSS)
            - Database, Cloud, and Container Scanners
                - Focus on securing specific parts of the infrastructure
        - Importance
            - Scanning helps prioritize vulnerabilities for remediation based on severity and potential impact
    - Assessments
        - Definition
            - Evaluations of security controls through audits, penetration tests, or compliance checks to ensure they are functioning as intended
        - Types of Assessments
            - Audits
                - Evaluate compliance with internal policies or regulatory standards (e.g., HIPAA, PCI DSS)
            - Penetration Testing
                - Simulates real-world attacks to test the effectiveness of security defenses
            - Compliance Checks

            - Ensure adherence to industry-specific regulations and standards
        - Metrics Collected in Assessments
            - Compliance Violations
                - Tracks gaps that need to be addressed to meet regulatory or legal standards
            - Time to Detect and Respond
                - Measures the speed at which threats are identified and mitigated during penetration testing
            - Control Implementation
                - Assesses the percentage of security controls that are fully implemented and operational
- Summary
    - Control Effectiveness
        - Assesses how well security controls mitigate risks and protect assets
    - Metrics
        - Provide quantitative measures like incident response times, vulnerabilities mitigated, and patch compliance rates to evaluate control performance
    - Scanning
        - Uses tools to detect system vulnerabilities and misconfigurations, allowing teams to prioritize remediation efforts
    - Assessments
        - Through audits, penetration testing, and compliance checks, assessments evaluate the overall effectiveness of security

controls, ensuring they function as expected and identifying areas for improvement

Security in Systems

Objective 2.2: Implement security in the early and subsequent states of a system’s life cycle

- **07_02: Hardware Assurance**
    - Hardware Assurance
        - The process of ensuring that physical components are secure, reliable, and free from malicious alterations or defects
    - Certification
        - Definition
            - A formal evaluation process to ensure hardware components meet specific security standards and operate as expected
        - Framework
            - Common Criteria (CC)
                - Evaluation Assurance Level (EAL)
                    - Defines the depth of evaluation, ranging from EAL 1 (functionally tested) to EAL 7 (formally verified design and tested)
                - Protection Profile (PP)
                    - Describes the expected security environment for a category of products (e.g., microchips) and outlines potential threats and countermeasures
                - Security Target
                    - Details how a product will meet the security requirements in the Protection Profile (e.g.,

implementing tamper resistance, encryption, and secure boot processes)

        - Trusted Foundry Program
            - Ensures secure manufacturing of critical hardware for national defense
            - Certifies manufacturing facilities to prevent tampering or insertion of malicious components during production
            - Includes strict protocols for access control, supply chain security, and data protection
    - Validation
        - Definition
            - The ongoing process of testing and verifying that hardware continues to meet design and security standards, particularly after updates or changes
        - Frameworks
            - NIST 800-161
                - Focuses on supply chain risk management for information and communication technology systems
                - Emphasizes continuous validation throughout hardware development, manufacturing, and delivery
            - NIST 800-171
                - Provides guidelines for protecting Controlled Unclassified Information (CUI) in non-federal systems, especially important for contractors working with the federal government
            - IoT Cybersecurity Improvement Act of 2020

      - Sets cybersecurity standards for IoT devices used by the federal government, requiring vulnerability management, patching, and authentication

- Examples
  - Common Criteria Certification
    - A new microchip undergoing certification to meet EAL 4, ensuring it defends against physical tampering and uses strong encryption for data protection
  - Trusted Foundry Program
    - A facility certified under the Trusted Foundry Program to produce microelectronics for national defense, ensuring that hardware components are not compromised during production
  - Validation Using NIST 800-161
    - A contractor regularly validates hardware components used in communication systems to meet NIST 800-161 supply chain risk management guidelines
  - IoT Cybersecurity Improvement Act
    - A company developing IoT devices for government use validates that its devices meet the cybersecurity requirements outlined in the IoT Cybersecurity Improvement Act of 2020
- Summary
  - Hardware Assurance ensures that physical components are secure, reliable, and free from tampering or defects
  - Certification
    - Verifies that hardware meets specific security standards (e.g., Common Criteria and the Trusted Foundry Program)

        - Validation
            - Continuously verifies that hardware remains secure throughout its lifecycle, following guidelines like NIST 800-161, NIST 800-171, and the IoT Cybersecurity Improvement Act of 2020

- **Security Requirements**
    - Security Requirements
        - Specific criteria that a system must meet to protect against threats and vulnerabilities
    - Functional Security Requirements
        - Definition
            - Security features and behaviors that a system must have to protect against threats
        - Examples
            - Authentication Mechanisms
                - Ensures that only authorized users can access the system (e.g., passwords, biometrics, multi-factor authentication)
            - Access Controls
                - Determines which users have permission to access specific system resources (e.g., role-based access control, attribute-based access control)
            - Encryption Protocols
                - Secures data by converting it into unreadable ciphertext during transmission or storage to prevent unauthorized access (e.g., AES encryption, SSL/TLS protocols)
    - Non-Functional Security Requirements

        - Definition
            - Qualities of the system that ensure it performs reliably and securely under stress or operational challenges
        - Examples
            - Performance
                - Ensures that the system remains efficient and fast even when security features like encryption are applied.
            - Reliability
                - Guarantees that the system remains operational during high traffic loads or partial failures (e.g., redundancy and failover mechanisms)
            - Usability
                - Ensures that security measures do not overly hinder legitimate users' access to the system
    - Security vs Usability Trade-offs
        - Multi-factor Authentication (MFA)
            - Security
                - Requires users to authenticate using two or more methods (e.g., password, fingerprint, mobile code)
            - Usability Impact
                - MFA adds complexity and can slow down legitimate users, especially if required frequently
            - Solution
                - Implementing single sign-on (SSO) systems can reduce the number of times MFA is required while maintaining security

        - Encryption
            - Security
                - Protects sensitive data by encrypting it during transmission and storage
            - Usability Impact
                - Encryption can introduce latency, causing slower system performance, especially in environments requiring real-time data processing (e.g., financial systems)
            - Trade-off
                - Administrators may need to choose between stronger encryption algorithms and ensuring system efficiency
        - Access Control Policies
            - Security
                - Granular access control limits what specific users can access within the system, reducing the risk of insider threats
            - Usability Impact
                - Overly restrictive access control can delay workflows, requiring multiple approvals for simple tasks
            - Trade-off
                - Finding a balance between security and efficient workflows is crucial to avoid hindering legitimate operations
    - Summary
        - Security Requirements
            - Divided into functional and non-functional categories

            - Functional Security Requirements include specific security features like authentication, access controls, and encryption
            - Non-Functional Security Requirements focus on system qualities like performance, reliability, and usability under stress
    - Security vs Usability Trade-off
        - Achieving robust security can sometimes reduce system efficiency or user convenience, and balancing these aspects is key to effective security implementation

- **Software Assurance**
    - Software Assurance
        - The process of ensuring software is developed and maintained securely and reliably, protecting it from vulnerabilities and threats
    - Software Bill of Materials (SBoM)
        - Definition
            - A detailed list of every component, library, and dependency used in a software application, providing transparency.
        - Purpose
            - Helps track components to identify any vulnerabilities or security issues in third-party dependencies
        - Components
            - Direct Dependencies
                - Libraries or modules explicitly used in the software (e.g., React, Express.js)

        - Transitive Dependencies
            - Components that the direct dependencies rely on (e.g., Library A depends on Library B)
    - Example
        - If a vulnerability is discovered in a specific version of a library (e.g., React), the SBoM helps quickly identify its use in your software, allowing for fast updates or patches
- Software Composition Analysis (SCA)
    - Definition
        - A process used to scan software for third-party dependencies and check them for known vulnerabilities
    - Purpose
        - Ensures that all third-party libraries and frameworks used in your software are secure and up to date
    - Tools
        - OWASP Dependency-Check
            - Scans software for publicly disclosed vulnerabilities
        - OWASP Dependency-Track
            - Continuously monitors the security of components within a software project
    - Example
        - The Equifax breach occurred because of a vulnerability (CVE-2017-5638) in Apache Struts
        - SCA tools could have detected this vulnerability, allowing for timely patching
- Formal Methods of Validation

    - Definition
        - Mathematically-based techniques used to prove that software is free from errors, bugs, and security vulnerabilities
    - Purpose
        - Guarantees software behaves as expected under all conditions, especially in critical systems where errors can lead to major consequences (e.g., aviation, healthcare, finance)
    - Techniques
        - Model Checking
            - Verifies a system model against predefined rules (e.g., SPIN tool for distributed systems)
        - Theorem Proving
            - Uses formal proofs to validate software correctness (e.g., Coq and Isabelle tools)
        - Abstract Interpretation
            - Analyzes software without running it to detect runtime errors (e.g., Astrée tool used in Airbus flight control software)
    - Example
        - A traffic light control system can be validated using formal methods to ensure no conflicting signals are given, preventing accidents
- Summary
    - Software Bill of Materials (SBoM) provides a detailed breakdown of all components in a software application, ensuring transparency and quick identification of vulnerabilities

        - Software Composition Analysis (SCA) scans and checks third-party components for vulnerabilities, ensuring secure and up-to-date software
        - Formal Methods of Validation use mathematical techniques to guarantee software correctness, safety, and security, especially in critical systems

- **Supply Chain Assurance**
    - Supply Chain Assurance
        - The process of ensuring that all components and processes within a supply chain meet security and quality standards to prevent vulnerabilities from being introduced into a final system
    - Hardware Supply Chain Risk Management
        - Definition
            - Focuses on ensuring the security and integrity of physical components throughout the supply chain
        - Risks
            - Tampering
                - Unauthorized modifications to components during manufacturing or distribution that could introduce malicious hardware (e.g., backdoors)
            - Counterfeit Components
                - Fake or substandard parts that are difficult to detect, more prone to failure, and may be designed with malicious intent
        - Mitigation Strategies
            - Hardware Authentication

                - Embedding cryptographic modules or digital signatures into hardware components to verify their authenticity and integrity
            - Transparency
                - Maintaining close relationships with suppliers and requiring detailed documentation about the origins of each component (e.g., Apple and Cisco’s transparency practices)
- Software Supply Chain Risk Management
    - Definition
        - Ensures that the software components integrated into a system, including third-party libraries and open-source tools, are secure and free from vulnerabilities
    - Risks
        - Hidden Vulnerabilities
            - Unknown weaknesses in third-party software or libraries that could compromise security
        - Software Supply Chain Attacks
            - Malicious code inserted into trusted software, as seen in the SolarWinds breach, which gave attackers access to sensitive systems through a compromised software update
    - Mitigation Strategies
        - Software Composition Analysis (SCA)
            - Tools like OWASP Dependency-Check and OWASP Dependency-Track scan software for vulnerable or outdated components

                - Software Audits
                    - Regular assessment of third-party software for vulnerabilities, licensing issues, and compliance with security standards
                - Digital Signatures
                    - Using certificates to verify the authenticity and integrity of software updates to prevent malicious code from being introduced during delivery
    - Summary
        - Hardware Supply Chain Risk Management protects physical components from tampering and counterfeit parts by using methods like hardware authentication and supply chain transparency
        - Software Supply Chain Risk Management focuses on ensuring the security of third-party libraries and software tools by using Software Composition Analysis, regular audits, and digital signatures for software updates
        - Both hardware and software supply chain assurance emphasize transparency, authenticity, and regular assessments to mitigate potential threats throughout the supply chain

- **Pre-Deployment Testing**
    - Pre-Deployment Testing
        - The process of evaluating software for security vulnerabilities, functional issues, and performance concerns before releasing it into a live environment
    - Static Application Security Testing (SAST)
        - Definition

            - SAST analyzes a software application's source code or binaries for vulnerabilities without executing the software
        - Objective
            - Identifies potential security weaknesses early in the development process
        - Tools
            - Checkmarx, Veracode, Fortify.
        - Strengths
            - Catches vulnerabilities early in development
            - Automated tools flag common issues like SQL injection and buffer overflows
            - Manual reviews can find more complex issues (e.g., logical flaws)
        - Weaknesses
            - False Positives
                - Flagging non-vulnerabilities, leading to time-consuming investigations
            - Runtime Vulnerabilities
                - Cannot detect issues that only appear when the application runs
    - Dynamic Application Security Testing (DAST)
        - Definition
            - DAST evaluates the security of a running application by simulating real-time attacks
        - Objective
            - Detects flaws that only occur when the application is operating, such as SQL injection or cross-site scripting

        - Tools
            - Zed Attack Proxy (ZAP), Burp Suite, Peach Fuzzer (for fuzzing)
        - Strengths
            - Finds vulnerabilities in real-time that are missed by static analysis
            - Excellent for identifying flaws in authentication processes
        - Weaknesses
            - Internal Flaws
                - Cannot see vulnerabilities within the source code
            - False Positives
                - Can flag issues that are not actually exploitable
            - Business Logic Vulnerabilities
                - Less effective at detecting complex vulnerabilities related to application logic
    - Interactive Application Security Testing (IAST)
        - Definition
            - IAST is a hybrid approach combining SAST and DAST, providing real-time feedback on vulnerabilities while the application runs
        - Objective
            - Offers continuous, real-time analysis of both the code and system behavior
        - Tools
            - Contrast Security, Veracode
        - Strengths
            - Provides comprehensive security assessments during functional testing

            - Immediate feedback allows developers to address issues on the spot
        - Weaknesses
            - Missed Vulnerabilities
                - Parts of the application that aren't tested may go unchecked
            - Performance Impacts
                - Can slow down application tests or affect development workflows
    - Summary
        - SAST checks source code without running the application, finding issues early but missing runtime vulnerabilities
        - DAST evaluates a running application, simulating real-time attacks to detect security flaws in operation, though it can miss internal issues
        - IAST combines SAST and DAST for real-time analysis of both code and runtime behavior, providing continuous feedback but potentially missing untested areas

- **Post-Deployment Testing**
    - Post-Deployment Testing
        - The process of evaluating software in a live environment after deployment to identify and address any security vulnerabilities and performance issues, ensuring the software remains secure and performs as expected
    - Software Vulnerability Analysis
        - Definition

            - Identifies security weaknesses in a live environment by scanning a deployed application and its surrounding infrastructure
        - Purpose
            - Finds and addresses vulnerabilities that attackers could exploit, continuously monitoring for new threats as they emerge
        - Example
            - OpenVAS, an open-source vulnerability scanner, detects issues like outdated software, insecure configurations, and missing patches
        - Benefits
            - Enables continuous monitoring and detection of vulnerabilities
            - Helps prioritize vulnerabilities based on severity, reducing risks
        - Limitations
            - May not detect zero-day vulnerabilities, as scanners rely on databases of known threats
            - Can produce false positives, leading to potential delays as non-issues are investigated
    - Runtime Application Self-Protection (RASP)
        - Definition
            - A security technology that monitors and protects applications in real time by analyzing their behavior within the live environment
        - Purpose
            - Provides real-time threat detection and protection, blocking detected threats and suspicious activities directly within the application
        - Example

        - RASP tools like Contrast Security or Imperva RASP detect and prevent runtime attacks, such as SQL injections, by analyzing unexpected input patterns and stopping them before they affect the application
    - Benefits
        - Offers protection against new, unknown threats that bypass traditional defenses
        - Blocks malicious actions within the application itself based on real-time behavior
    - Limitations
        - Potential performance impact, especially in resource-intensive applications
        - Not a replacement for vulnerability scanning or code reviews, which address foundational security flaws
- Summary
    - Post-deployment testing ensures the ongoing security and performance of software in a live environment by identifying vulnerabilities and providing real-time protection
        - Software Vulnerability Analysis
            - Regularly scans for known vulnerabilities and provides reports for remediation
            - Tools like OpenVAS help detect issues across systems, although they may miss zero-day threats and generate false positives
        - Runtime Application Self-Protection (RASP)

        - Integrates within the application, monitoring and blocking malicious behavior in real-time to prevent runtime attacks
        - While effective for immediate threat detection, it can impact performance and is not a substitute for pre-deployment testing
    - Together, these practices provide a comprehensive approach to securing applications after they go live, allowing organizations to continuously monitor and protect against evolving threats

- **Continuous Integration/Continuous Deployment (CI/CD) Management**
  - CI/CD Management
    - The process of automating the integration of code changes, testing, and deployment of applications to ensure consistent and efficient software delivery
  - Coding Standards
    - Definition
      - A set of guidelines for writing code in a consistent and readable manner
    - Objective
      - To ensure code is understandable and maintainable by the entire development team
    - Tools
      - Prettier, ESLint
    - Key Elements
      - Consistency

            - Ensures code adheres to uniform practices (e.g., CamelCase vs snake_case)
        - Readability
            - Makes it easier for developers to work on each other's code
        - Maintenance
            - Reduces the risk of errors by following a standardized format
- Linting
    - Definition
        - The process of analyzing code for potential errors and adherence to best practices before it runs
    - Objective
        - To catch bugs, errors, and security vulnerabilities early in the development process
    - Tools
        - ESLint (for JavaScript), Flake8 (for Python)
    - Key Elements
        - Error Detection: Flags issues like unused variables or unreachable code
        - Best Practices
            - Ensures that code follows industry standards and reduces potential security risks
        - Early Warnings
            - Helps prevent problematic code from progressing to production

- Branch Protection
  - Definition
    - A set of rules applied to version control systems that restricts direct changes to important branches
  - Objective
    - To ensure that only thoroughly tested and reviewed code is merged into the production environment
  - Tools
    - GitHub Actions, CircleCI
  - Key Elements
    - Code Reviews
      - Requires that changes are reviewed by other team members
    - Automated Testing
      - Runs tests and linting as part of the merge process
    - Protection of Main Branch
      - Ensures that untested or unreviewed code cannot be merged directly into critical branches
- Continuous Improvement
  - Definition
    - The ongoing process of refining and optimizing the CI/CD pipeline based on feedback, performance, and new security practices
  - Objective
    - To ensure that the pipeline remains efficient, secure, and up-to-date over time
  - Tools

            - Dependabot, monitoring tools
        - Key Elements
            - Pipeline Efficiency
                - Optimizing build times, eliminating bottlenecks
            - Security Updates
                - Regularly scanning for vulnerabilities and automating dependency updates
            - Workflow Optimization
                - Introducing new tools or techniques to improve the overall development and deployment process
    - Summary
        - Coding Standards ensure that all developers follow the same guidelines, making code more maintainable and less error-prone
        - Linting acts as an early warning system to catch errors and ensure best practices, making code safer and cleaner
        - Branch Protection safeguards the main branches by requiring reviews and tests before code can be merged, preventing issues from reaching production
        - Continuous Improvement involves constantly evaluating and refining the CI/CD process to improve efficiency, security, and performance

- **Continuous Integration/Continuous Deployment (CI/CD) Testing**
    - CI/CD Testing
        - The systematic evaluation of code changes to identify and resolve security vulnerabilities and functional issues before deployment to production

- Canary Testing
  - Definition
    - A phased deployment approach where a new feature or update is rolled out to a small group of users before the full release
  - Objective
    - To detect issues such as security vulnerabilities, crashes, or performance problems on a limited scale
  - Key Elements
    - Early Detection
      - Monitors system behavior with a small group before full-scale deployment
    - Rollback Capability
      - Allows for easy rollback if issues are detected, minimizing impact on the entire user base
    - Real-World Testing
      - Provides feedback from live usage conditions, ensuring quality control
- Regression Testing
  - Definition
    - Re-running existing test cases to ensure that new code changes do not negatively impact previously functioning features
  - Objective
    - To confirm that system functionality remains stable after new features or updates are introduced
  - Key Elements
    - Consistency

                - Ensures that new updates do not break existing functionality
            - Automation
                - Commonly automated in CI/CD pipelines to save time and resources
            - Stability
                - Focuses on maintaining system stability and preventing regressions
    - Automated Testing and Retesting
        - Definition
            - Automated scripts are run every time code is integrated to detect bugs early in the process
        - Objective
            - To accelerate the testing process and reduce human error
        - Key Elements
            - Early Detection
                - Automated tests run frequently to catch issues before they affect the main codebase
            - Retesting
                - Ensures that fixed bugs are resolved and that no new issues are introduced
            - Types Covered
                - Includes unit testing, integration testing, and regression testing
    - Unit Testing
        - Definition

      - The process of testing individual components or functions in isolation
    - Objective
      - To verify that each part of the code behaves as expected
    - Key Elements
      - Precision
        - Focuses on small, specific units of code
      - Early Detection
        - Detects issues early in development, allowing for quick fixes
      - Complemented by Integration Tests
        - Unit tests work alongside integration tests to ensure overall system functionality
  - Integration Testing
    - Definition
      - Tests the interaction between different units or modules of a system to ensure they work together properly
    - Objective
      - To verify that components and services interact as expected in a complex environment
    - Key Elements
      - Cross-Module Interaction
        - Tests how different components (e.g., front-end and back-end, APIs, databases) communicate
      - Identifying Communication Issues

            - Helps detect data mismatches, broken APIs, or misconfigurations
        - Essential for Multi-Layered Systems
            - Critical for systems with microservices or distributed components
    - Summary
        - Canary Testing
            - A controlled deployment to a small group of users to identify issues before releasing updates widely
        - Regression Testing
            - Ensures that new code does not interfere with existing functionality, often automated to save time
        - Automated Testing and Retesting
            - Continuously runs tests and retests to ensure early detection and resolution of issues
        - Unit Testing
            - Tests individual code components to ensure they work as intended in isolation
        - Integration Testing
            - Focuses on how different components of a system work together, ensuring smooth communication and functionality across services

- **End-of-Life (EOL) Considerations**
    - End-of-Life (EOL)
        - The stage in a product's lifecycle when manufacturers or vendors stop providing updates, including security patches

        - This leaves systems vulnerable to exploitation
    - Lifecycle Management
        - Definition
            - Planning and managing a system's lifespan from deployment to decommissioning, ensuring outdated hardware or software is replaced before posing security risks
        - Key Elements
            - Configuration Management Database (CMDB)
                - A database tracking all hardware and software within an organization, helping forecast when systems will reach EOL or End-of-Service-Life (EOSL)
            - EOL (End-of-Life)
                - When the vendor slows or stops updates
                - The product remains functional but no longer receives regular support
            - EOSL (End-of-Service-Life)
                - When the vendor halts all support, leaving the system highly vulnerable to security risks
            - Planning for Upgrades or Replacement
                - Systems approaching EOL or EOSL should be upgraded or replaced to maintain security and functionality
    - Data Migration
        - Definition
            - The process of transferring data from one system or location to another before decommissioning the old system
        - Key Elements

        - Data Integrity and Compatibility
            - Ensuring data remains intact and compatible with the new system, avoiding corruption during the migration
        - Migration Strategies
            - Incremental Migration
                - Data is moved in phases to prevent extended downtime
            - Parallel Migration
                - Running both old and new systems simultaneously for a smooth transition
            - Big Bang Migration
                - Moving all data in one step during planned downtime
        - Data Transformation
            - Adjusting data schemas or file structures to ensure compatibility with the new system
        - Pilot Migrations
            - Testing smaller datasets before full migration to validate data handling and functionality
- Secure Deletion
    - Definition
        - The process of securely wiping data from old systems after migration to prevent unauthorized recovery
    - Key Elements
        - NIST-Compliant Tools

            - Tools that meet security standards for disk erasure, ensuring data is unrecoverable
        - Physical Media Destruction
            - For sensitive environments, physical destruction methods like degaussing or shredding may be necessary
        - Methods
            - Techniques such as the DoD 5220.22-M method (a three-pass wipe process) ensure secure overwriting of data
- Summary
    - Lifecycle Management
        - Involves tracking hardware and software lifecycles through tools like CMDBs and planning upgrades or replacements before EOL/EOSL
    - Data Migration
        - Moves data to a new system before decommissioning the old one, considering data integrity, compatibility, and secure deletion of residual data
    - Secure Deletion
        - Ensures data from decommissioned systems is rendered irretrievable to prevent unauthorized access

Access, Authentication, Authorization

Objective 2.4: Apply security to the design of access, authentication, and authorization systems

- **08_02: Access Control Systems**
    - Access Control Systems
        - Mechanisms designed to ensure only authorized individuals can access resources based on their identity and permissions
        - These systems are categorized as physical or logical
    - Physical Access Control Systems
        - Definition
            - Systems that manage entry to physical locations like office buildings or server rooms, using methods that verify a person's identity before granting access
        - Technologies Used
            - RFID Key Cards
                - Uses Radio Frequency Identification technology
                - RFID keycards contain chips with unique IDs, read by RFID readers to verify authorization
                - Common in office buildings, hotels, and parking garages
            - Biometric Scanners
                - Uses biological traits (fingerprints, facial recognition, iris scans) to verify identity
                - Provides a higher level of security, often used in high-security environments (e.g., data centers)
            - Keypad Entry Systems

            - Requires a user to enter a PIN to gain access
            - May be used in conjunction with other methods like RFID keycards or biometric scanners
        - Access Control Vestibule (Mantrap)
            - A setup with two sets of doors allowing only one person at a time into secure areas
            - Prevents tailgating (unauthorized following into secure areas)
            - Often used in high-security environments like government buildings and data centers
- Logical Access Control Systems
    - Definition
        - Systems that manage digital access to networks, ensuring only authorized users can access sensitive data or resources
    - Technologies Used
        - Passwords
            - Common form of logical access control but comes with limitations (e.g., weak passwords, password reuse)
            - Organizations promote using complex passwords and regular changes for better security
        - Multi-Factor Authentication (MFA)
            - Requires two or more authentication factors to verify identity
            - Authentication factors include
                - Something you know (password or PIN)

                ■ Something you have (smartphone or security token)
                ■ Something you are (biometrics like fingerprints)
            ○ MFA significantly strengthens security and reduces the risk of compromise
        ● Smart Cards
            ○ Physical cards with embedded chips containing encrypted user data
            ○ Often used with a PIN for an additional security layer
            ○ Common in corporate and government settings for securing access to networks or specific software applications
- Summary
    - Physical Access Control Systems
        - Manage entry to physical locations using technologies like RFID keycards, biometric scanners, keypad entry systems, and access control vestibules (mantraps)
    - Logical Access Control Systems
        - Manage access to digital networks or resources using methods like passwords, multi-factor authentication (MFA), and smart cards
    - Both types of systems verify identity through something a user knows, has, or is, ensuring security for both physical and digital assets

- **Access Provision**
    - Access Provision

        - The process of granting or revoking access rights to resources based on user roles and permissions
        - This ensures that individuals have the right level of access to perform their roles securely and efficiently
    - Credential Issuance
        - Definition
            - The process of creating and distributing authentication credentials (e.g., passwords, security tokens) to users after verifying their identity
        - Steps
            - Identity Proofing
                - Ensuring the user’s identity is verified, such as through ID documents or background checks
            - Credential Generation
                - Issuing credentials like usernames, passwords, or multi-factor authentication (MFA) tokens
        - Examples
            - Temporary passwords sent via email for users to change on their first login
            - Issuing security tokens or configuring mobile authentication apps
            - Biometric credentials (e.g., fingerprint scans) for authentication
    - Provisioning
        - Definition
            - The process of assigning access rights to users based on their roles
        - Role-Based Access Control (RBAC)
            - Automatically assigns access based on predefined user roles

        - Example
            - A marketing employee is given access to marketing tools, while a developer is provided access to coding environments
        - Purpose
            - Ensures users have the right access to perform their jobs
    - Self-Provisioning
        - Definition
            - Allows users to request access to systems without direct IT administrator involvement
        - Process
            - Users request access through a self-service portal based on their role or job function
            - Requests are reviewed and approved by administrators, or automatically approved based on predefined policies (e.g., RBAC)
        - Examples
            - New employees using a self-service portal to request access to HR systems or communication tools
        - Benefits
            - Speeds up onboarding, allowing users to initiate access requests while maintaining security checks
    - Deprovisioning
        - Definition
            - The process of removing access when it is no longer needed, such as when an employee leaves or changes roles
        - Purpose

        - Prevents unauthorized access after an employee's role changes or their employment ends
    - Automated Deprovisioning
        - Automatically revokes access to all systems tied to a user’s account when their role or employment status changes
    - Examples
        - Removing access to company databases and systems when an employee leaves the organization
- Summary
    - Access Provision
        - Involves managing user access to resources through credential issuance, provisioning, self-provisioning, and deprovisioning
    - Credential Issuance
        - Verifies the user's identity and provides authentication credentials (passwords, MFA tokens)
    - Provisioning
        - Assigns access rights to users based on their roles (e.g., via RBAC)
    - Self-Provisioning
        - Allows users to request access through a portal, with requests either automatically approved or reviewed by administrators
    - Deprovisioning
        - Revokes access when it's no longer needed, ensuring former employees or role-changed individuals can no longer access sensitive data or systems

- **Rule-Based Access Control**
  - Rule-Based Access Control
    - A method for managing user access to resources based on predefined rules and conditions set by administrators
    - These rules dictate access permissions and conditions under which access is allowed, providing flexibility and control
  - Mandatory Access Control (MAC)
    - Definition
      - Enforces access policies based on strict security classifications, restricting access to information based on the user's clearance level
    - Purpose
      - Provides high security by limiting access based on security classifications; only administrators can modify access permissions
    - Example
      - In a government setting, users with “secret” clearance can only access resources classified as “secret” or lower
    - Characteristics
      - Complex to configure, typically used in high-security environments
      - Supported by systems like SELinux, which incorporates mandatory access control in Linux for additional security
  - Discretionary Access Control (DAC)
    - Definition
      - Allows resource owners to control access to their own resources, granting or denying access at their discretion
    - Purpose

        - Provides flexibility by giving resource owners full control over permissions
    - Example
        - A user who creates a file can decide who can read, edit, or delete it
    - Characteristics
        - Default access control model in Windows and Linux systems
        - Offers flexibility but may be less secure if permissions aren't carefully managed
- Attribute-Based Access Control (ABAC)
    - Definition
        - Grants access based on a combination of user, resource, and environmental attributes, providing dynamic and adaptable control
    - Purpose
        - Ideal for complex environments requiring fine-grained, real-time access control
    - Example
        - An employee can access certain resources only if they are in the correct department, logged in during business hours, and connected through a secure network
    - Characteristics
        - Attributes may include user role, location, time of access, and resource sensitivity
        - Highly adaptable and flexible, adjusting access in real time based on multiple factors

- Summary
  - Rule-Based Access Control uses predefined rules to control resource access
  - Key models within this framework include
    - Mandatory Access Control (MAC)
      - Enforces strict access based on security classifications, with only administrators able to modify rules
    - Discretionary Access Control (DAC)
      - Allows resource owners to control access to their own files, providing flexibility but with potential security risks
    - Attribute-Based Access Control (ABAC)
      - Grants access based on multiple attributes, making it highly adaptable for complex environments
  - Together, these models allow organizations to implement security policies that match their access control needs, from strict classifications to flexible, attribute-based decisions

- **Role-Based Access Control (RBAC)**
  - Role-Based Access Control (RBAC)
    - A method of managing user access to resources by assigning permissions based on the user's role within an organization
    - Instead of giving individual permissions, roles are created to represent job functions, and users are assigned to those roles to inherit the associated permissions
  - Key Concepts
    - Role

        - Definition
            - Roles are created to represent job functions within an organization, and each role is assigned specific permissions
        - Example
            - The "HR" role is given access to employee records
            - The "Finance" role is given access to financial reports
    - Permissions
        - Definition
            - Permissions are linked to roles, determining what resources or systems users in that role can access
            - Permissions are typically assigned based on the resources a role needs to perform its duties
        - Examples
            - The "IT Support" role might have permissions to access system configurations or troubleshoot technical issues
    - User Assignment
        - Definition
            - Users are assigned to roles, and they automatically inherit the permissions associated with those roles
        - Example
            - When a new employee joins the HR department, they are added to the "HR" role, and automatically receive access to HR systems
        - Benefits
            - Easier management of access rights

            - Instead of individually assigning permissions, users can simply be added or removed from roles
        - Role-based updates
            - If a user's role changes or they leave the organization, adjusting their access only requires changing their role assignment
- Windows Enterprise Environment
    - RBAC in Windows
        - Role-Based Access Control is commonly implemented using Active Directory (AD) groups
    - Process
        - Create AD groups representing roles (e.g., "HR", "Finance")
        - Assign permissions to these groups for accessing specific resources (e.g., HR files, financial reports)
        - Add users to the relevant AD groups based on their job role
- Linux Enterprise Environment
    - RBAC in Linux
        - Role-Based Access Control is implemented using Linux user groups
    - Process
        - Create user groups representing roles (e.g., "admin", "developers")
        - Assign permissions for specific directories or files (e.g., the "developers" group has write access to source code directories)

        - Add users to the appropriate groups based on their job responsibilities
- Benefits of RBAC
    - Simplified Access Management
        - Administrators can manage permissions more efficiently by assigning users to roles rather than individually configuring each user’s permissions
    - Consistency
        - Ensures that users in similar roles have consistent access to the same resources
    - Easier Updates
        - When a user's responsibilities change (e.g., moving to a different department), their access can be easily adjusted by updating their role assignment
    - Scalability
        - RBAC is especially useful in larger organizations with complex systems and multiple users, as it simplifies the process of managing access rights
- Summary
    - RBAC manages access by assigning users to roles that define their permissions
    - Instead of configuring individual permissions for each user, administrators create roles based on job functions and assign users to these roles
    - This system is easier to manage, especially in large organizations, and ensures consistency in access control

        - In Windows, RBAC is often implemented through Active Directory groups, while in Linux, it's implemented using Linux user groups
        - By using roles, access can be easily modified when users change roles or leave the organization

- **Identity and Authentication**
    - Identity and Authentication
        - The process of verifying a user's identity to ensure they are authorized to access resources and perform actions
        - Key elements include Identity Providers (IdPs), Single Sign-On (SSO), Service Providers (SPs), Federation, and Attestation
    - Key Concepts
        - Identity Providers (IdPs)
            - Definition
                - Systems or organizations that verify a user's identity and provide authentication services for access to multiple resources
            - Example
                - Google acts as an Identity Provider when you log in to different services (e.g., YouTube, Gmail) using your Google credentials
        - Single Sign-On (SSO)
            - Definition
                - A method allowing users to authenticate once with an Identity Provider and then access multiple systems without logging in again

        - Example
            - Kerberos
                - In a corporate network, when a user logs in to their computer, they get a ticket (TGT) allowing access to other services (email, file servers) without needing to log in again
            - Shibboleth
                - An open-source system used to authenticate across multiple organizations, protecting user privacy by only verifying credentials without sharing user details with service providers
    - Service Providers (SPs)
        - Definition
            - Applications or systems that rely on the Identity Provider for authentication to grant user access
        - Example
            - Platforms like Salesforce or Dropbox allow users to log in using their Google credentials, trusting Google (the IdP) to verify their identity
        - Authentication Information
            - Often transferred using JWTs (JSON Web Tokens), which securely transmit user identity and authorization details in JSON format.
    - Federation
        - Definition

- An agreement between multiple organizations allowing users to access services across organizations using the same identity verification
- Example
    - eduGAIN network
        - Allows students from one university to access resources at another university using their home institution's credentials
- Transitive Trust
    - If Federation A trusts Federation B, and Federation B trusts Federation C, then Federation A automatically trusts Federation C, allowing seamless access across multiple federations

- Attestation
    - Definition
        - A message from an Identity Provider (IdP) to a Service Provider (SP) confirming that a user has been authenticated and specifying their access rights
    - Example
        - When you log in to a website using Google, Google sends an attestation to the website, verifying your identity and permissions
    - Protocols
        - Attestations are often sent using protocols like SAML (Security Assertion Markup Language), OpenID, or others, enabling secure transmission of identity information

- Benefits of Identity and Authentication Systems
  - Simplified User Experience
    - With SSO, users only need to authenticate once and can access multiple services without re-entering their credentials
  - Enhanced Security
    - Centralizing identity management through IdPs and Federation agreements reduces the risk of unauthorized access, as authentication is handled by trusted entities
  - Scalability and Collaboration
    - Federated systems allow multiple organizations to work together securely, enabling users to access shared resources across different domains without re-authenticating
  - Efficient Access Management
    - Service Providers can rely on attestations from Identity Providers, eliminating the need to manage authentication directly
- Summary
  - Identity and Authentication systems streamline access management by using Identity Providers (IdPs) to authenticate users and allowing them to access Service Providers (SPs) through Single Sign-On (SSO)
  - Federation extends this model by enabling multiple organizations to collaborate on identity management, while attestations provide proof of authentication between IdPs and SPs
  - These concepts simplify user authentication, enhance security, and allow for scalable, cross-organizational access to resources

- **Access Control Policies**
    - Access Control Policies
        - Rules and criteria for granting or denying access to resources, ensuring only authorized users can perform specific actions based on their identity, role, and additional conditions
    - Key Concepts
        - Conditional Access
            - Definition
                - Access is granted or denied based on specific conditions, such as user location, device type, or time of access, rather than just identity and role
            - Example
                - A company restricts access to sensitive data when users are not on the corporate network or when they are using an unmanaged personal device
                - Access might require additional steps like multi-factor authentication
            - Tool
                - Microsoft Azure Active Directory Conditional Access
                    - Controls access based on identity, location, and device type
        - Policy Decision Points (PDPs)
            - Definition
                - The system component responsible for evaluating whether an access request complies with the defined policies

        - It checks conditions like the user's role, location, and time of access before deciding whether to grant or deny access
      - Example
        - If a user attempts to access work files after business hours, the PDP checks the policy and denies access based on the time of the request
      - Tool
        - AWS Identity and Access Management (IAM)
          - Evaluates access requests based on policies and user conditions
    - Policy Enforcement Points (PEPs)
      - Definition
        - The system component that executes the decisions made by the Policy Decision Points
        - It either grants or denies access based on the decision from the PDP
      - Example
        - After the PDP checks and denies access based on policy, the PEP physically blocks the user from accessing the resource
      - Tool
        - Palo Alto Networks Next-Generation Firewall
          - Acts as a PEP, enforcing security policies by granting or blocking access based on PDP decisions
  - How Access Control Policies Work Together
    - Conditional Access

                - Evaluates extra criteria like user location or device type to add a dynamic layer to access decisions
            - Policy Decision Points (PDPs)
                - Analyze access requests based on identity, role, and conditions and decide whether the user should be granted or denied access
            - Policy Enforcement Points (PEPs)
                - Enforce the decision made by the PDP, allowing or blocking the user from accessing the resource based on policy
        - Summary
            - Access Control Policies
                - Define who can access what, based on a user's identity and role. Conditional access provides extra layers of security by considering additional factors like location and device type
                - Policy Decision Points (PDPs) decide whether to grant or deny access, and Policy Enforcement Points (PEPs) ensure these decisions are followed, acting as the gatekeepers to resources
                - Together, these elements enhance security by ensuring only authorized users access critical systems under the right conditions

- **Monitoring and Oversight**
    - Monitoring and Oversight
        - The continuous tracking and reviewing of access, authentication, and authorization activities to ensure compliance with security policies and detect unauthorized or anomalous behavior
    - Logging
        - Definition

                - Collecting detailed records of events like logins, access attempts, and system changes
                - Logs help track user activities and are useful for both troubleshooting and security investigations.
            - Centralized Logging
                - Logs from various systems are gathered in one place for easier management
                - Syslog
                    - A protocol for sending logs to a central server, assigning each log a PRI code based on
                        - Facility
                            - Represents the type of system generating the log (e.g., security, kernel, mail services)
                        - Severity
                            - Prioritizes the urgency of the event (0 = Emergency, 7 = Debugging)
                        - Example
                            - PRI code for Facility 4 and Severity 5 would be 37
            - Security Information and Event Management (SIEM)
                - Definition
                    - A platform that collects, categorizes, and analyzes logs from multiple sources in real-time, normalizing different log formats for better analysis
        - Auditing
            - Definition

        - Analyzing collected logs to assess compliance with regulations and identify potential security risks
    - Regulatory Compliance
        - Regular auditing ensures that organizations meet industry-specific standards (e.g., HIPAA in healthcare) by confirming only authorized personnel access sensitive information
    - Tamper-Proof Audit Trails
        - Definition
            - Logs that cannot be altered or deleted, ensuring integrity for forensic investigations and regulatory compliance
        - Technologies
            - Blockchain
                - Logs are stored in cryptographically linked blocks, preventing unauthorized modifications
            - WORM (Write Once, Read Many)
                - Data can be written to once but not modified or erased, ensuring permanent records
- How Logging and Auditing Work Together
    - Logging
        - Records events in a centralized system, allowing real-time tracking of user activity and system changes
        - Logs are typically stored in SIEM platforms, where they can be analyzed and correlated for patterns
    - Auditing
        - Reviews logs to ensure compliance with security policies and industry regulations

        - Auditors use tamper-proof trails to verify the integrity of data and check for policy violations or potential breaches
- Summary
    - Monitoring and Oversight rely on two key processes
        - logging and auditing
    - Logging collects and centralizes event records, making them accessible for tracking activities and troubleshooting
    - Auditing then analyzes these logs to ensure compliance with regulations and detect potential security threats
    - Technologies like SIEM platforms and tamper-proof systems (e.g., blockchain or WORM) provide organizations with the tools to monitor access activities securely, ensuring compliance and enabling effective incident response

Zero Trust Design

Objective 2.6: Integrate Zero Trust concepts into system architecture design

- **Security Boundaries**
  - Security Boundaries
    - These are defined by strict access controls and continuous verification rather than physical or network perimeters
    - They ensure that every cross-network interaction is monitored and validated
  - System Components
    - Definition
      - The individual elements making up an organization's IT infrastructure (e.g., servers, routers, databases, workstations, applications)
    - Zero Trust Model
      - Every device, application, and user must be verified at each stage, with no inherent trust
    - Example
      - A web server shouldn't have direct access to sensitive databases; each access must be strictly controlled using Role-Based Access Controls (RBAC), Multi-Factor Authentication (MFA), and network segmentation
    - Zone Segmentation

            - Dividing system components into zones based on their role (e.g., credit card processing, user authentication systems) allows specific security policies to be applied to each zone
    - Data Perimeters
        - Definition
            - Boundaries that protect sensitive data by controlling access through encryption, Access Control Lists (ACLs), and monitoring
        - Data Protection
            - Ensures that only authorized users can access data whether stored on-premises or in the cloud
        - Example
            - Sensitive data like customer or employee records are protected with encryption and Data Loss Prevention (DLP) tools to secure data both in transit and at rest
        - Continuous Validation
            - Traditional network perimeters are no longer enough, and ongoing validation of data access requests is required for protection, particularly with the shift to cloud environments
    - Secure Zones
        - Definition
            - Tightly controlled network areas designed to protect critical systems and data
        - Types of Secure Zones
            - Trusted Internal Zone
                - Contains sensitive internal systems
            - Untrusted External Zone

            - Typically the public-facing area (e.g., the internet)
        - Demilitarized Zone (DMZ)
            - Also known as a Screened Subnet, this is the buffer zone between the internal network and the external network
    - Access Control
        - Firewalls and Network Access Control (NAC) enforce strict access rules between zones
    - Examples
        - Bastion Hosts
            - Hardened servers placed outside secure zones to manage external access to critical systems
        - Jump Box
            - A gateway used to provide secure administrative access to systems in secure zones
        - Air-Gapped Networks
            - Completely isolated networks (no external connections), often used in highly sensitive environments like nuclear facilities
- Summary
    - Security Boundaries involve strict access controls, encryption, and continuous monitoring of every interaction within a network, rather than relying solely on physical or network perimeters
    - System Components
        - Each part of the IT infrastructure (servers, routers, etc.) must be secured individually, with different zones applying tailored security policies

        - Data Perimeters
            - Protect sensitive data through encryption, continuous access control, and monitoring, regardless of where the data is stored (on-premises or cloud)
        - Secure Zones
            - Critical systems are divided into controlled areas with limited and monitored access
            - Tools such as bastion hosts, jump boxes, and air-gapped networks help secure systems from both internal and external threats

- **VPN Architecture**
    - VPN Architecture
        - A network structure extending secure access controls beyond traditional boundaries, allowing users and sites to connect securely over the internet using encrypted tunnels
    - Client-Server VPN
        - Definition
            - Connects an individual client device to a central server through an encrypted tunnel
        - Purpose
            - Provides secure remote access for individual devices, ensuring encrypted data transmission between the client and the organization’s network
        - Example
            - A remote employee uses a client-server VPN to access their company's internal network securely from their laptop

        - Characteristics
            - Typically uses IPsec or TLS protocols for encryption
            - Authentication methods include username-password or digital certificates
            - Provides access to internal network resources like files, applications, and databases
    - Site-to-Site VPN
        - Definition
            - Connects two or more entire networks, such as the network of a company's headquarters and its branch offices
        - Purpose
            - Enables secure communication between multiple, geographically dispersed office networks as if they were part of the same local area network (LAN)
        - Example
            - A company's headquarters and branch offices communicate securely using a site-to-site VPN to share resources like file servers and internal services
        - Characteristics
            - Operates at the router level, connecting networks rather than individual devices
            - Commonly uses IPsec protocol for encryption and data authentication
            - Ideal for companies with multiple locations needing secure resource sharing
    - Always-On VPN

        - Definition
            - Provides a continuous, uninterrupted VPN connection between a device and the organization’s network
        - Purpose
            - Ensures that any traffic between the user’s device and the corporate network is always encrypted, regardless of location
        - Example
            - An employee’s device automatically establishes a VPN connection whenever it is on, maintaining secure access across different networks like home, office, or public Wi-Fi
        - Characteristics
            - Automatically connects when the device is turned on
            - Often integrates IPsec or TLS protocols with multi-factor authentication (MFA)
            - Includes network access control (NAC) to verify device security posture before granting access
    - Summary
        - VPN Architecture extends secure access beyond traditional network boundaries
        - Key VPN types include
            - Client-Server VPN
                - Secures individual remote connections to a central server, using encrypted tunnels and authentication protocols
            - Site-to-Site VPN

                - Connects entire networks over the internet, allowing secure communication and resource sharing between multiple offices
            - Always-On VPN
                - Maintains continuous encrypted connections, ensuring that security policies are consistently enforced across all networks
        - Each VPN type serves a specific purpose, from enabling remote access to connecting distributed networks securely, and ensuring continuous secure connections regardless of user location

- **Segmentation**
    - Segmentation and Microsegmentation
        - Segmentation
            - Dividing a network into distinct zones or segments to limit and control access between different areas
            - This enhances security by reducing the attack surface
        - Microsegmentation
            - A more granular approach to segmentation, where smaller zones are created within each larger segment, isolating workloads and securing them individually
    - Segmentation
        - Data Topology Mapping
            - Before segmenting a network, organizations map out their data zones by classifying data based on users, constraints, flow, and importance

        - This helps apply specific security policies based on the type of data (e.g., financial data, proprietary data)
    - Cloud Segmentation
        - Region-Based Segmentation
            - Cloud providers offer region-based segmentation, storing data in specific geographic locations for compliance with regional regulations
        - Availability Zones
            - Within cloud regions, availability zones provide redundancy and failover protection, ensuring high availability for critical services
        - Virtual Private Cloud (VPC) / Virtual Network (VNet)
            - Cloud providers like AWS and Azure allow organizations to create private networks using VPC or VNet, controlling traffic with subnets, security groups, and Network Access Control Lists (NACLs)
    - Internal Segmentation
        - Organizations use segmentation within environments such as production, staging, and guest environments, each having specific security needs
        - Production Environment
            - Live systems accessed by customers
        - Staging Environment
            - Used for testing before deploying to production
        - Guest Environment

            - Limited access for visitors without exposing internal resources
    - Peer-to-Peer Segmentation
        - Involves direct device communication, typically used in virtual local area networks (VLANs) or cloud-based environments
        - It is less secure and less common in enterprise environments
- Microsegmentation
    - Definition
        - Microsegmentation creates smaller zones within data centers and cloud environments, isolating workloads and applying specific security policies
    - Zero Trust Approach
        - No entity inside the network is trusted automatically
        - Each zone is isolated, and movement between zones is tightly controlled
    - Benefits
        - Isolates Breaches
            - If one part of the network is compromised, attackers are restricted to that segment and cannot access other systems or data
        - Granular Policy Control
            - Security policies are applied based on roles, application tags, and templates
        - Audit Trails
            - Full accountability and tracking of actions, making it easier to investigate security incidents

        - Example
            - If an attacker gains access to a web server, they are isolated within the microsegment and cannot reach the database or other critical systems
    - Summary
        - Segmentation divides a network into distinct zones to enhance security, reducing the risk of attacks
            - It allows administrators to apply specific protections for sensitive data and limit access between different environments (e.g., production, staging, guest)
        - Microsegmentation goes further by creating smaller, more detailed zones that isolate workloads and restrict traffic between them
            - This prevents attackers from moving freely across the network and increases control through detailed security policies

- **Deperimeterization**
    - Deperimeterization
        - Moving away from relying on traditional network boundaries and focusing on securing every user and device, regardless of their physical location within or outside the network
        - This shift ensures that all access to the network is verified and protected
    - Software Defined Networking (SDN)
        - Definition
            - SDN enables the management of network functions using software, rather than relying on traditional hardware-based methods

            - This enhances flexibility, scalability, and control within cloud environments
        - Layers
            - Control Plane
                - Manages decisions like traffic routing and network management
            - Data Plane
                - Handles the actual movement of traffic between devices
            - Management Plane
                - Provides administrative control and oversight of the network
        - Advantages
            - Increased agility and automation of network components
            - Easier to monitor network traffic and respond to threats
        - Challenges
            - If the SDN controller is compromised, it can bring down the entire network
        - Types
            - Open SDN
                - Using open standards to control network traffic
            - Hybrid SDN
                - Combines traditional and software-defined approaches
            - SDN Overlays
                - Provides virtualized network functions over existing infrastructure
    - Software-Defined Wide Area Network (SD-WAN)

        - Definition
            - SD-WAN builds on SDN by optimizing and securing wide-area network (WAN) connections, especially between geographically dispersed sites
        - Function
            - Connects branch offices and remote users to the central network using cheaper internet connections while ensuring performance and security
            - Routes traffic based on application priority, ensuring critical applications like video conferencing get better resources
        - Benefits
            - Centralized management of all network connections
            - Consistent enforcement of security policies across all locations
            - Ideal for remote and hybrid work environments
        - Supports Deperimeterization
            - Shifts focus from a single physical perimeter to securing all network connections across different locations
    - Secure Access Service Edge (SASE)
        - Definition
            - SASE combines networking and security functions into a single cloud-based service, ensuring secure access regardless of user location
        - Integration
            - Combines security tools like firewalls, secure web gateways, and zero trust network access into one service
        - Key Functions

        - Ensures continuous security for data moving between on-premises and cloud environments
        - Encrypts, monitors, and controls access to data across different environments (cloud, office, remote locations)
    - Benefits
        - Simplifies security management by handling everything through one service
        - Consistent enforcement of security policies and monitoring, regardless of user or device location
    - Supports Deperimeterization
        - Moves network security away from physical boundaries and extends security to wherever users or devices are, ensuring continuous protection
- Summary
    - Deperimeterization
        - Acknowledges that network boundaries are no longer tied to physical locations due to the rise of remote work and cloud services
    - Software Defined Networking (SDN)
        - Enables centralized control of network functions using software, enhancing flexibility and scalability
    - Software-Defined Wide Area Network (SD-WAN)
        - Optimizes and secures connections across different locations, ensuring efficient use of network resources and strong security standards
    - Secure Access Service Edge (SASE)

        - Integrates security and networking into one cloud-based service, allowing consistent security policies and continuous monitoring across diverse environments

- **Access Management**
    - Access Management
        - A security approach used to continuously verify and control user and device access to resources based on policies and contextual factors
        - Unlike traditional methods that assume trust based on network location, access management provides ongoing security by constantly assessing access privileges
    - Defining Subject-Object Relationships
        - Definition
            - Establishes clear boundaries for how subjects (users, devices, applications) can interact with objects (data, files, systems)
            - Access permissions are based on roles and rules defined within an access control system
        - Example
            - In a library system, readers (subjects) may only access general books (objects), while librarians (subjects with higher permissions) may access restricted collections
        - Purpose
            - Ensures that each subject has the appropriate access based on their role, maintaining security over sensitive resources
    - Continuous Authorization
        - Definition

        - Continuously evaluates and monitors user access throughout a session rather than checking access rights only once
        - If a user's behavior or actions deviate from the expected, permissions may be dynamically adjusted in real time
    - Example
        - If a finance team member tries to access human resources records, the system detects this suspicious behavior, restricts their access, and notifies security administrators
    - Purpose
        - Dynamically adapts access permissions based on real-time behavior, ensuring ongoing security and preventing unauthorized access during active sessions
- Context-Based Reauthentication
    - Definition
        - Requires users to reauthenticate based on changes in contextual factors such as location, device, or security posture
        - This ensures that access is still appropriate given the new conditions
    - Example
        - An employee working from a secure office network may be asked to re authenticate when switching to a public Wi-Fi network, as it poses higher security risks
    - Purpose
        - Adds an extra layer of security by verifying that the correct person is accessing resources when there are significant changes in the environment or context

- Summary
  - Access Management ensures secure, dynamic, and appropriate access by continuously verifying user permissions
  - Defining Subject-Object Relationships establishes rules for how users and devices interact with resources, ensuring that access is based on role and permissions
  - Continuous Authorization evaluates access throughout a session, adapting to changes in user behavior or context to prevent unauthorized actions
  - Context-Based Reauthentication adds security by requiring users to reauthenticate when factors like location or network security change, maintaining appropriate access levels

- **Application Programming Interface (API) Integration and Validation**
  - API Integration and Validation
    - API integration connects different applications and services, enabling secure data sharing and communication
    - API validation ensures that all interactions through APIs comply with security policies by verifying credentials, permissions, and data integrity
  - API Integration
    - Definition
      - Connects applications and services, allowing them to communicate and share data, saving time and resources by enabling different systems to work together
    - Example

        - An e-commerce platform uses an API to integrate with a payment gateway, allowing seamless processing of online transactions
    - Common Types of APIs
        - RESTful APIs
            - Lightweight, fast, and ideal for web services. Commonly used in modern web applications for high performance and scalability
        - SOAP APIs
            - Structured and reliable, often used in industries requiring strict data validation and security, such as financial services and telecommunications
    - Middleware/Integration Platforms
        - Manage data flow, handle data format changes, manage errors, and ensure secure transfers during API integration
- API Validation
    - Definition
        - The process of ensuring that all API interactions are secure, including checking credentials, permissions, and data formats
    - Authentication Protocols
        - OAuth
            - An authorization framework that enables third-party applications to access user information securely
        - OpenID Connect
            - An identity layer built on OAuth 2.0 for verifying user identities
        - API Keys

            - Unique tokens that authenticate and authorize API requests
        - JSON Web Tokens (JWT)
            - A token format used to securely transmit information between parties
    - Input Validation
        - Ensures that data sent through APIs follows correct formats and business rules, preventing malicious activity such as SQL injection
    - Example
        - An API for updating inventory levels validates that only authorized users can modify data and ensures correct input formats (e.g., valid product IDs and quantities)
- Summary
    - API Integration
        - Enables different systems to share data and automate processes without creating new software
        - It uses standardized protocols like RESTful APIs for high-performance applications and SOAP APIs for secure, reliable transactions
    - API Validation
        - Ensures secure API communication by checking credentials, permissions, and data formats, using authentication methods like OAuth and API keys
        - Input validation prevents security risks such as SQL injection or data corruption

- **Asset Control**
  - Asset Control
    - Involves maintaining a detailed inventory of all network-connected assets (devices, applications, data) and continuously validating their security status to ensure they comply with security policies and are protected from threats
  - Asset Identification
    - Definition
      - The process of cataloging all devices, applications, and data within a network to create a comprehensive inventory
    - Example
      - Identifying every computer, mobile device, server, and application in a corporate network
    - Tools
      - Nmap or Nessus
        - For network scanning to identify connected devices
      - Microsoft System Center Configuration Manager
        - For tracking installed software
      - AWS Asset Manager / Azure Resource Manager
        - For cataloging cloud-based resources
      - CrowdStrike / SentinelOne
        - For continuous endpoint tracking and monitoring
    - Purpose
      - Provides an up-to-date picture of all assets to ensure network security and compliance
  - Asset Management

        - Definition
            - Ongoing monitoring and maintenance of identified assets to ensure they are secure and compliant with security policies
        - Example
            - Applying the latest security updates, ensuring correct configurations, and monitoring compliance
        - Tools
            - Microsoft WSUS
                - For patch management. Ansible, Puppet, Chef: For configuration management
            - Microsoft Intune / VMware Workspace ONE
                - For monitoring and remediating endpoint compliance
            - ServiceNow / SolarWinds
                - For tracking asset status and security compliance
        - Purpose
            - Ensures assets remain secure, updated, and in line with company policies
    - Asset Attestation
        - Definition
            - Verifying the security posture and compliance of assets to ensure they meet required security standards
        - Example
            - Regular checks to ensure devices have up-to-date antivirus software or specific configurations
        - Tools
            - CrowdStrike / SentinelOne

        - For continuous monitoring of asset compliance
      - Splunk / IBM QRadar (SIEM platforms)
        - For analyzing security data and providing insights into compliance failures
      - Tenable / Qualys
        - For vulnerability scanning and compliance checks
    - Purpose
      - Ensures that only secure and compliant assets can operate within the network, preventing unauthorized access or risky configurations
  - Summary
    - Asset Identification catalogs devices, applications, and data using tools like Nmap and cloud management platforms to maintain a real-time inventory
    - Asset Management involves monitoring, patching, and configuring assets to ensure they comply with security policies and are updated regularly
    - Asset Attestation verifies compliance and security status using continuous assessments and tools like SIEM and vulnerability scanning platforms

Hardware Security

Objective 3.4: Implement hardware security technologies and techniques

- **Roots of Trust**
    - Roots of Trust
        - Foundational security components in a computing system responsible for managing cryptographic keys, ensuring system integrity, and enabling secure boot processes
    - Trusted Platform Modules (TPMs)
        - Definition
            - A physical chip installed on a motherboard that acts as a hardware Root of Trust, responsible for storing cryptographic keys, digital certificates, and performing cryptographic operations
        - Functions
            - Ensures secure boot by validating BIOS and critical components during system startup
            - Manages cryptographic keys using RSA, AES, and other algorithms
            - Provides a true random number generator for secure key generation
            - Persistent memory for critical keys, such as the Endorsement Key (EK) and Storage Root Key (SRK)
        - Use Case
            - A server uses TPM to secure its boot process and works with full disk encryption solutions like BitLocker to keep data safe

        - For Exam
            - Remember TPM’s role as a hardware Root of Trust for ensuring secure boot and system integrity
    - Virtual Trusted Platform Modules (vTPMs)
        - Definition
            - A software-based implementation that extends TPM functionality to virtual machines by providing a virtual Root of Trust
        - Functions
            - Secures virtual machine boot processes and ensures system integrity
            - Manages cryptographic keys and performs encryption, decryption, and signature verification in virtual environments.
        - Management
            - Managed through platforms like VMware or Microsoft Hyper-V
        - Use Case
            - Virtual machines on a physical server use vTPM to secure their own boot processes and manage cryptographic operations
        - For Exam
            - Focus on the fact that vTPMs provide similar functionality as TPMs, but for virtualized environments
    - Hardware Security Modules (HSMs)
        - Definition
            - Standalone devices or specialized hardware designed to manage and protect cryptographic keys across multiple machines
        - Functions

        - Generates and protects cryptographic keys (RSA, AES, Elliptic Curve)
        - Ensures centralized, secure key management and offloads encryption functions from application servers
        - Tamper-evident and protects against insider threats
    - Use Case
        - Used in enterprises for securing master encryption keys, Public Key Infrastructure (PKI), and DNSSEC key management
    - For Exam
        - Remember that HSMs are dedicated hardware devices providing robust cryptographic security across multiple systems, unlike TPMs which are embedded on individual motherboards
- Summary
    - TPMs are physical chips embedded in systems to manage cryptographic keys and validate system boot processes
    - vTPMs extend TPM functionality into virtual environments, securing virtual machine operations and cryptographic management
    - HSMs are standalone, high-security hardware devices used for key management across multiple systems, providing enterprise-level cryptographic protection

- **Boot Options**
    - Boot Options
        - Methods and sequences that determine how a system starts and verifies its integrity before the operating system loads

- The key boot options are Secure Boot and Measured Boot, which help ensure the security and integrity of a system during startup

- Secure Boot
  - Definition
    - A UEFI security feature that ensures only trusted, digitally signed components are loaded during the boot process
  - How It Works
    - Secure Boot checks that each boot component, such as UEFI executable files and operating system loaders, has a valid digital signature
    - Verifies the integrity of boot-critical drivers against their known good hashes
    - Blocks any components that fail the validation process, preventing the execution of untrusted software or malicious code
  - Use Case
    - Secure Boot is often used in systems running Windows to ensure that boot components and drivers have not been tampered with
  - For Exam
    - Focus on understanding that Secure Boot ensures that only trusted components are loaded during the boot process
    - It is a feature of UEFI (not BIOS) and only works with operating systems that support it, like Windows
- Measured Boot
  - Definition

        - A boot process that records cryptographic hashes of each boot component into a Trusted Platform Module (TPM) to create a verifiable log of the boot sequence
    - How It Works
        - Hashes of key system components (UEFI firmware, boot loader, kernel, and drivers) are measured and stored in the TPM during startup
        - These logs provide visibility into what was loaded during the boot process, even if the boot is not halted
        - Measured Boot does not block components but creates a detailed log of the boot process, which security teams can use to verify system integrity and detect tampering or malware
    - Use Case
        - Organizations use Measured Boot to maintain visibility into the boot sequence and detect any unauthorized changes in the system
    - For Exam
        - Understand that Measured Boot complements Secure Boot by providing a record of the boot process for further validation
        - It logs events without preventing booting, allowing security teams to analyze boot integrity afterward
- Summary
    - Secure Boot
        - Focuses on blocking untrusted software by ensuring each boot component has a valid digital signature

            - It actively prevents tampered or unauthorized code from running during the boot process
        - Measured Boot
            - Records cryptographic hashes of each boot component into a TPM, providing a detailed log of the boot sequence that can be used to detect unauthorized changes after the system has started.

- **Security Coprocessors**
    - Security Coprocessors
        - Specialized hardware components designed to perform cryptographic operations and store sensitive data within a protected environment, enhancing overall system security
    - CPU Security Extensions
        - Definition
            - Hardware-based technologies integrated into the CPU that create secure execution environments for critical data and processes.
        - How It Works
            - Isolates sensitive operations from the rest of the system to protect them from unauthorized access and malware
        - Example technologies
            - Intel Trusted Execution Technology (TXT)
                - Verifies the integrity of the system environment before executing sensitive applications
            - AMD Secure Encrypted Virtualization (SEV)
                - Encrypts virtual machine data, ensuring that even if the hypervisor is compromised, the VMs remain protected

        - Use Case
            - Protects against memory corruption, unauthorized data access, and breaches in virtual machine isolation
        - For Exam
            - Focus on understanding how CPU Security Extensions isolate critical processes and protect them from external threats, creating a trusted execution environment within the CPU
    - Secure Enclaves
        - Definition
            - A type of secure execution environment specific to Apple devices, providing isolated, independent areas for sensitive operations
        - How It Works
            - Dedicated processors and memory operate separately from the main CPU, securing sensitive data like encryption keys and biometric information
            - Protects critical data even if the main operating system is compromised
        - Use Case
            - Used for handling sensitive operations such as cryptographic key storage, biometric data processing (e.g., Face ID, Touch ID), and secure transactions
        - For Exam
            - Focus on how Secure Enclaves create a completely isolated environment for sensitive data on Apple devices, securing key operations from tampering or unauthorized access
    - Summary

        - Security Coprocessors are crucial for securely handling cryptographic functions and sensitive data
        - CPU Security Extensions
            - Isolate sensitive operations within the CPU
            - Protect critical processes from attacks
            - Examples
                - Intel TXT and AMD SEV
        - Secure Enclaves
            - Specific to Apple devices
            - Provide a dedicated, isolated area for storing encryption keys, biometric data, and secure transactions
            - Operates independently from the main CPU for additional security

- **Self-Encrypting Drives (SED)**
    - Self-Encrypting Drives (SEDs)
        - Storage devices, such as hard disk drives (HDDs) or solid-state drives (SSDs), with built-in encryption capabilities that automatically secure data as it is written to or read from the drive
        - SEDs perform encryption and decryption transparently without user input or additional software
    - How SEDs Work
        - Onboard Encryption
            - SEDs have an integrated encryption engine that handles encryption and decryption of data on-the-fly as it is written to and read from the drive
        - Transparency

        - Users do not need to interact with the encryption process, and it happens seamlessly without affecting the drive's performance
    - Encryption Key Management
        - Encryption keys are securely stored within the hardware of the drive, ensuring they are not exposed to the operating system or software, minimizing the risk of compromise
- Advantages of SEDs
    - Ease of Use
        - SEDs require no additional software or complex configuration, as encryption happens automatically when data is written to the drive
    - Consistent Performance
        - Since the encryption is handled by the drive's hardware and not the CPU, there is no performance slowdown typically associated with software-based encryption
    - Strong Security
        - Encryption keys are securely stored within the drive's hardware and never exposed to the operating system, protecting against data breaches
    - Simple Deployment
        - No need for encryption software or manual management of encryption keys, which reduces setup and maintenance efforts
- Drawbacks of SEDs
    - Cost

                - While consumer-grade SEDs are affordable, enterprise-grade models that meet strict security standards (e.g., FIPS 140-2, IEEE 1667) can be significantly more expensive
            - Lack of Upgradability
                - As SEDs are hardware-based, they may require complete replacement if encryption standards evolve, since updating encryption algorithms on the drive itself is difficult
            - Complex Security Standards
                - Compliance with high-security standards is essential in regulated environments, which can increase the overall cost and make selecting the right SED more complex
    - Summary
        - Self-encrypting drives (SEDs) provide a secure, hardware-based solution for automatically encrypting and decrypting data on storage devices
        - They do not require user interaction or additional software, making them easy to use
        - By using onboard encryption engines, SEDs maintain consistent performance without overburdening the CPU, offering strong protection against data breaches by securely storing encryption keys within the hardware
        - However, SEDs can be more expensive, especially for enterprise models that meet stringent security standards, and may be difficult to update if new encryption standards are required

- **Host-Based Encryption**
    - Host-based Encryption

        - Encryption managed by the local operating system to protect data at rest on a storage device, ensuring that data remains secure even when the device is not in use or the operating system is not running
    - Windows: BitLocker
        - Definition
            - A built-in Windows tool that provides full disk encryption using the Advanced Encryption Standard (AES)
        - Purpose
            - Secures data on Windows systems, making it inaccessible without proper credentials, especially if the device is stolen or accessed without permission
        - Characteristics
            - Uses AES encryption to protect entire disks
            - Often relies on the Trusted Platform Module (TPM) for secure encryption key management
        - Example
            - Encrypting a laptop's hard drive with BitLocker protects sensitive data, so if the device is lost, data remains unreadable without the user's credentials
    - Linux: Cryptsetup with Linux Unified Key Setup (LUKS)
        - Definition
            - A Linux tool for full disk encryption that uses the AES standard to secure data
        - Purpose

        - Protects data at rest on Linux systems by ensuring data is encrypted before being stored and decrypted only when accessed with the correct passphrase or key
    - Characteristics
        - Manages encryption keys and secures data with AES
        - Supports a wide range of Linux environments, including desktops, servers, and portable devices
    - Example
        - A Linux server running Cryptsetup with LUKS can be secured so that data is only accessible when the correct passphrase is provided, adding a layer of security for sensitive server data
- macOS: FileVault
    - Definition
        - A macOS tool for full disk encryption, integrated directly into Apple’s operating system
    - Purpose
        - Secures data on macOS systems, preventing unauthorized access when the device is powered off or left unattended
    - Characteristics
        - Uses AES encryption to protect Mac devices
        - Provides seamless integration into macOS for easy management
    - Example
        - FileVault encrypts a MacBook’s disk, so if it is lost or stolen, data remains secure and accessible only by the authorized user
- Summary

        - Host-based encryption secures data at rest on devices managed by the operating system, providing robust protection against unauthorized access in case of device loss or theft
        - Key tools include
            - BitLocker
                - A Windows tool that encrypts entire drives, using AES and often TPM to manage keys securely
            - Cryptsetup with LUKS
                - A Linux tool for full disk encryption, securing data using AES and managed through LUKS to control encryption keys
            - FileVault
                - A macOS tool that integrates into Apple's operating system to encrypt data on Mac devices
        - Each tool offers strong security for data stored on different operating systems, making host-based encryption essential for organizations and users to protect sensitive information across Windows, Linux, and macOS environments

- **Self-Healing Hardware**
    - Self-Healing Hardware
        - Technology that automatically detects, corrects, and recovers from faults or damage without manual intervention, increasing system reliability, efficiency, durability, and sustainability
    - Efficiency
        - Quick Fault Detection

        - The system can rapidly identify issues and initiate repair processes automatically
    - Minimizing Downtime
        - By addressing hardware faults immediately, self-healing hardware reduces operational disruptions
    - Example
        - In a RAID system, when a disk drive fails, the system detects the failure and rebuilds the data on a new drive automatically, ensuring minimal downtime and smooth operation
- Durability
    - Recovery from Failures
        - Self-healing hardware can recover from both physical and logical failures, making the system more resilient
    - Enhanced Reliability
        - The ability to reroute electrical pathways or repair damaged components extends hardware life and reduces the likelihood of total system failure
    - Example
        - Self-healing circuits can reroute power through alternate pathways when a component fails, maintaining system functionality
- Sustainability
    - Extended Lifecycle
        - Self-healing technologies lengthen the lifespan of hardware, reducing the need for frequent replacements
    - Reducing Waste

        - By repairing rather than replacing damaged hardware, self-healing systems contribute to more environmentally friendly practices
    - Example
        - Self-healing polymers in devices repair scratches or cracks, keeping the device in better condition for a longer time
- Innovation
    - Advanced Diagnostics
        - Smart sensors, AI, and machine learning allow systems to predict hardware failures before they occur
    - Preventative Actions
        - Self-healing hardware can take action to prevent damage, further reducing the need for manual intervention
    - Example
        - Smart systems that use machine learning to monitor component health and preemptively address issues before they lead to system failures
- Summary
    - Self-healing hardware refers to systems that automatically detect and fix their own faults, ensuring higher reliability and minimizing downtime
    - These systems excel in efficiency by addressing issues rapidly without user intervention
    - Durability is improved as the hardware can recover from failures and prevent complete system breakdowns
    - Sustainability is enhanced by extending the lifespan of devices, reducing the environmental impact associated with frequent hardware replacement

        - Finally, innovation in self-healing hardware introduces advanced diagnostic and predictive capabilities, driven by AI and smart technology, to further optimize system performance and resilience

- **Virtual Hardware**
    - Virtual Hardware
        - Simulated hardware created and managed by hypervisor software that enables running multiple operating systems and applications on a single physical server by dynamically allocating resources like CPU, memory, and storage
    - Scalability
        - Multiple Virtual Machines
            - Virtual hardware allows multiple virtual machines (VMs) to run on a single physical server, sharing resources as needed
        - Dynamic Resource Allocation
            - CPU, memory, and storage are allocated based on the demands of each virtual machine, allowing businesses to scale up or down as needed
        - Example
            - During peak seasons, a business can allocate more resources to handle higher demand and scale down during quieter periods without needing additional physical hardware
    - Efficiency
        - Optimized Resource Usage

        - Virtual hardware makes sure that all computing resources are used efficiently, minimizing waste by sharing the physical server's resources
    - Quick Deployment
        - New virtual environments can be created in minutes, allowing for faster setup of applications and services compared to physically installing new hardware
    - Example
        - A company can deploy a new virtual machine quickly to respond to business changes, such as launching a new service, without having to purchase or configure physical hardware
- Cost Effectiveness
    - Reduced Hardware Investment
        - Virtual hardware allows for the consolidation of multiple virtual machines on fewer physical servers, lowering upfront hardware costs
    - Lower Operational Costs
        - Fewer physical servers mean reduced expenses for power, cooling, and maintenance, making virtualized infrastructure more economical
    - Example
        - A company that previously needed ten physical servers might reduce this to two or three, saving significantly on capital expenses and operational costs
- Summary

- Virtual Hardware simulates physical hardware components, enabling multiple virtual machines to run on the same physical server
- This virtual environment is managed by a hypervisor, which allocates resources like CPU, memory, and storage as needed
- Key advantages include
  - Scalability
    - Easily adjust resources to meet changing demands without the need for additional physical hardware
  - Efficiency
    - Optimize resource usage and deploy new virtual machines quickly, improving overall IT responsiveness
  - Cost Effectiveness
    - Lower hardware investments and operational costs by consolidating multiple virtual machines onto fewer physical server

Endpoint and Server Security

Objective 3.2: Analyze requirements to enhance the security of endpoints and servers

- **Configuration and Privilege Management**
    - Configuration and Privilege Management
        - Processes and tools used to manage system settings and user permissions across enterprise devices and systems to ensure secure, consistent, and compliant operation
    - Configuration Management
        - Purpose
            - Ensures that systems remain secure, consistent, and up-to-date across the organization by controlling software versions, system settings, and updates
        - Tools
            - Microsoft System Center Configuration Manager (SCCM)
                - Automates software updates, patches, and configuration settings on Windows-based devices to ensure consistency and security
            - Ansible
                - An open-source platform that automates configuration tasks across various systems (Linux, Windows, etc.) using human-readable playbooks
                - Ansible operates declaratively, ensuring systems reach a desired state

        - Example
            - Ansible can be configured to ensure a software package is installed and running on all systems by specifying this desired state in the playbook
            - Ansible automatically takes the necessary steps to meet this requirement
    - Endpoint Privilege Management
        - Purpose
            - Reduces security risks by restricting user permissions and access levels on individual endpoints, enforcing the principle of least privilege to prevent unauthorized actions
        - Tools
            - Linux (Sudo)
                - In Linux environments, the sudo command allows specific administrative tasks to be performed without giving users full admin rights
                - Permissions are defined in the /etc/sudoers file
            - Microsoft Local Administrator Password Solution (LAPS)
                - In Windows environments, LAPS manages local admin account passwords by automatically generating complex, unique passwords for each device and securely storing them in Active Directory
                - LAPS also provides audit trails for enhanced security
        - Example

            - A Linux user can execute specific administrative tasks (such as restarting services) using sudo, without having full root access, ensuring minimal privilege is granted
    - Summary
        - Configuration Management
            - Involves controlling and maintaining system settings, software versions, and updates across an organization
            - It ensures that devices are consistently configured and secure, using tools like SCCM and Ansible to automate and simplify management tasks
        - Endpoint Privilege Management
            - Focuses on controlling user permissions at the device level
            - It reduces risks by granting users only the permissions they need through tools like sudo for Linux and LAPS for Windows
            - These tools enforce the principle of least privilege and provide secure password management and audit trails

- **Operating System Security**
    - Operating System Security
        - Mechanisms within an operating system to protect it from unauthorized access, malware, and other security threats
        - SELinux is a key component of OS security in Linux systems, enforcing strict security policies at the kernel level
    - Security-Enhanced Linux (SELinux)
        - Definition

        - A security module integrated into the Linux kernel that provides a framework for enforcing strict access control policies across the operating system
    - Purpose
        - Prevents unauthorized access to files, processes, and resources by checking each access attempt against predefined security policies
    - Example
        - When a web server tries to access a configuration file, SELinux verifies the security context of both the process and file
        - If the action is not explicitly allowed by the policy, SELinux denies it, protecting the system from potential unauthorized access
- Mandatory Access Control (MAC)
    - Definition
        - A security model that enforces strict rules on access rights, applying policies that specify permissible actions for users, applications, and processes
    - Purpose
        - Ensures access is controlled by security policies rather than traditional user permissions, adding a layer of control to protect against unauthorized actions
    - Characteristics
        - Uses labels and tags in the security context (user, role, type, and level) to define interactions between system resources
        - Enforces rules based on these attributes to prevent unauthorized interactions
- Security Contexts

        - Definition
            - Sets of security attributes assigned to files, processes, and other system resources that define access permissions under SELinux policies
        - Purpose
            - Governs interactions based on labels and tags, allowing or denying actions based on the security context rather than traditional permissions
        - Example
            - A database server might have policies restricting it from accessing files outside its data directory, blocking any unauthorized actions even if a vulnerability is exploited
    - Tools for Managing SELinux Policies
        - setsebool
            - Enables or disables Boolean settings within SELinux, allowing administrators to toggle specific security policies as needed
        - semanage
            - Manages and modifies SELinux policies, such as adding rules to permit applications access to designated directories
        - audit2allow
            - Generates custom SELinux policies by analyzing logs of denied actions, enabling administrators to create rules for legitimate activities previously blocked
    - Summary

    - Operating System Security measures, such as SELinux, protect Linux systems from unauthorized actions by enforcing strict access control policies
    - SELinux operates at the kernel level, controlling access through Mandatory Access Control (MAC) policies based on security contexts that define user, role, type, and level labels
    - Key tools like setsebool, semanage, and audit2allow assist administrators in managing and customizing SELinux policies, ensuring that only authorized actions are permitted and enhancing overall system security

- **Threat Protection**
  - Threat Protection
    - The implementation of security measures to detect, prevent, and respond to threats and attacks on systems and networks
  - Antimalware
    - Purpose
      - Detect, prevent, and remove malicious software, including viruses, worms, ransomware, adware, and spyware
    - Methods
      - Signature Detection
        - Identifies known malware by comparing files to a database of known signatures
      - Behavior Analysis and Heuristics
        - Detects unusual or malicious activity based on behavior patterns, even for new or unknown malware
    - Tools

            - Microsoft Defender Antivirus
                - Combines antivirus and antispyware functionalities, offering real-time protection against malware
        - Email and Browser Protection
            - Disabling automatic email previews and using spam filters reduce the risk of phishing and malware
            - Browser extensions warn users of suspicious websites, adding an extra layer of protection
    - Host-based Firewalls
        - Purpose
            - Installed directly on individual devices (endpoints) to control inbound and outbound network traffic
        - Function
            - Access Control Lists (ACLs)
                - Define which traffic should be allowed or blocked based on ports, applications, or IP addresses
            - Example
                - Microsoft Windows Firewall blocks unauthorized traffic based on pre-set rules
        - Configuration
            - Block IP ranges such as 10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16 to prevent spoofing attacks
        - Linux Firewalls
            - IP Tables
                - A powerful tool in Linux for defining firewall rules to manage and filter traffic at the system level

- Host-based Intrusion Detection Systems (HIDS)
    - Purpose
        - Installed directly on endpoints to detect unauthorized activities by monitoring log files, system configurations, and application behavior
    - Function
        - Detects suspicious behavior like unusual file changes, unauthorized access attempts, or unexpected processes
        - Alerts and Logs
            - HIDS generates alerts and logs details for investigation by administrators
    - Use Case
        - Provides real-time visibility into potential attacks that might not be detected by network-based security tools
- Host-based Intrusion Prevention Systems (HIPS)
    - Purpose
        - Detects and blocks attacks before they cause harm, going beyond detection to actively prevent threats
    - Detection Methods
        - Signature-based Detection
            - Identifies known attack patterns
        - Anomaly-based Detection
            - Identifies deviations from normal system behavior
    - Example

            - If HIPS detects an attempt to modify critical system settings or run a malicious process, it immediately blocks the activity and alerts the user
        - Centralized Management
            - HIPS can be integrated with a centralized system to provide a unified view of security events across all endpoints in an organization
    - Summary
        - Threat protection includes key security measures such as Antimalware, Host-based Firewalls, Host-based Intrusion Detection Systems (HIDS), and Host-based Intrusion Prevention Systems (HIPS)
        - Anti Malware detects and removes malicious software, using both signature-based and behavior-based detection
        - Host-based Firewalls control network traffic at the device level by filtering based on ports, applications, and IP addresses
        - HIDS monitors system activities for suspicious behavior and logs alerts for investigation
        - HIPS proactively blocks attacks, offering a more active defense against system compromises by using both signature and anomaly-based detection

- **Application Management**
    - Application Management
        - The process of overseeing and controlling applications to prevent vulnerabilities from being introduced into the network and ensure the integrity of enterprise systems

- Application Control
    - Purpose
        - To control which applications are allowed to run on a system and prevent unauthorized or malicious software from compromising network security
    - Methods
        - Allow Lists and Block Lists
            - Policies that define which applications are permitted (allow lists) and which are prohibited (block lists)
        - Windows Group Policy
            - An example of a tool that can enforce these policies, pushing rules to all systems in the domain to ensure compliance
        - Host-based Intrusion Prevention Systems (HIPS)
            - These can prevent the execution of unauthorized software, adding an extra layer of security by blocking potentially harmful software even before it runs
    - Benefits
        - Reduces the risk of attacks by controlling what software is installed or executed
        - Helps maintain a secure and consistent application baseline across an organization
    - Example
        - Blocking games or unapproved software from running on company workstations to ensure security and productivity
- Browser Isolation

- Purpose
  - To isolate web browsing activities from the rest of the operating system, protecting the system from web-based threats such as malware or drive-by downloads
- Methods
  - Virtual Machines/Remote Servers
    - Web browsing can be isolated in a virtual environment, ensuring that malicious content is contained within a separate area and cannot affect the host system
  - Cloud-based or On-premises Solutions
    - Browser isolation can be implemented as a cloud service or integrated into an organization's existing infrastructure
  - Secure Web Gateways or Firewalls
    - These can be used to route high-risk web traffic through isolated environments
- Benefits
  - Prevents harmful web content from spreading to the main operating system or network
  - Can be customized with policies that specify when and for whom isolation should be applied
- Example
  - Isolating browsing activities when users visit high-risk websites, ensuring that malicious scripts or downloads are contained within a secure, virtual browser environment

○ Summary

        - Application Management is essential for ensuring that applications do not introduce security risks or compromise system integrity
        - Application Control uses allow and block lists to restrict which applications can be installed or run, preventing unauthorized software from being executed and ensuring that only approved software is in use
        - Browser Isolation creates a secure, isolated environment for web browsing, preventing harmful web content from reaching the host system and containing potential threats in a protected space
        - Together, these measures strengthen system security, minimize risks, and ensure that enterprise systems are safeguarded from unauthorized software and web-based attacks

- **Monitoring and Response**
    - Monitoring and Response
        - The continuous process of observing system activities to detect, analyze, and respond to threats
    - Event Logging and Monitoring
        - Purpose
            - To capture and analyze system events, application activities, user actions, and network behaviors to detect abnormal behavior, indicators of attacks, and help with incident response
        - Challenges
            - Balancing Logging
                - Logging too much can degrade system performance, while logging too little can result in missing critical security information

        - Audit Log Management
            - Defines what should be logged, how long logs should be kept, and how logs are backed up
            - Includes controls like two-person approval for log modifications to ensure log integrity
        - SIEM (Security Information and Event Management)
            - Purpose
                - Centralizes and automates log collection, analysis, and alerting
            - Functions
                - Data Aggregation
                    - Collects logs from multiple sources, like servers, firewalls, and applications
                - Data Correlation
                    - Analyzes relationships between logs to understand broader attack patterns
            - Examples
                - ArcSight, AlienVault, QRadar
    - Endpoint Detection and Response (EDR)
        - Purpose
            - Provides real-time visibility, threat detection, and response capabilities at the endpoint level (e.g., desktops, laptops, servers)
        - Features
            - Continuous Monitoring
                - Tracks processes, file access, network activity, and changes to system configurations

        - Threat Isolation
            - Immediately isolates compromised endpoints to prevent threats from spreading across the network
        - Machine Learning
            - Uses AI to detect new and unknown threats by identifying anomalies in behavior
        - Historical Data
            - Provides detailed logs for forensic investigation to track how the attack began, spread, and which files or systems were impacted
    - Benefits
        - Proactive Detection
            - Detects both known and unknown threats, going beyond traditional anti-malware solutions
        - Rapid Response
            - Quickly isolates compromised systems and helps security teams address the threat without affecting the broader network
        - Forensic Capabilities
            - Offers a clear view of attack vectors and system impact, assisting in thorough remediation efforts
- Summary
    - Monitoring and Response is crucial for detecting, analyzing, and responding to threats within an enterprise network
    - Event Logging and Monitoring

            - Involves capturing and reviewing system logs to identify abnormal activities. SIEM systems are essential for centralizing, aggregating, and correlating these logs to provide real-time insights and notifications
        - Endpoint Detection and Response (EDR)
            - Focuses on protecting individual endpoints by continuously monitoring activity, detecting threats, and responding to incidents by isolating affected devices and providing detailed forensic data
        - Together, these concepts strengthen an organization's ability to detect, respond to, and prevent security incidents across their network and endpoints

- **Mobile Management**
    - Mobile Management
        - A set of tools and strategies that help secure and manage mobile devices within an organization, typically through Mobile Device Management (MDM), which is part of Enterprise Mobility Management (EMM)
    - Enterprise Mobility Management (EMM)
        - Purpose
            - Comprehensive approach to managing and securing mobile devices by combining policies and technical tools
        - Components
            - Mobile Device Management (MDM) is a key technical aspect of EMM, providing centralized control and enforcement of security policies on mobile devices
    - Mobile Device Management (MDM)

- Purpose
    - Centralized control over mobile devices, enforcing security policies to ensure compliance and protect sensitive data
- Key Features
    - Application Control
        - Manages app installations and configurations, blocking unapproved apps that could introduce risks
    - Password Management
        - Enforces strong passwords and supports biometric authentication, ensuring secure access
    - Multi-Factor Authentication (MFA)
        - Adds an extra layer of security by requiring multiple forms of authentication, especially when devices are in untrusted locations
    - Token-based Access
        - Uses digital certificates to authenticate devices and ensure only trusted devices can access the network
    - Patch Management
        - Ensures that devices are updated with the latest security patches and operating system updates, preventing vulnerabilities
    - Remote Wipe
        - Allows for the remote erasure of lost or stolen devices to protect sensitive data
    - Geofencing

                - Creates virtual boundaries to apply specific security policies when a device enters or leaves defined geographic areas
            - Device Certificates
                - Provides an extra layer of security by identifying devices with unique certificates that can be revoked if necessary
    - Summary
        - Mobile Device Management (MDM) is a critical component of Enterprise Mobility Management (EMM)
            - It provides centralized control over mobile devices, ensuring that security policies are enforced across the organization
            - MDM features include Application Control, Password Management, Multi-Factor Authentication, Token-based Access, Patch Management, Remote Wipe, Geofencing, and Device Certificates
        - These features help organizations reduce risks associated with mobile devices, especially those connecting to unsecured networks, ensuring sensitive data and enterprise systems are protected

- **Attack Surface Management**
    - Attack Surface Management
        - The practice of identifying, monitoring, and reducing the attack vectors available in an enterprise's network to minimize exposure to potential threats
    - Attack Surface Monitoring
        - Purpose

        - Continuous observation of all potential entry points into a network (services, interfaces, devices, exposed endpoints) to detect vulnerabilities, unauthorized changes, and security misconfigurations
    - Tools Used
        - Nmap
            - Network mapper used to scan open ports and identify active services
        - Nessus & Qualys
            - Vulnerability scanners that analyze services for known vulnerabilities, outdated software, and misconfigurations
        - SIEM Systems (e.g., Splunk, IBM QRadar, ArcSight)
            - Centralized systems that aggregate logs, detect unusual patterns, and provide real-time alerts for abnormal activities
        - Endpoint Detection and Response (EDR) Tools (e.g., CrowdStrike, Microsoft Defender for Endpoint)
            - Monitor device behavior and detect suspicious network connections or activities
    - Outcome
        - Early detection of potential threats, enabling the security team to take proactive steps before vulnerabilities are exploited
- Attack Surface Reduction
    - Purpose

        - Minimizing the number of entry points available to attackers by removing or securing unnecessary or risky elements within the network
    - Actions Taken
        - Close Unused Ports
            - Use firewalls (e.g., Palo Alto Networks, Cisco ASA) to block unnecessary ports
        - Disable Unnecessary Services
            - Turn off or replace outdated services like Telnet or FTP with secure alternatives like SSH and SFTP
        - System Hardening
            - Apply security benchmarks (e.g., CIS benchmarks) to secure operating systems, applications, and databases
        - Patch Management
            - Use tools like Microsoft SCCM, WSUS, or Spacewalk to automate patch deployment and ensure systems are updated against known vulnerabilities
    - Outcome
        - A reduced attack surface, making it harder for attackers to find exploitable entry points
- Summary
    - Attack Surface Management involves two key activities
        - Attack Surface Monitoring
            - Continuously observes the network to detect vulnerabilities and unauthorized changes using tools like Nmap, Nessus, and SIEMs

- Attack Surface Reduction
    - Takes proactive steps to minimize attack vectors by disabling unnecessary services, closing unused ports, and applying patches to keep systems secure
- Together, these efforts help organizations detect and prevent potential attacks by reducing their exposure to threats

Data Security Concepts

Objective 3.8: Apply an appropriate cryptographic use case and/or technique

- **12_02: Data Integrity**
  - Data Integrity
    - Ensures data is accurate, consistent, and reliable throughout its lifecycle, validating that it has not been altered or corrupted
    - Hashing is a common technique used to validate data integrity by creating a unique "fingerprint" of the data
  - Hashing
    - Definition
      - A one-way cryptographic function that transforms data of any size into a fixed-length output called a hash digest
    - Characteristics
      - Fixed-length output
        - Regardless of input size, the hash output (digest) is always the same length (e.g., MD5 produces 128-bit hashes)
      - Consistency
        - The same input always results in the same output
      - Irreversibility
        - Hashing is a one-way process; you cannot reverse a hash to retrieve the original data
      - Uniqueness

            - Even the slightest change in the input data will produce a completely different hash
- Hashing Algorithms
    - MD5 (Message Digest Algorithm 5)
        - Produces 128-bit hash digest. Vulnerable to collisions (when two different inputs produce the same hash), making it unsuitable for secure applications
    - SHA (Secure Hash Algorithm) Family
        - SHA-1
            - 160-bit digest, but also vulnerable to collisions
        - SHA-2
            - Includes SHA-224, SHA-256, SHA-384, and SHA-512, providing stronger protection against collisions
        - SHA-3
            - The latest, most secure version, offering up to 120 rounds of computation for enhanced security
    - RIPEMD (Race Integrity Primitives Evaluation Message Digest)
        - Developed independently, used for privacy and decentralized applications
        - Versions include 128, 160, 256, and 320-bit digests
- Applications of Hashing
    - File Integrity Verification
        - Hashes ensure that files have not been altered during transfer or storage
        - By comparing the hash of a downloaded file to the hash provided by the vendor, users can confirm file integrity

        - Software Download Verification
            - Hash values provided by software vendors allow users to verify that the downloaded software has not been tampered with
        - File Integrity Management (FIM)
            - FIM tools (e.g., Tripwire, OSSEC, AIDE) monitor critical files by creating baseline hashes and detecting unauthorized changes
        - Digital Signatures
            - Combine data integrity with non-repudiation by hashing the message and encrypting the hash with the sender's private key
            - The recipient verifies the message's integrity by decrypting the hash and comparing it to their own hash of the message
    - Summary
        - Hashing plays a crucial role in ensuring data integrity by creating a unique digital "fingerprint" (hash) for any given data
        - Reliable hashing algorithms like SHA-2, SHA-3, and RIPEMD provide stronger protection than outdated algorithms like MD5, which are prone to collisions
        - Hashing is widely used for file integrity verification, software download validation, file integrity management, and digital signatures to ensure data authenticity and integrity throughout its lifecycle

- **Integrity Use Cases**
    - Integrity Use Cases
        - Maintain the accuracy, trustworthiness, and security of data or software, ensuring its functionality has not been altered
    - Software Provenance

- Definition
  - Refers to understanding the origin and history of software components, ensuring transparency about who developed it, where it came from, and how it has evolved over time
- Importance
  - Essential for verifying trust in the software components used within an organization, helping prevent supply chain attacks
- Tools
  - Version Control Systems (e.g., Git)
    - Tracks every change made to the software, including who made the changes and when
  - Software Bill of Materials (SBOM)
    - A detailed inventory of all components, dependencies, and libraries in a software application, providing transparency into what makes up the software
  - Software Composition Analysis (SCA)
    - Tools like Snyk, Black Duck, or WhiteSource scan the SBOM to verify that components are from trusted sources and have not been compromised
    - They also check for vulnerabilities and licensing issues
  - Hashing
    - Verifies the integrity of software components by comparing cryptographic hashes with those from known trusted sources
- How Software Provenance is Ensured

        - Enterprises track and verify the history of software components using version control and maintain a clear record of software composition using SBOM
        - SCA tools scan the codebase to ensure only trusted, secure components are used, preventing compromised versions from entering the development pipeline
        - Helps prevent supply chain attacks, where malicious actors attempt to insert compromised components during the software development process

- Software and Code Integrity
    - Definition
        - Ensures that software and code remain secure, reliable, and unaltered from their original trusted state
    - Importance
        - Protects software from unauthorized modifications that could introduce vulnerabilities or malicious code
    - Techniques
        - Cryptographic Hashing
            - Used to create a digital fingerprint for the software
            - Hashes are compared to the original versions to detect unauthorized changes
        - Code Signing
            - Developers sign their code with a digital certificate from a trusted Certificate Authority
            - The code's hash is encrypted with the developer's private key, forming a digital signature

            - Users verify the signature with the public key to ensure the code has not been altered
    - How Software Integrity is Ensured
        - Cryptographic Hashes are generated for each version of the code and compared to ensure no unauthorized changes have occurred
        - Code Signing confirms the authenticity and integrity of software during deployment
            - In CI/CD pipelines, tools like Jenkins and GitLab CI automatically check hashes and signatures for every code change to ensure no tampering
- Summary
    - Software Provenance
        - Ensures that organizations know the origin and history of their software components
        - Tools like Version Control Systems, SBOMs, and SCA tools help maintain transparency and verify that components are trusted and secure
    - Software and Code Integrity
        - Protect the software from unauthorized modifications through cryptographic hashing and code signing, ensuring the software remains in its original, trusted state
    - Together, these practices safeguard the trustworthiness, security, and reliability of software, preventing unauthorized changes and potential security risks

- **Blockchain**
    - Blockchain
        - A decentralized, distributed ledger technology that records data in a secure, immutable, and transparent manner across a network of nodes, forming an unchangeable series of blocks that ensures data integrity and trustworthiness
    - Key Components of Blockchain
        - Decentralization and Distribution
            - Definition
                - Blockchain operates without a central authority; instead, it is maintained by a network of computers, or nodes, distributed worldwide
            - Purpose
                - Prevents any single entity from controlling or tampering with the blockchain, enhancing security and trust
            - Example
                - In a financial network using blockchain, all participating nodes validate and store each transaction, ensuring a reliable and fraud-resistant record
        - Blocks and Chains
            - Definition
                - Data in a blockchain is stored in blocks. When a block is full, it is linked to the previous block, forming a continuous chain
            - Purpose

            - Each block's link to the previous one ensures data integrity, making it nearly impossible to alter previous records
        - Example
            - A series of transactions in a blockchain forms a sequence, where each transaction block is permanently connected to the last, preserving the history of all transactions
    - Immutability
        - Definition
            - Once data is recorded on a blockchain, it cannot be modified or deleted
        - Purpose
            - Provides a tamper-proof record that is ideal for applications where data integrity is critical, such as in financial transactions or voting systems
        - Example
            - If a transaction is recorded in the blockchain, it cannot be reversed or altered, providing a secure, permanent record
- Immutable Databases
    - Immutable Database
        - Definition
            - A database in which data, once written, cannot be altered or deleted, preserving a permanent record
        - Purpose
            - Ensures that all recorded data remains as it was originally entered, which is valuable for auditing, compliance, and legal documentation

            - Example
                - A financial record stored in an immutable database cannot be modified, making it ideal for regulatory compliance where accurate historical records are essential
        - Blockchain vs. Immutable Database
            - Blockchain
                - Combines immutability with decentralization and cryptography for secure, transparent data management across a network.
            - Immutable Database
                - Ensures data permanence but is not necessarily decentralized or cryptographically secured
    - Summary
        - Blockchain technology provides a secure, transparent, and immutable way to record data across a decentralized network
        - Data is stored in linked blocks, forming an unchangeable chain that resists tampering
        - Immutable databases, while similar in ensuring data permanence, do not inherently include blockchain's decentralization or cryptographic security
        - Blockchain's unique combination of immutability, decentralization, and cryptography makes it particularly suited for applications requiring transparent, fraud-resistant records

- **Data Protection**
    - Data Protection

        - The process of safeguarding data from unauthorized access, alteration, or loss to ensure its confidentiality, integrity, and availability
    - Tokenization
        - Definition
            - Replaces sensitive data with non-sensitive identifiers called tokens, which have no exploitable value outside a secure environment
        - Use Case
            - Commonly used in payment systems (e.g., Apple Pay, Google Pay) to protect credit card information by transmitting a token instead of the actual card details
        - Tools
            - Thales CipherTrust, Protegrity, and TokenEx securely manage the mapping between tokens and original data
        - Advantage
            - Tokens cannot be reverse-engineered without the tokenization system
    - Cryptographic Erase
        - Definition
            - Securely deletes data by encrypting it and destroying the encryption keys, making the data permanently irrecoverable
        - Use Case
            - Effective for decommissioning hardware, ensuring compliance with data protection regulations while keeping drives reusable
        - Tools

        - Drives with Self-Encrypting Drives (SED) technology or third-party software like Secure Erase can perform cryptographic erase
    - Advantage
        - Ideal for SSDs, where traditional data erasure methods are less effective
- Obfuscation
    - Definition
        - Changes the format of data to make it hard to understand, often using masking techniques (e.g., hiding parts of a Social Security number as "*--6789")
    - Use Case
        - Protects personal data in databases or applications where full encryption may be unnecessary
    - Advantage
        - Provides a lightweight way to obscure data without needing cryptographic algorithms
- Cryptographic Obfuscation
    - Definition
        - Applies encryption to transform data into an unreadable format, adding cryptographic security measures to the obfuscation process
    - Use Case
        - Protects software code from reverse engineering by encrypting key elements of the code
    - Tools

            - VeraCrypt for file encryption or obfuscating compilers in development environments
        - Advantage
            - Stronger protection than standard obfuscation, as encrypted data cannot be easily accessed or altered
    - Serialization
        - Definition
            - Converts complex data structures (e.g., objects, arrays) into a format (e.g., JSON, XML, or binary) that can be stored or transmitted and accurately reconstructed
        - Use Case
            - Common in web services, data exchanges, or session storage, ensuring that data maintains integrity during transmission
        - Tools
            - Python Pickle module, Java Serializable interface, and .NET JsonSerializer class manage serialization and deserialization processes
        - Advantage
            - Maintains the structure and integrity of data across systems and platforms
    - Summary
        - Tokenization
            - Replaces sensitive data with non-sensitive tokens for secure transactions
        - Cryptographic Erase

        - Securely deletes data by destroying encryption keys, rendering data unrecoverable while keeping drives reusable
      - Obfuscation
        - Masks or scrambles data to obscure its meaning without using encryption
      - Cryptographic Obfuscation
        - Encrypts data to protect it, making it unreadable without decryption
      - Serialization
        - Converts data into standardized formats for storage and transmission, ensuring integrity during the process

- **Data State Protection**
  - Data State Protection
    - Safeguards data based on its current condition, whether stored, transmitted, or being actively used
  - Data at Rest
    - Definition
      - Data that is stored on physical devices (e.g., hard drives, SSDs, cloud storage) and not actively moving or being processed
    - Use Case
      - Includes stored files, databases, backups, and other static data
    - Protection Methods
      - Encryption

            - The Advanced Encryption Standard (AES) is commonly used to encrypt data into ciphertext, readable only by those with the correct cryptographic key
        - Access Controls
            - Passwords, multi-factor authentication (MFA), and permissions ensure only authorized access
    - Types of Encryption
        - Disk-Level Encryption
            - Encrypts entire storage volumes or drives (e.g., BitLocker, FileVault)
        - Block-Level Encryption
            - Secures specific data segments, often used in virtual storage (e.g., storage area networks)
        - File-Level Encryption
            - Protects individual files with unique encryption keys
        - Record-Level Encryption
            - Encrypts individual database records for more granular control in high-security environments
- Data in Transit
    - Definition
        - Data actively moving across networks, such as over the internet, corporate networks, or wireless connections
    - Use Case
        - Includes information sent via web browsers, email, and other communication protocols
    - Protection Methods

                - TLS (Transport Layer Security)
                    - Encrypts data sent between devices, commonly used in web browsing, file transfers, and email
                - IPsec (Internet Protocol Security)
                    - Creates encrypted tunnels between devices, often used in Virtual Private Networks (VPNs)
            - Additional Considerations
                - TLS Cipher Suites
                    - Define the encryption, key exchange, and authentication algorithms used (e.g., ECDHE_RSA_AES128_GCM_SHA256)
                - IPsec Protocols
                    - Authentication Header (AH) for authentication and integrity, Encapsulating Security Payload (ESP) for confidentiality
                - IPsec Modes
                    - Transport Mode
                        - Encrypts only the data payload
                    - Tunnel Mode
                        - Encrypts both data and headers, often used for site-to-site VPNs
                - IKE (Internet Key Exchange)
                    - Negotiates security associations for secure data exchange
        - Data in Use (Data in Processing)
            - Definition
                - Data actively being accessed or processed by applications, temporarily residing in system memory (e.g., RAM, CPU registers)

- Use Case
  - Includes data being viewed, edited, or manipulated in real-time, such as when working on documents
- Protection Methods
  - In-Memory Encryption
    - Encrypts data stored in memory, ensuring data remains protected even during active processing (e.g., AMD Secure Memory Encryption)
  - Secure Enclaves
    - Isolate sensitive data from the rest of the system during processing (e.g., Intel Software Guard Extensions), protecting against memory scraping attacks

- Summary
  - Data at Rest
    - Stored data needs encryption (e.g., AES) and access controls to protect against unauthorized access
    - Different encryption techniques (disk-level, file-level) can be applied depending on the use case
  - Data in Transit
    - Data moving through networks is protected by TLS or IPsec, creating secure tunnels that prevent interception or tampering during transmission
  - Data in Use
    - Data actively being processed is vulnerable to attacks and must be protected with in-memory encryption and secure enclaves, safeguarding sensitive information while in active use

- **Data Handling and Management**
  - Data Handling and Management
    - Safeguarding data throughout its lifecycle to ensure its confidentiality, integrity, and availability
  - Data Anonymization
    - Definition
      - Modifies data to prevent the identification of individuals, while still allowing for analysis
    - Use Case
      - Used when analyzing data for insights without exposing personal details
    - Protection Methods
      - Data Masking
        - Replaces sensitive data with placeholder values (e.g., replacing credit card numbers with "XXXX-XXXX-XXXX-6789")
        - This method is non-reversible
      - Tokenization
        - Replaces sensitive data with unique tokens that reference the original data, stored securely elsewhere
        - This method is reversible for authorized users
      - Aggregation and Banding
        - Groups data to prevent the identification of specific individuals
        - For example, salaries can be grouped into ranges rather than listing individual salaries

        - Considerations
            - Potential Re-identification
                - Improper anonymization (e.g., small data sets) can still lead to re-identification of individuals
    - Data Sanitization
        - Definition
            - The secure destruction of data to ensure it is irrecoverable before storage media is disposed of, repurposed, or transferred
        - Use Case
            - Used when decommissioning storage devices containing sensitive information (e.g., old hard drives, smartphones)
        - Protection Methods
            - Overwriting
                - Data is overwritten multiple times with random data, zeros, or ones to render it irretrievable
                - Tool Example
                    - Microsoft Sysinternals SDelete
            - Degaussing
                - Uses strong magnetic fields to scramble data on magnetic media (e.g., hard drives, tapes), making it unreadable
                - Note
                    - Only works on magnetic media and destroys the storage device permanently
            - Physical Destruction
                - Shredding, crushing, or incinerating storage media to make data recovery impossible

        - Example
            - Banks often use industrial shredders to destroy hard drives
    - Factory Reset and Sanitization Tools
        - Used on smartphones and other personal devices before selling or donating
- Summary
    - Data Anonymization
        - Removes identifying details from data (e.g., masking, tokenization, aggregation) to protect individual privacy while allowing data analysis
        - However, anonymization must be done properly to prevent re-identification
    - Data Sanitization
        - Ensures sensitive data is securely destroyed and irrecoverable before storage media is disposed of or repurposed
        - Techniques like overwriting, degaussing, and physical destruction are used to prevent unauthorized recovery of data

- **Data Compliance and Privacy**
    - Data Compliance and Privacy
        - Adhering to legal and regulatory requirements to manage and protect personal data while ensuring individual privacy and meeting legal obligations
    - Privacy Applications
        - Definition

- Tools and systems designed to protect personal information, ensuring it is handled according to privacy policies and regulations
- Examples
  - Data Encryption
    - Converts sensitive data into ciphertext, which can only be decrypted by authorized users with the correct key
  - Secure Access Controls
    - Uses technologies like Multi-factor Authentication (MFA) and Role-Based Access Control (RBAC) to limit data access to authorized users
  - Automated Data Anonymization
    - Strips personal identifiers from data to allow its use for analysis without compromising privacy (e.g., masking names and addresses)
- Use Case
  - Hospitals ensuring that only authorized personnel (e.g., doctors, nurses) can access specific patient records using secure access controls

- Legal Considerations
  - Definition
    - The legal frameworks that dictate how personal data must be managed and protected, ensuring transparency and safeguarding individual rights
  - Key Regulations
    - General Data Protection Regulation (GDPR) (EU)
      - Key Principles

            - Data minimization, explicit user consent, right to access, correct, or delete data
        - Penalties
            - Fines up to 20 million euros or 4% of global revenue for non-compliance
    - California Consumer Privacy Act (CCPA) (US)
        - Key Rights
            - Right to know what data is collected, request data deletion, and opt out of data sales
        - Penalties
            - Fines up to $7,500 per intentional violation
    - Non-Compliance Consequences
        - Financial penalties, reputational damage, and legal actions
    - Compliance Measures
        - Regular audits, data protection impact assessments, and staff training
- Regulatory Considerations
    - Definition
        - Specific standards and best practices set by regulatory bodies that organizations must follow to ensure compliance with data protection laws
    - Examples
        - Data Protection by Design and by Default (GDPR)
            - Ensuring privacy is considered at every stage of the data lifecycle
        - Breach Reporting

            - GDPR requires reporting data breaches within 72 hours
        - CCPA
            - Requires businesses to provide clear privacy notices and honor user requests related to their data
    - Compliance Example
        - An online retailer operating in both the EU and California must follow both GDPR and CCPA, implementing data encryption and honoring user data rights
- Summary
    - Privacy Applications
        - Tools like encryption, access controls, and data anonymization that protect personal information according to privacy policies and regulations
    - Legal Considerations
        - Laws like GDPR and CCPA that establish rules for data handling, including user rights and penalties for non-compliance
    - Regulatory Considerations
        - Standards and best practices that guide how organizations collect, store, process, and protect personal data to maintain compliance

Cryptographic Types

Objective 3.8: Apply an appropriate cryptographic use case and/or technique

- **Symmetric Cryptography**
  - Symmetric Cryptography
    - A method of encryption where the same key is used for both encrypting and decrypting data
  - Symmetric Cryptography
    - Definition
      - Uses a single key for both encryption and decryption, ensuring efficiency for bulk encryption and secure communications
    - Common Algorithms
      - AES (Advanced Encryption Standard)
        - Secure and efficient, popular for its performance in both block and stream cipher modes
      - DES (Data Encryption Standard)
        - Outdated 56-bit key algorithm, vulnerable to brute-force attacks
      - 3DES (Triple DES)
        - Applies DES encryption three times, offering more security but at a slower speed
      - Blowfish/Twofish
        - Blowfish is fast and flexible but replaced by Twofish, which offers improved security with up to 256-bit keys

            - RC4
                - Once widely used stream cipher, now considered insecure. ChaCha20 is a modern and secure alternative
        - Use Cases
            - Encrypting file storage, secure network communications, and bulk data protection
    - One-Time Pad
        - Definition
            - Symmetric encryption technique that uses a random key as long as the message to create theoretically unbreakable encryption
        - Key Features
            - Perfect Secrecy
                - Each bit of the message is combined with a random bit of the key (e.g., XOR), making the ciphertext random and unbreakable if the key is never reused
            - Challenges
                - Secure distribution and management of long random keys make it impractical for everyday use
        - Use Cases
            - Ideal for high-security communications like military or diplomatic messages
    - Lightweight Cryptography
        - Definition
            - Designed for environments with limited computational resources, such as IoT devices and embedded systems
        - Key Features

                - Lower Computational Demands
                    - Uses simpler mathematical operations, smaller key sizes, and fewer encryption rounds
                - Shorter Block Sizes
                    - Handles smaller chunks of data to reduce the processing load on devices with limited power
            - Use Cases
                - Protecting data in low-power devices like IoT sensors, smart devices, and embedded systems, where resource efficiency is key
        - Summary
            - Symmetric Cryptography
                - Efficiently uses the same key for both encryption and decryption, making it ideal for secure communications and bulk data encryption
            - One-Time Pad
                - Offers unbreakable encryption using a truly random key as long as the message but faces practical challenges in key management
            - Lightweight Cryptography
                - Tailored for low-power environments, balancing security with minimal computational demands, especially in IoT and embedded systems

- **Symmetric Algorithms**
    - Symmetric Encryption
        - A type of encryption where the same key is used for both encryption and decryption

- Key Concept
    - Encryption Overview
        - Encryption
            - Converts readable information (plaintext) into unreadable ciphertext using an encryption algorithm and key
        - Symmetric Cryptography
            - Uses the same key to lock (encrypt) and unlock (decrypt) data, making it fast and efficient for large datasets
        - Analogy
            - Like a house key used to both lock and unlock the same door—both parties share the same key for encryption and decryption
    - Stream Ciphers
        - Definition
            - Encrypts data bit by bit or byte by byte, making it ideal for real-time applications such as audio or video streaming
        - How It Works
            - Uses a key stream generator that produces a pseudo-random bit stream combined with plaintext using XOR to create ciphertext
        - Key Concept
            - Security depends on the uniqueness of the key stream, often generated using an Initialization Vector (IV)
        - Example
            - RC4 (an older, now insecure stream cipher) and ChaCha20 (a modern, secure stream cipher)

        - Block Ciphers
            - Definition
                - Encrypts data in fixed-size blocks (e.g., 64-bit or 128-bit blocks) using a secret key
            - How It Works
                - Processes data in chunks, offering strong security for file encryption and secure communications
            - Key Features
                - Utilizes confusion (complex relationship between key and ciphertext) and diffusion (spreads plaintext influence throughout ciphertext)
            - Common Modes of Operation
                - Electronic Codebook (ECB)
                    - Encrypts each block independently (less secure)
                - Cipher Block Chaining (CBC)
                    - Links blocks together, improving security
                - Galois/Counter Mode (GCM)
                    - Supports both encryption and integrity checks
    - Common Symmetric Encryption Algorithms
        - Advanced Encryption Standard (AES)
            - Key Lengths
                - 128, 192, or 256 bits
            - Block Size
                - 128 bits
            - Uses

        - WPA2/WPA3 Wi-Fi security, file encryption, secure messaging
      - Strength
        - Strong security and fast processing, can operate as both a block and stream cipher (e.g., AES-GCM)
    - Data Encryption Standard (DES) & Triple DES (3DES)
      - DES Key Length
        - 56 bits (considered insecure today)
      - 3DES Key Length
        - 112 bits (applies DES three times)
      - Uses
        - Historically used in banking and legacy systems but largely replaced by AES due to vulnerability to brute-force attacks
    - Blowfish & Twofish
      - Blowfish Key Length
        - 32 to 448 bits (uses 64-bit block size, now considered less secure)
      - Twofish Key Length
        - Up to 256 bits (uses 128-bit block size)
      - Uses
        - Twofish is an improvement over Blowfish, offering stronger security and better performance
    - Rivest Cipher 4 (RC4)
      - Type
        - Stream cipher
      - Use

            - Was widely used in SSL and WEP, but vulnerabilities have rendered it insecure
        - Replacement
            - ChaCha20—secure, fast, and widely used in internet protocols like TLS
    - ChaCha20
        - Type
            - Modern stream cipher
        - Uses
            - Secures web browsing (TLS), especially in low-power and high-performance environments
        - Strength
            - Efficient, secure, and resistant to cryptographic attacks
- Summary
    - Symmetric Encryption
        - Uses the same key for both encryption and decryption, providing efficient security for large datasets
    - Stream Ciphers
        - Encrypt data bit by bit or byte by byte (e.g., RC4, ChaCha20)
    - Block Ciphers
        - Encrypt data in fixed-size blocks (e.g., AES, DES, 3DES, Twofish)
    - Advanced Encryption Standard (AES)
        - The most widely used algorithm, offering strong security and flexibility in both block and stream encryption

- **Symmetric Cryptography Considerations**
    - Symmetric Cryptography
        - A type of encryption where the same key is used for both encrypting and decrypting data
    - Resource Considerations
        - Importance
            - Resource constraints impact performance, efficiency, and practicality of encryption algorithms
        - Key Factors
            - Computational Power
                - The processing power required for encryption and decryption
            - Memory Usage
                - Storage needed for keys, initialization vectors, and intermediate data
            - Hardware Acceleration
                - Boosts performance for resource-intensive algorithms like AES
            - Power Consumption
                - Critical for battery-operated devices, lightweight algorithms help preserve battery life
        - Algorithm Resource Profiles
            - AES
                - Efficient with hardware support, but resource-heavy with larger key sizes
            - 3DES

            - Resource-heavy due to repeated encryption steps
        - Blowfish
            - Fast and memory-efficient, but uses a smaller block size, posing security risks
        - ChaCha20
            - Optimized for low-resource environments, offering strong security with minimal power consumption
- Centralized vs. Decentralized Key Management
    - Centralized Key Management
        - Definition
            - A single entity controls and manages all encryption keys
        - Advantages
            - Simplified management and updates from a single location
            - Easier to control key access within an organization
        - Disadvantages
            - Single point of failure
                - If the central system is compromised, all keys could be exposed
        - Example
            - Large organizations using centralized systems to manage keys for employees
    - Decentralized Key Management
        - Definition
            - Control of encryption keys is distributed across multiple entities
        - Advantages

            - Stronger security due to the absence of a single point of failure
            - Keys are managed separately across different entities, enhancing security
        - Disadvantages
            - Increased complexity in managing and coordinating keys across multiple platforms
        - Example
            - Multi-cloud environments where each cloud provider manages its own encryption keys independently
- Summary
    - Resource Considerations
        - Resource constraints (computational power, memory, hardware acceleration, power consumption) influence algorithm selection
    - Key Management
        - Centralized
            - Easier to manage but vulnerable to a single point of failure
        - Decentralized
            - More secure but more complex to manage across multiple entities

- **Asymmetric Cryptography**
    - Asymmetric Cryptography
        - Also known as public-key cryptography, it uses a pair of keys—one public and one private—for encryption and decryption

            - One key encrypts, and the other decrypts, making them a pair, but the same key cannot perform both functions
    - Key Concepts
        - Asymmetric Cryptography
            - Key Pair
                - Involves a public key for encryption and a private key for decryption, ensuring secure communication without sharing secret keys
            - How It Works
                - Data encrypted with the public key can only be decrypted with the corresponding private key and vice versa
            - Common Algorithms
                - RSA
                    - Relies on the difficulty of factoring large prime numbers, used in SSL/TLS and digital signatures
                - DSA (Digital Signature Algorithm)
                    - Used for digital signatures to ensure data authenticity and integrity
                - Diffie-Hellman (DH)
                    - A key exchange protocol that allows two parties to derive a shared secret key over an unsecured communication channel
                - El Gamal
                    - Used for encryption and digital signatures but requires larger key sizes for security
                - Elliptic Curve Cryptography (ECC)

                - Offers strong security with smaller key sizes, ideal for mobile and IoT devices
        - Strengths and Weaknesses
            - Advantages
                - No need to share a secret key, reducing risk during transmission
                - Ideal for digital signatures and secure key exchanges
            - Disadvantages
                - Computationally intensive, making it slower than symmetric encryption
                - Limited to encrypting small amounts of data
                - The private key must remain secret—compromise of the private key risks the entire system
        - Applications
            - Key Exchange
                - Asymmetric encryption is often used to securely exchange a symmetric key, which is then used for bulk data encryption
            - Digital Signatures
                - Ensures the authenticity and integrity of data or software, verifying the sender and that the content has not been tampered with
    - Code Signing
        - Definition
            - A method of using asymmetric cryptography to sign software, ensuring it is from a trusted source and has not been altered

        - How It Works
            - A developer signs the code with their private key, creating a digital signature
            - The user's system uses the public key to verify the signature, ensuring that the software is legitimate and unaltered
        - Importance
            - Prevents tampered or malicious software from being installed. Builds trust between software developers and users
    - Summary
        - Asymmetric Cryptography
            - Uses two keys—one public and one private—for secure communication, digital signatures, and key exchanges
        - RSA, DSA, Diffie-Hellman, ECC, and El Gamal
            - Common asymmetric algorithms used in various security applications
        - Code Signing
            - Ensures software authenticity and integrity using digital signatures, protecting users from malicious tampering

- **Asymmetric Algorithms**
    - Asymmetric Algorithms
        - Cryptographic techniques that use two keys—a public key for encryption and a private key for decryption
        - This method enhances security by ensuring that only the private key holder can decrypt data encrypted with the public key
    - RSA (Rivest-Shamir-Adleman)

        - Definition
            - RSA is a widely used asymmetric encryption algorithm based on the difficulty of factoring large prime numbers
        - How It Works
            - Two large prime numbers are multiplied to form a product (n), part of the public key
            - Factoring this product back into primes is computationally infeasible, ensuring security
        - Key Sizes
            - 1024, 2048, 4096 bits (larger key sizes offer stronger security)
        - Applications
            - Secure web communications (SSL/TLS), key exchanges, and digital signatures
        - Strength
            - Strong security and flexibility in key size, but computationally intensive
    - DSA (Digital Signature Algorithm)
        - Definition
            - DSA is an asymmetric algorithm designed specifically for creating digital signatures
        - How It Works
            - DSA uses complex mathematical problems (discrete logarithms) to verify that a message comes from a legitimate sender and has not been altered
        - Key Feature

            - Faster signature generation compared to RSA, but slower in verifying signatures
        - Applications
            - Verifying data authenticity and integrity in secure communications
    - Diffie-Hellman (DH)
        - Definition
            - DH is used for securely exchanging symmetric keys over an unsecured communication channel
        - How It Works
            - Two parties independently generate a shared secret by mixing their private information with a public element
        - Analogy
            - Like mixing colors without revealing the secret base colors, both parties create the same final color without sharing the private information
        - Applications
            - Secure key exchanges for VPN tunnels, encryption protocols
        - Weakness
            - Does not provide authentication; vulnerable to man-in-the-middle attacks unless supplemented with digital certificates
    - Elliptic Curve Cryptography (ECC)
        - Definition
            - ECC uses elliptic curves for encryption, providing high security with smaller key sizes compared to RSA
        - How It Works

            - ECC is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is computationally difficult to reverse
        - Efficiency
            - Offers similar security to RSA but with much shorter keys (e.g., 256-bit ECC key ≈ 3072-bit RSA key)
        - Applications
            - Mobile devices, IoT, secure communications, and digital signatures (ECDSA)
        - Strength
            - Provides strong security with reduced computational and resource demands, ideal for low-power devices
    - Summary
        - Asymmetric Cryptography
            - Uses two keys—one public and one private—for encryption and decryption, enhancing security by eliminating the need to share a secret key
        - RSA
            - Based on the difficulty of factoring large prime numbers, commonly used for secure data transmission and digital signatures
        - DSA
            - Focuses on digital signatures, ensuring data authenticity and integrity, but slower at verifying signatures
        - Diffie-Hellman (DH)

            - Used for secure key exchanges, allowing two parties to establish a shared secret without directly transmitting it
        - Elliptic Curve Cryptography (ECC)
            - Offers strong security with smaller key sizes, making it efficient for mobile and IoT devices, and ideal for secure communications and digital signatures

- **Digital Signatures**
    - Digital Signature
        - A cryptographic method used to verify the authenticity and integrity of a message or document
        - Digital signatures use two cryptographic keys
            - Private Key
                - Used to create (or "sign") the digital signature
            - Public Key
                - Used by the recipient to verify the digital signature
    - Key Components of Digital Signatures
        - Digital Signature Process
            - Definition
                - A digital signature is created by generating a unique hash digest of the message and encrypting it with the sender’s private key
            - Purpose
                - Ensures that the message has not been altered (integrity) and verifies the sender's identity (authenticity)
            - Example Walkthrough

- Message Hashing
  - The sender's email client runs the message through a hashing algorithm (e.g., SHA-256), creating a unique hash digest
- Hash Encryption
  - The hash digest is encrypted with the sender's private key, creating the digital signature
- Verification
  - The recipient decrypts the signature with the sender's public key
  - The recipient's email client then compares the decrypted hash digest with a newly calculated hash of the received message
  - If they match, the message is confirmed as authentic and unaltered

- Non-repudiation
  - Definition
    - Assurance that the sender cannot deny having signed the message because the digital signature was created with the sender's private key, which only the sender possesses
  - Purpose
    - Verifies the origin of the message, providing proof that the message was signed by the claimed sender
  - Example

- In the example, only the sender (Alice) could have created the digital signature because she is the only one with access to her private key

- Confidentiality
  - Definition
    - Ensures that only the intended recipient can read the message by encrypting the content with the recipient's public key
  - Purpose
    - Protects the message from unauthorized access
  - Example
    - For confidentiality, the sender (Alice) encrypts the message itself using the recipient's (Bob's) public key
    - Only Bob, with his private key, can decrypt and read the message

- Combined Use of Digital Signatures and Encryption
  - Encrypting and Digitally Signing a Message
    - Process
      - Step 1
        - The sender hashes the message to create a unique hash digest
      - Step 2
        - The sender encrypts the hash digest with their private key to create the digital signature
      - Step 3

                ■ For confidentiality, the sender encrypts the message itself with the recipient's public key
            ● Purpose
                ○ Ensures the message's authenticity, integrity, non-repudiation, and confidentiality
            ● Requirements
                ○ Both sender and recipient must have their own public/private key pairs
        ■ Practical Application
            ● An email client configured with public/private key pairs can be set to both encrypt and sign emails, providing a secure and trustworthy communication channel
    ○ Summary
        ■ Digital signatures are a cryptographic method for ensuring the authenticity and integrity of a message or document
        ■ They use a private key for signing and a public key for verification
        ■ Digital signatures offer
            ● Authenticity
                ○ Verifying the sender's identity
            ● Integrity
                ○ Ensuring the message has not been altered
            ● Non-repudiation
                ○ Preventing the sender from denying the signature
        ■ To provide confidentiality, the message itself must be encrypted using the recipient's public key

        - Together, digital signatures and encryption create a comprehensive security process for protecting data

- **Asymmetric Cryptography Use Cases**
    - Asymmetric Cryptography
        - A cryptographic method that uses a pair of keys—one public and one private—for encrypting and decrypting data, providing secure communications and user authentication
    - Certificate-Based Authentication
        - Definition
            - A secure method of verifying identities using digital certificates issued by a trusted Certificate Authority (CA) within a Public Key Infrastructure (PKI)
        - How It Works
            - Client Authentication
                - A server verifies the identity of a device or user based on the digital certificate presented by the client
            - Server Authentication
                - A client verifies the authenticity of a web server through its digital certificate, ensuring the user is connecting to a legitimate server
        - Application
            - Integrated with Single Sign-On (SSO) systems to simplify access, allowing users to authenticate once and access multiple resources
        - Strength

            - Provides strong security by eliminating passwords and ensuring only authorized users or systems access protected resources
    - Passwordless Authentication
        - Definition
            - An authentication method that allows users to access systems without entering traditional passwords, using alternatives like cryptographic keys and biometrics
        - Types of Factors
            - Ownership Factors
                - Something the user has, like a hardware token or smartphone
            - Biometric Factors
                - Something the user is, such as fingerprints or facial recognition
        - Strength
            - Eliminates vulnerabilities associated with passwords (e.g., phishing, brute-force attacks) and improves user experience by removing the need to remember complex passwords
        - Example
            - Apple's Face ID and Touch ID, which use biometrics for secure device access
        - Multi-Factor Authentication (MFA)
            - Combining biometric or ownership factors with a second factor, such as a one-time passcode, for enhanced security
    - Secure Email
        - Definition

            - Asymmetric encryption is used to secure email communications, ensuring confidentiality, integrity, authentication, and non-repudiation
        - How It Works
            - Encryption
                - The email message is encrypted with the recipient's public key, ensuring only the intended recipient can decrypt and read it
            - Digital Signatures
                - The sender encrypts the email's hash with their private key, providing authenticity and ensuring the message hasn't been tampered with
        - S/MIME (Secure/Multipurpose Internet Mail Extensions)
            - Standard
                - Applies encryption and digital signatures directly to each email message, ensuring both sender and receiver can trust the message's confidentiality and integrity
            - Application
                - Integrated into email clients like Microsoft Outlook and Apple Mail to provide end-to-end email encryption
        - Strength
            - Ensures messages are confidential and authentic, protecting against tampering and unauthorized access
    - Summary
        - Certificate-Based Authentication

        - Uses digital certificates to verify identities, providing strong authentication for users and devices
    - Passwordless Authentication
        - Replaces passwords with cryptographic keys and biometrics, improving security and user experience while eliminating password vulnerabilities
    - Secure Email
        - Encrypts email messages with asymmetric encryption, ensuring confidentiality, integrity, and authenticity through encryption and digital signatures (e.g., S/MIME)

PKI Architecture

Objective 2.4: Apply security concepts to the design of access, authentication, and authorization systems

- **Certificate Management**
  - Certificate Management
    - The process of issuing, renewing, revoking, and managing digital certificates throughout their lifecycle to ensure secure communications
  - Certificate Types
    - SSL/TLS Certificates
      - Definition
        - Secure data transmission between browsers and web servers through HTTPS connections
      - Validation Levels
        - Domain Validation (DV)
          - Basic encryption and domain verification
        - Organization Validation (OV
          - Includes organization identity verification
        - Extended Validation (EV)
          - Highest level of verification with visible trust indicators (e.g., green padlock)
      - Application
        - Used to secure websites, indicated by a padlock icon or company name in the browser
    - Client Certificates

- Definition
  - Used to authenticate users or devices within networks
- Application
  - Replaces passwords with cryptographic verification, commonly used for secure email and access control in enterprises

■ Code-Signing Certificates

- Definition
  - Verifies the authenticity and integrity of software and code
- Application
  - Used by developers to sign applications and drivers, ensuring that the code has not been altered or tampered with

■ Wildcard Certificates

- Definition
  - Secures multiple subdomains under one primary domain (e.g., *.example.com)
- Application
  - Simplifies certificate management for organizations managing several subdomains

■ Subject Alternative Name (SAN) Certificates

- Definition
  - Secures multiple domains or hostnames with one certificate
- Application

        - Ideal for organizations managing multiple websites or services under different domains
- Certificate Extensions
    - Definition
        - Fields within a certificate that specify its intended use, restrictions, and other critical information
    - Critical Extensions
        - Must be processed and understood by the application, or the certificate is rejected
    - Non-Critical Extensions
        - Optional for the application to process, allowing flexibility.
    - Common Extensions
        - Key Usage
            - Specifies functions like digital signature, key encipherment, or certificate signing
        - Extended Key Usage
            - Defines more specific purposes such as server/client authentication or email protection
        - Subject Alternative Name (SAN)
            - Lists additional domains, IP addresses, or email addresses tied to the certificate
        - Basic Constraints
            - Defines if the certificate is for an end-user or a Certificate Authority
        - Authority Key Identifier/Subject Key Identifier

                - Establishes relationships between a certificate and the CA for validation
            - Certificate Revocation List (CRL) Distribution Points
                - Lists where to find the CRL to check for revoked certificates
- Certificate Encoding and File Formats
    - Certificate Encoding
        - Basic Encoding Rules (BER)
            - Flexible, supports multiple encoding types
        - Canonical Encoding Rules (CER)
            - Standardized version of BER
        - Distinguished Encoding Rules (DER)
            - Strict encoding, ideal for cryptographic operations
    - File Formats
        - .pem
            - Commonly used for server certificates, CA certificates, and private keys
        - .p12/.pfx
            - Used for bundling certificates, intermediate certificates, and private keys, ideal for secure transfer
        - .p7b
            - Contains certificate chains without private keys, used for distribution and validation of certificates
- Summary
    - Certificate Management
        - Involves managing the entire lifecycle of digital certificates, from issuance to renewal and revocation

        - Certificate Types
            - Include SSL/TLS certificates for secure communications, client certificates for user authentication, code-signing certificates for verifying software, wildcard certificates for securing multiple subdomains, and SAN certificates for securing multiple domains
        - Certificate Extensions
            - Provide additional information about a certificate's intended use, including key usage, SAN, and revocation details
        - File Formats
            - PEM, P12/PFX, and P7B formats are used for storing and transferring certificates and private keys

- **Certificate Authority (CA) Functions**
    - Certificate Authority Functions
        - Certificate Authority (CA)
            - A trusted entity responsible for issuing, managing, revoking, and validating digital certificates in a Public Key Infrastructure (PKI) framework, thereby ensuring secure and trustworthy digital communications
        - Registration Authority (RA)
            - Acts as an intermediary between end-users and the Certificate Authority, accepting and validating certificate requests before forwarding them to the CA for issuance
    - Certificate Authorities (CA)
        - Definition

        - Trusted entities that create and sign digital certificates, binding public keys to identities
    - Purpose
        - Facilitates authentication within PKI, enabling trust among parties
    - Public vs. Private CAs
        - Public CAs
            - Widely trusted by devices and browsers (e.g., Verisign, Let’s Encrypt)
        - Private CAs
            - Used internally within organizations, requiring internal trust mechanisms
    - Primary Functions
        - Issuance and Management
            - Providing certificates and overseeing their lifecycle
        - Validation
            - Verifying identities of certificate applicants
        - Establishing Trust
            - Building and maintaining trusted relationships
        - Repository Management
            - Storing and managing certificate data
        - Lifecycle Oversight
            - Managing issuance through to certificate revocation
    - Hierarchical Structure
        - Root CA
            - The top-level, trusted authority
        - Intermediate CAs

            - Subordinate entities issuing certificates to end-users
        - Example
            - A military Root CA has branches for different armed forces as Intermediate CAs, ensuring trust across branches
- Registration Authorities (RA)
    - Definition: Entities that accept and validate digital certificate requests before CA issuance.
    - Purpose: Ensures that requesters have authorization for the requested certificates.
- Certificate Signing Request (CSR)
    - Definition
        - A Base64-encoded file containing the requester's details, public key, and key information for certificate creation
    - Required CSR Data
        - Organization Details
            - Common Name (CN)
                - Fully qualified domain (e.g., www.example.com)
            - Subject Alternative Name (SAN)
                - Optional, for multi-domain support
            - Organization (O)
                - Legal name of the entity
            - Organizational Unit (OU)
                - Department within the organization
            - Location (L), State (S), Country (C)
                - Geographical details
        - Public Key

            - Used by the CA for digital endorsement
        - Key Type and Length
            - Specifies encryption method and security strength (e.g., RSA 2048-bit)
    - Summary
        - Certificate Authority functions, within a PKI framework, involve the issuance, management, and validation of digital certificates to establish trust in secure digital communications
        - Key entities include
            - Certificate Authorities (CAs)
                - Issue, manage, and validate certificates, ensuring secure, trusted communication by binding identities to public keys
            - Registration Authorities (RAs)
                - Validate certificate requests, verifying authorization before CA approval
        - The structured hierarchy of CAs, supported by RAs, provides a robust framework for digital security, enabling secure, verified interactions across networks

- **Certificate Validation**
    - Certificate Validation
        - The process of ensuring that a digital certificate within the Public Key Infrastructure (PKI) remains trustworthy, unexpired, and has not been revoked
    - OCSP Stapling
        - Online Certificate Status Protocol (OCSP)

        - Definition
            - A protocol that checks the revocation status of a specific certificate without downloading the entire Certificate Revocation List (CRL)
        - Application
            - When a browser requests a website's certificate, it queries the OCSP Responder for the certificate’s status (valid, revoked, or unknown)
    - OCSP Stapling
        - Definition
            - Allows a web server to fetch the certificate’s status from the OCSP Responder and present it directly to the client during the SSL/TLS handshake
        - Application
            - Improves validation speed and privacy, as the client doesn't need to make a separate OCSP request
        - Benefit
            - Enhances the efficiency and privacy of certificate validation during web connections
- Certificate Revocation List (CRL)
    - Definition
        - A list maintained by a Certificate Authority (CA) that contains the serial numbers of certificates that have been revoked before their expiration dates
    - Application

        - Devices use the CRL to check the revocation status of a certificate, ensuring that it hasn't been invalidated.
    - Reasons for Revocation
        - Key compromise
        - Cessation of operation
        - Certificate superseded or no longer needed
    - Suspension
        - A temporary status where a certificate can be reinstated, unlike permanent revocation
    - Use Case
        - Similar to how a driver's license can be revoked before its expiration due to misuse, digital certificates can also be revoked to ensure security
- Certificate Pinning
    - Definition
        - A security technique where a known valid certificate or public key is stored within an application
    - Application
        - When establishing a connection, the application compares the stored certificate with the one presented by the server
    - Benefit
        - Protects against man-in-the-middle attacks by bypassing the standard CA chain of trust
    - Example
        - Ensures that only a specific server’s certificate is trusted during secure communications, such as banking apps

        - HTTP Public Key Pinning (HPKP)
            - Definition
                - A deprecated form of certificate pinning where websites sent a list of trusted public keys in an HTTP header
            - Issue
                - Vulnerabilities in HPKP and misconfiguration risks (e.g., permanently blocking users) led to its deprecation
    - Summary
        - Certificate Validation
            - Ensures the trustworthiness of digital certificates through techniques like OCSP stapling, CRLs, and certificate pinning
        - OCSP Stapling
            - Enhances the validation process by allowing servers to provide certificate status updates directly during the SSL/TLS handshake, improving speed and privacy
        - CRL
            - A list maintained by Certificate Authorities that tracks revoked certificates to prevent their continued use
        - Certificate Pinning
            - Secures applications by hard-coding a trusted certificate or public key, preventing man-in-the-middle attacks and securing communication channels

- **Certificate Deployment**
    - Certificate Deployment
        - The process of distributing and installing digital certificates across systems and devices to enable secure communications
    - Certificate Templates
        - Definition
            - Predefined configurations that standardize the settings and permissions of certificates across an organization
        - Purpose
            - Ensures consistency, simplifies management, and enforces security standards for all issued certificates
        - Application
            - Templates include settings like encryption level, expiration date, and allowed uses, acting as a blueprint for issuing certificates
        - Benefits
            - Faster and more consistent certificate issuance
            - Easy updates to certificate configurations when security policies change
            - Control over who can request and receive certificates, adding a layer of security
    - Certificate Deployment Approach
        - Definition
            - The method used to distribute and install digital certificates across servers, devices, and applications
        - Types
            - Manual Deployment

            - Process
                - Installing certificates one by one on each device
            - Application
                - Suitable for small environments but time-consuming and prone to errors in larger networks
        - Automated Deployment
            - Process
                - Using automated tools to distribute certificates across multiple devices simultaneously
            - Application
                - Preferred in large environments for its efficiency, consistency, and error reduction
    - Choosing an Approach
        - Depends on organization size, infrastructure complexity, and the number of certificates needed
        - Automated deployment is often preferred for large organizations
- Certificate Integration
    - Definition
        - The process of configuring systems and applications to use the installed certificates for secure communications
    - Application
        - Involves setting up servers, devices, and applications to actively use certificates for encryption and authentication
        - Examples
            - S/MIME Certificates

                - Used in email servers for encrypting and signing emails
            - SSL/TLS Certificates
                - Configured on web servers to secure web traffic
    - Importance
        - Ensures that certificates are not just installed but are actively securing communications by encrypting data and verifying identities
- Summary
    - Certificate Deployment
        - Distributes and installs digital certificates across systems and devices to ensure secure communications
    - Certificate Templates
        - Standardize the settings and permissions of certificates, making the deployment process faster and more consistent
    - Certificate Deployment Approach
        - Determines how certificates are distributed—manually or through automated tools—depending on the organization's needs and infrastructure size
    - Certificate Integration
        - Ensures certificates are properly set up and actively used within systems to secure communications, like encrypting emails and protecting web traffic

Advanced Cryptographic Concepts

Objective 3.7: Explain the importance of advanced cryptographic concepts

- **Cryptographic Blockers**
  - Cryptographic Blockers
    - Challenges or limitations that hinder the implementation and effectiveness of encryption techniques
    - These blockers often involve a trade-off between performance and security
  - Performance vs Security
    - Definition
      - The balance between the computational speed of cryptographic processes and the strength of security provided by the encryption algorithms
    - Trade-Off
      - Stronger encryption algorithms (e.g., longer key lengths, more rounds of encryption) provide higher security but increase computational overhead, which can reduce system performance
    - Examples of Performance vs Security
      - AES-256
        - Security
          - Highly secure due to 256-bit key length, making it resistant to brute force attacks
        - Performance

            - More rounds of encryption, leading to higher computational resource requirements and slower processing times
        - Application
            - Used in environments where data confidentiality is critical (e.g., military communications, financial transactions)
    - AES-128
        - Security
            - Secure but with lower resistance to future threats (e.g., quantum computing) due to the shorter key length
        - Performance
            - Fewer encryption rounds, resulting in faster processing
        - Application
            - Suitable for environments requiring fast data processing where slightly lower security is acceptable
- Latency and Bandwidth Considerations
    - Latency
        - The time delay introduced when encryption algorithms require heavy computation, affecting the speed of secure data transmission
    - Bandwidth

                - Cryptographic operations can impact the amount of data that can be processed in a given time, especially in systems with limited bandwidth
            - Example
                - Transport Layer Security (TLS) can be configured with different cryptographic suites
                - Using more secure configurations (e.g., Elliptic Curve Cryptography with longer key sizes) improves security but can introduce delays, affecting user experience
        - Mitigating Cryptographic Blockers
            - Hardware Acceleration
                - Definition
                    - The use of dedicated cryptographic processors or hardware security modules to improve the performance of strong encryption algorithms
                - Benefit
                    - Enhances system performance while maintaining high levels of security, allowing robust encryption in high-performance environments
            - Offloading to Specialized Hardware
                - Offloads complex cryptographic tasks to external hardware, freeing up system resources and reducing processing delays
        - Matching Cryptographic Strength to Risk
            - Risk Assessment

            - Organizations must assess the level of risk they are willing to accept and choose encryption algorithms that balance security with acceptable performance
        - Critical Applications
            - In high-risk environments, the focus is on maximizing security even at the cost of performance (e.g., military systems, secure financial platforms)
        - Low-Risk Applications
            - For lower-risk scenarios, a balance is found by opting for faster algorithms that provide adequate security (e.g., streaming services)
    - Summary
        - Cryptographic Blockers
            - Challenges that impact the implementation of encryption, often revolving around the balance between performance and security
        - Performance vs Security
            - Stronger encryption provides better protection but requires more computational resources, potentially slowing down systems
            - AES-256 vs AES-128
                - Example of balancing high security (AES-256) with performance (AES-128)
        - Latency and Bandwidth
            - Cryptographic operations can introduce delays and affect data transmission efficiency, especially in systems with limited bandwidth
        - Mitigation

                - Using hardware acceleration and offloading tasks to specialized hardware can improve performance without sacrificing security
            - Risk Management
                - Cryptographic strength must be aligned with the acceptable risk level, ensuring that performance remains adequate without compromising data protection

- **Key Management**
    - Key Management
        - The processes and protocols that generate, distribute, store, and securely handle cryptographic keys throughout their lifecycle
    - Key Stretching
        - Definition
            - A technique used to enhance the security of weak or short cryptographic keys by applying iterative hashing or encryption functions to transform a simpler key into a longer and more complex one
        - How It Works
            - The key is processed through thousands of rounds of hashing or encryption, increasing the time and computational resources needed to crack it
        - Common Methods
            - PBKDF2 (Password-Based Key Derivation Function 2)
                - Combines the input password with a unique salt and mixes them through thousands of iterations using HMAC
            - Bcrypt

        - Uses adaptive key stretching with salting, increasing iteration count over time to maintain security against improving hardware capabilities
    - Salting
      - Adds random data to each password before hashing, making it unique and protecting against attacks like dictionary attacks and rainbow tables
    - Purpose
      - Key stretching makes brute-force attacks more time-consuming and difficult by requiring attackers to process the same extensive iterations for each guess
    - Examples
      - PBKDF2 with 310,000 Iterations
        - This method increases the time required for each brute-force attempt, as each guess must go through all the iterations, significantly slowing down attackers
      - Bcrypt in Linux Systems
        - Bcrypt’s adaptability allows systems to increase the iterations over time, keeping it secure against evolving hardware capabilities
  - Key Splitting
    - Definition
      - The process of dividing a cryptographic key into multiple parts, which must be combined to reconstruct the original key and access encrypted data
    - How It Works

            - Each part of the key is stored separately and independently. Accessing the data requires combining all the parts of the key
        - Example
            - Financial Transactions
                - Multiple parts of a key are stored in different secure locations, ensuring that no single person or system can decrypt data without the cooperation of all necessary parties
                - In secure systems like online banking, one part of the key might be stored on a server, another in a hardware module, and the final part is sent to the user via a secure channel
        - Purpose
            - Key splitting increases security by ensuring that no single party has full access to the key, mitigating the risks associated with key compromise
    - Summary
        - Key Management
            - Involves securely generating, distributing, and handling cryptographic keys throughout their lifecycle
        - Key Stretching
            - Strengthens weak or short cryptographic keys by applying iterative hashing or encryption, making brute-force attacks more difficult
        - Key Splitting

            - Divides a cryptographic key into multiple parts to prevent any one entity from having full access to the key, enhancing security in high-trust environments
        - Key Management Techniques
            - Protect cryptographic keys from unauthorized access or attacks by making key cracking more difficult or requiring multiple parties to access sensitive data

- **Encryption Techniques**
    - Encryption Techniques
        - Methods and algorithms used to convert plaintext into ciphertext to protect data from unauthorized access
    - Authenticated Encryption with Associated Data (AEAD)
        - Definition
            - A form of encryption that provides confidentiality, integrity, and authenticity for both the encrypted data and associated data
        - How It Works
            - Encrypts the plaintext while ensuring the integrity of both the ciphertext and additional associated data (AD), which remains visible but protected from tampering
            - Four inputs are used
                - Secret Key
                    - To perform the encryption
                - Nonce/Initialization Vector
                    - A unique value used once to ensure different ciphertexts for identical plaintexts

        - Plaintext
            - The data to be encrypted
        - Associated Data
            - Data that is not encrypted but needs integrity protection
    - Outputs
        - Ciphertext and an authentication tag, which verifies integrity and authenticity
- Examples
    - AEAD
        - Used in secure communication protocols like TLS (Transport Layer Security) to ensure both message confidentiality and integrity
    - AES-GCM (Advanced Encryption Standard - Galois/Counter Mode)
        - Commonly used in secure web traffic to protect both the data and associated metadata like headers
        - Binds associated data to encrypted messages, ensuring any alterations are detected
- Analogy
    - Like sending a secure package, where the package (encrypted data) is sealed for confidentiality, and the label (associated data) is protected from tampering

- Envelope Encryption
    - Definition
        - A two-layer encryption technique where a data key encrypts the data, and a master key encrypts the data key

        - How It Works
            - First Layer
                - Data is encrypted using a data key
            - Second Layer
                - The data key is encrypted using a stronger master key, which is often managed securely using asymmetric encryption (e.g., RSA)
        - Example
            - Cloud Storage
                - Encrypts files with a data key, then encrypts the data key with a master key managed by the cloud provider, ensuring both the data and keys are protected
        - Analogy
            - Like placing valuables in a locked box (data key) and then putting the box inside a safe (master key) for extra protection
    - Summary
        - Encryption Techniques
            - Convert readable data into unreadable ciphertext to protect it from unauthorized access
        - AEAD
            - Combines encryption with integrity checks, ensuring that both encrypted data and associated data are secure and have not been altered
        - Envelope Encryption

            - Adds an extra layer of security by encrypting the data with a data key and protecting that key with a master key, making the overall system more secure and flexible for managing encryption

- **Security Properties**
    - Security Properties
        - Key features in cryptographic systems that ensure data protection by verifying identities and securing past and future sessions
    - Mutual Authentication
        - Definition
            - A security process where both parties in a communication verify each other’s identities before data is exchanged
        - How It Works
            - Both the client and server exchange credentials, such as digital certificates, to confirm each other's legitimacy
        - Example
            - In secure environments like online banking or VPNs, mutual authentication helps ensure that both the user and the service provider are verified before sensitive data is shared
    - Forward Secrecy (Perfect Forward Secrecy)
        - Definition
            - A property in cryptographic protocols that ensures session keys are unique and not reused, so past and future sessions remain secure even if a private key is compromised
        - How It Works

            - A unique session key is created for each session and used only once, independent of the server's long-term private key
        - Example
            - In protocols like TLS 1.3 and VPNs, each session is protected by a unique key derived from temporary Diffie-Hellman key exchanges, securing past communications even if a private key is exposed
    - Summary
        - Mutual Authentication
            - Ensures both parties in a digital conversation verify each other's identities, reducing risks of unauthorized access and enhancing trust
        - Forward Secrecy
            - Provides security by generating unique session keys that prevent decrypted access to past or future sessions if a private key is compromised
            - This process is used in protocols like TLS 1.3 and WPA3 to maintain secure, independent session encryption

- **Collaborative Considerations**
    - Collaborative Considerations
        - Techniques and protocols that enable secure data sharing and cooperative computing among multiple parties, ensuring data privacy and integrity throughout the process
    - Homomorphic Encryption
        - Definition

        - An encryption technique allowing computations on encrypted data without needing decryption, maintaining data privacy during processing
    - How It Works
        - Data is encrypted with a public key, and an algebraic system enables mathematical operations on the encrypted data without exposing the plaintext
    - Types
        - Partially Homomorphic
            - Supports specific operations like addition or multiplication
        - Somewhat Homomorphic
            - Allows limited operations a certain number of times
        - Fully Homomorphic
            - Permits any computation on encrypted data
    - Application
        - Useful in industries like finance and healthcare, where data processing by third-party services requires privacy protection
    - Analogy
        - Like a locked package that can be rearranged without opening, protecting contents from unauthorized access during transport
- Secure Multiparty Computation (SMPC)
    - Definition
        - A cryptographic method enabling multiple parties to compute a shared result without revealing individual inputs, ensuring privacy in collaborative settings
    - How It Works

      - Each party provides encrypted inputs, and cryptographic protocols ensure only the final result is revealed
    - Techniques
      - Secure Function Evaluation (SFE)
        - Allows a known function computation with hidden inputs
      - Private Function Evaluation (PFE)
        - Extends SFE to keep both inputs and the function private
    - Application
      - Ideal for collaborative data analysis (e.g., joint data statistics), secure voting, and privacy-preserved computations
    - Analogy
      - Like multiple organizations submitting secret ballots to calculate an industry average without revealing individual values
  - Examples
    - Homomorphic Encryption
      - Finance
        - Enables outsourced data analysis on client financial data by third-party services while maintaining client confidentiality
      - Healthcare
        - Allows secure, private data analysis on patient information for research purposes without exposing sensitive health data
    - Secure Multiparty Computation (SMPC)
      - Joint Data Analysis

            - Companies compute shared metrics (e.g., industry averages) without disclosing proprietary data
        - Secure Voting
            - Allows votes to be counted accurately without revealing individual choices, preserving voter privacy
- Summary
    - Collaborative Security Techniques
        - Methods enabling secure data sharing and computing among multiple parties without compromising privacy
    - Homomorphic Encryption
        - Allows operations on encrypted data without decryption, maintaining data confidentiality during processing
    - Secure Multiparty Computation (SMPC)
        - Ensures private data remains confidential in collaborative settings by allowing joint computation without revealing individual inputs

- **Performance Considerations**
    - Performance Considerations
        - Balancing the need for strong security with the speed and efficiency of cryptographic operations to ensure that systems remain both secure and responsive
    - Hardware Acceleration
        - Definition
            - The use of specialized hardware components to perform encryption and decryption tasks faster than standard software-based methods

- Purpose
  - Enhances the speed and efficiency of cryptographic processes, minimizing the load on general-purpose CPUs
- Examples of Hardware Acceleration Components
  - Hardware Security Modules (HSM)
    - Manages cryptographic keys and performs encryption directly within the hardware
  - Trusted Platform Modules (TPM)
    - Provides secure encryption, authentication, and key storage on computers
  - Graphics Processing Units (GPU)
    - Leverages parallel processing capabilities for cryptographic tasks
  - Network Cards with Built-in Encryption (e.g., Intel QuickAssist Technology)
    - Accelerates cryptographic functions at the network level
- Analogy
  - Like adding specialized staff members at a busy checkout line, hardware acceleration divides tasks among components optimized for speed and efficiency, making cryptographic operations faster without overwhelming a single resource

○ Applications of Hardware Acceleration
- High-Frequency Trading Platforms
  - Need
    - Ensures millisecond-level performance without compromising data security

    - Benefit
        - Enables rapid, secure processing of complex encryption tasks, maintaining high system performance
  - Virtual Private Networks (VPNs)
    - Need
        - Balances security with speed for encrypted communication over the internet
    - Benefit
        - Uses dedicated hardware to process encryption, ensuring secure, efficient connections
  - SSL/TLS Protocols
    - Need
        - Establishes secure communication channels for data transmission, especially in high-traffic environments
    - Benefit
        - Hardware acceleration offloads cryptographic tasks from CPUs, reducing the time required to establish secure connections and making the encryption process faster without compromising security
- Summary
  - Performance considerations ensure secure and efficient cryptographic processes by leveraging hardware acceleration
  - Specialized components such as HSMs, TPMs, GPUs, and network cards with built-in encryption enhance system responsiveness by offloading encryption tasks from general-purpose CPUs

        - This balance between speed and security is crucial in high-demand applications like high-frequency trading platforms, Virtual Private Networks, and SSL/TLS protocols, where both robust security and high performance are essential

- **Post-Quantum Cryptography (PQC)**
    - Post-Quantum Cryptography (PQC)
        - Cryptographic algorithms specifically designed to resist quantum computing attacks, ensuring the security of data even as quantum technology advances
    - Quantum Computing and Quantum Cryptography
        - Definition
            - Quantum computing uses qubits, which can exist in multiple states simultaneously due to principles like superposition and entanglement, enabling faster computations than classical computers
        - Impact on Cryptography
            - Quantum computers could break traditional encryption by solving complex mathematical problems that secure today's cryptographic methods
        - Key Properties
            - Superposition
                - Qubits can represent both 0 and 1 states at once, increasing computational capacity
            - Entanglement

            - Linked qubits influence each other's states instantly, enabling coordinated processing
        - Example
            - Quantum algorithms, like Shor's algorithm, can quickly solve problems that underpin traditional encryption (e.g., factoring for RSA, Diffie-Hellman)
    - Post-Quantum Cryptography (PQC) vs. Diffie-Hellman and Elliptic Curve Cryptography (ECC)
        - Diffie-Hellman (DH)
            - Widely used for secure key exchanges, vulnerable to quantum attacks as quantum algorithms can solve the discrete logarithm problem
        - Elliptic Curve Cryptography (ECC)
            - Provides secure key generation with efficient performance, but quantum computers can solve elliptic curve problems efficiently, posing a security risk
        - PQC Approach
            - Develops algorithms that use problems believed to be resistant to quantum computers, ensuring data security beyond the lifespan of DH and ECC
    - Strategies for Quantum-Resistant Security
        - Increasing Key Sizes
            - Increasing key sizes exponentially raises the computational difficulty for brute-force attacks
            - Example

        - Moving from AES-128 to AES-256 increases resistance against quantum brute-force attacks
    - Quantum-Resistant Algorithms
        - Algorithms that can withstand quantum attacks are actively in development, including lattice-based cryptography and supersingular isogeny key exchanges
        - Lattice-Based Cryptography
            - Uses high-dimensional mathematical structures to secure data, resistant to both classical and quantum attacks
    - National Institute of Standards and Technology (NIST) Initiative
        - NIST is developing quantum-resistant standards expected to finalize soon, as quantum technology progresses rapidly
- Examples of Quantum-Resistant Techniques
    - Lattice-Based Cryptography
        - Learning With Errors (LWE)
            - Relies on complex lattice structures that are difficult for quantum computers to solve
        - Ring-LWE
            - A variation of LWE used in quantum-resistant encryption and digital signatures
        - Application
            - Ensures secure key exchanges, digital signatures, and data encryption against quantum threats
    - Other Quantum-Resistant Algorithms
        - Supersingular Isogeny Key Exchanges

        - Based on mathematical isogenies that remain challenging for quantum computers
    - Code-Based Cryptography
        - Relies on error-correcting codes, believed to be quantum-resistant
- Summary
    - Post-Quantum Cryptography
        - A field focused on developing cryptographic methods that withstand quantum computing attacks
    - Quantum Computing Threat
        - Quantum computers' ability to solve complex mathematical problems rapidly endangers traditional encryption methods
    - PQC vs Traditional Cryptography
        - Diffie-Hellman and ECC
            - Vulnerable to quantum attacks due to the solvability of their underlying math by quantum algorithm
        - Quantum-Resistant Algorithms
            - Built on problems that are computationally infeasible for quantum computers
    - Future-Proofing with PQC
        - New cryptographic methods are critical for maintaining security as quantum technology advances

- **Post-Quantum Implications**
  - Post-Quantum Implications
    - Challenges and adaptations required to secure data against the advanced decryption capabilities of quantum computers, involving quantum-resistant cryptographic algorithms and new secure implementations
  - Resistance to Quantum Computing Decryption Attacks
    - Definition
      - Development of cryptographic algorithms that can withstand the powerful decryption techniques enabled by quantum computers
    - Key Approaches
      - Lattice-Based Cryptography
        - Utilizes multi-dimensional lattice structures that are computationally intensive for quantum computers to solve
        - Provides strong security and versatility for encryption, digital signatures, and key exchange protocols
      - Hash-Based Cryptography
        - Builds security on the difficulty of reversing cryptographic hash functions, creating digital signatures resistant to quantum attacks
        - Example
          - Producing a unique hash for input data makes reversing the process computationally challenging, even for quantum systems
      - Multivariate Polynomial Cryptography

            - Based on solving complex systems of nonlinear equations with multiple variables
            - Presents a multi-variable "riddle" resistant to quantum decryption due to complex dependencies, making it impractical for quantum attack
- Emerging Implementations
    - Code-Based Cryptography
        - Relies on the complexity of decoding random linear codes, a problem that quantum computers cannot solve efficiently
        - Example
            - The McEliece cryptosystem, known for its resistance to quantum attacks and suitability for large-scale applications
        - Challenge
            - Optimization for practical use, as it often requires larger key sizes
    - Quantum Key Distribution (QKD)
        - Uses quantum mechanics to securely exchange encryption keys via qubits, alerting parties to any eavesdropping attempts by altering qubit states
        - Theoretical immunity to quantum attacks, making it a highly secure method for key exchange
        - Applications
            - Already in testing for high-stakes sectors like finance and government communications, representing a secure key exchange future
- Examples of Quantum-Resistant Techniques

        - Lattice-Based Cryptography
            - Application
                - Secure encryption, digital signatures, and key exchange protocols
            - Advantage
                - Hard for quantum computers to break due to multi-dimensional complexity
        - Quantum Key Distribution (QKD)
            - Application
                - Secure key exchanges in critical environments
            - Advantage
                - Immunity to quantum attacks due to the inherent properties of qubits
    - Summary
        - Post-Quantum Implications
            - The need to adapt current cryptographic practices to resist quantum decryption capabilities
        - Resistance to Quantum Decryption
            - Involves lattice-based, hash-based, and multivariate polynomial cryptography, which present complex challenges to quantum computers
        - Emerging Implementations
            - Code-Based Cryptography
                - Secure but requires larger key sizes for practical application
            - Quantum Key Distribution (QKD)

    - A revolutionary key exchange method based on quantum principles, immune to quantum attacks

Troubleshooting IAM

Objective 3.1: Troubleshoot common issues with identity and access management components in an enterprise environment

- **Management Frameworks**
    - Management Frameworks
        - Methods used to manage, monitor, and maintain identities and access controls in an organization, ensuring secure handling of authentication, authorization, and privileged access
    - Identity Proofing
        - Definition
            - The process of verifying an individual's claimed identity to ensure accuracy and legitimacy
        - Application
            - Initial Setup & Password Resets
                - Used during user onboarding or for verifying identity during password resets
            - Example
                - Security questions like "Where did you go to high school?" enhance security but are vulnerable if answers are publicly accessible
            - Best Practice
                - Use an alternative persona or unique answers for security questions that cannot easily be guessed
        - Verification Methods

        - Documentation Verification
            - Presenting photo ID or passport at an IT service desk for in-person verification
        - Identity Propagation
            - Once verified, identity details are passed on to other systems, especially in environments with Single Sign-On (SSO)
        - Example Systems
            - Kerberos (Microsoft Active Directory)
                - Uses ticketing to manage identity and authentication across services
            - Credential Security Support Provider Protocol (CredSSP)
                - Allows credential transmission across the network and should operate over encrypted connections (e.g., SSL/TLS)
- Privileged Identity Management (PIM)
    - Definition
        - Managing and controlling access to critical systems by granting elevated permissions only as needed
    - Principle
        - Users receive elevated permissions temporarily, only when necessary for specific tasks
    - Example
        - Temporary Elevated Access
            - An IT admin requests access to change security settings, and PIM grants elevated permissions for a limited time

- Access Removal Post-Task
  - After task completion, elevated access is revoked to reduce security risks
- Monitoring & Accountability
  - Audit Trail
    - Logs all privileged actions for tracking and accountability
  - Example
    - Any changes made by an administrator to critical systems are recorded, enabling quick response to unauthorized or suspicious activity
- Summary
  - Management Frameworks
    - Securely handle identities and access control by verifying users, managing permissions, and monitoring privileged actions
  - Identity Proofing
    - Confirms an individual’s identity, often required at account setup or for password resets, using both knowledge-based and documentation-based methods
  - Privileged Identity Management (PIM)
    - Grants elevated access only as needed and tracks actions to minimize risks and enforce accountability
  - Core Components
    - Authentication
      - Verifies a user's identity
    - Authorization
      - Determines accessible resources for each user

        - Identity Propagation
            - Shares verified identity information across networked systems to streamline access management

- **Subject Access Control**
    - Subject Access Control
        - Mechanisms that determine and enforce access permissions for specific resources within a network, based on the roles and identities of users, processes, devices, and services
    - Users
        - Definition
            - Individuals or entities requiring access to system resources
        - Example
            - An employee logging into a network to access email or files
        - Role-Based Access
            - Purpose
                - Ensures each user only has access to the resources needed for their specific role
            - Example
                - An accountant accesses financial records, while a developer does not
            - Benefit
                - Prevents unauthorized access, securing sensitive data by limiting visibility to essential resources
    - Processes
        - Definition

            - Instances of programs operating on behalf of users or system functions
        - Example
            - A web browser running on a computer
        - Access Control Role
            - Purpose
                - Manages permissions for processes, allowing them to perform only necessary and safe actions
            - Security Control
                - Prevents unauthorized access or potentially harmful actions, such as malware attempting to access restricted data
    - Devices
        - Definition
            - Physical or virtual hardware connecting to a network, such as laptops, smartphones, and virtual machines
        - Example
            - A work laptop connecting to office Wi-Fi
        - Authentication Requirement
            - Purpose
                - Ensures devices are authorized to access resources, requiring up-to-date security compliance
            - Example
                - A device must pass security checks before gaining network access
            - Benefit

- Protects the network by blocking unauthorized or compromised devices

- Services
  - Definition
    - System functions or applications that operate continuously, managing tasks without direct user input
  - Example
    - Email servers, file-sharing applications, database management systems
  - Access Control Role
    - Purpose
      - Defines what resources each service can access and what actions it may perform
    - Example
      - An email server may handle messages but lacks access to sensitive financial records
    - Benefit
      - Maintains operational integrity and prevents unauthorized access to critical data
- Summary
  - Subject Access Control
    - Mechanism to manage and enforce access permissions based on subject roles in a network, including users, processes, devices, and services
  - Users

            - Granted permissions based on roles, with limited access to prevent unauthorized information exposure
        - Processes
            - Running programs that have controlled access to perform only necessary tasks
        - Devices
            - Both physical and virtual hardware must authenticate to access network resources
        - Services
            - Background operations with defined access limits to secure interactions within the network

- **User Identity Control**
    - User Identity Control
        - The process of managing, verifying, and securing user identities to ensure appropriate access to resources
    - Credentials
        - Definition
            - Information that users provide to verify their identity, such as passwords, PINs, and digital certificates
        - Examples
            - Passwords
                - Simple and widely used but vulnerable to compromise
            - Digital Certificates
                - Use cryptography to verify identity and provide higher security, often for access to sensitive resources

        - Challenges
            - Weak Passwords
                - Easily guessed or cracked, especially if reused
            - Exposure
                - Risk of passwords being stolen in data breaches
    - Biometrics
        - Definition
            - Use of unique physical or behavioral traits, like fingerprints or facial recognition, to confirm identity
        - Types
            - Fingerprints
                - Unique to each individual and commonly used for smartphone access
            - Facial Recognition
                - Scans facial features and compares them to stored templates
        - Metrics
            - Crossover Error Rate (CER)
                - Balances False Acceptance Rate (FAR) and False Rejection Rate (FRR) to measure system accuracy
            - False Acceptance Rate (FAR)
                - Rate of incorrectly accepted unauthorized individuals
            - False Rejection Rate (FRR)
                - Rate of wrongly rejecting authorized users
    - Multi Factor Authentication (MFA)
        - Definition

        - Authentication process requiring two or more verification factors to confirm identity
    - Factors
        - Something You Know
            - Information like passwords or PINs
        - Something You Have
            - Physical items like a smart card or smartphone
        - Something You Are
            - Biometric data, such as fingerprints
        - Somewhere You Are
            - Location data, verified through GPS or IP address
    - Implementation
        - Single-Factor Authentication
            - Uses only one factor, like a password
        - Two-Factor Authentication (2FA)
            - Combines two factors, such as a PIN and a smart card
        - MFA
            - Uses multiple factors, such as a smart card, a biometric scan, and location verification for enhanced security
    - Advanced Options
        - Time-based One-Time Passwords (TOTP)
            - Single-use codes that change frequently
        - HMAC-based One-Time Passwords (HOTP)
            - Codes generated by hashing algorithms, typically out-of-band for added security
- Summary

        - User Identity Control
            - Involves verifying and securing user identities to control access to resources
        - Credentials
            - Basic verification information, like passwords or certificates, vulnerable to compromise if not managed securely
        - Biometrics
            - Uses physical traits like fingerprints to verify identity, providing enhanced security
        - Multi Factor Authentication (MFA)
            - Combines multiple authentication factors to increase security, preventing unauthorized access

- **Secrets Management**
    - Secrets Management
        - The secure storage, management, and control of sensitive information to prevent unauthorized access
        - Key components include tokens, certificates, passwords, keys, rotation, and deletion
    - Tokens
        - Definition
            - Temporary digital credentials for authentication or authorization, often used in web applications and APIs
        - Example
            - Session tokens keep a user logged in without repeated credentials entry

        - Types
            - JSON Web Tokens (JWTs):
                - Transmit user roles and permissions securely
            - OAuth Tokens
                - Delegate access between applications, such as logging into a third-party app with a Google account
        - Security
            - Designed to expire after a short period to reduce misuse risks, with options for secure storage, expiration, and revocation
    - Certificates
        - Definition
            - Digital documents verifying the identity of users, devices, or websites, enabling secure communication
        - Example
            - SSL/TLS certificates secure connections between browsers and websites
        - Types
            - SSL/TLS Certificates
                - Secure website connections
            - Code-Signing Certificates
                - Validate software authenticity
            - Client Certificates
                - Authenticate users within corporate systems
        - Security
            - Managed for timely updates, revocation, and proper configuration to maintain trust

- Passwords
  - Definition
    - Secret character strings used for user authentication across applications and systems
  - Security
    - Storage
      - Hashing passwords to prevent exposure
    - Rotation
      - Updating periodically to reduce risks of exposure
  - Best Practices
    - Use salting, two-factor authentication, and monitoring for compromised credentials
- Keys
  - Definition
    - Cryptographic elements for encryption and decryption, crucial for securing data
  - Types
    - Public/Private Keys
      - Used in Public Key Infrastructure (PKI) for data security
    - Symmetric Keys
      - Common for data encryption
  - Security Issues
    - Improper Handling
      - Secure keys during storage to prevent compromise
    - Rekeying
      - Regular session rekeying strengthens encryption over time

- Rotation
  - Definition
    - Regularly updating or replacing sensitive credentials (keys, passwords, tokens) to maintain security
  - Purpose
    - Minimizes exposure risk by limiting credential lifespan
  - Process
    - Involves re-encrypting data, updating access controls, and syncing systems with new keys
- Deletion
  - Definition
    - Securely destroying secrets that are no longer needed to prevent unauthorized access
  - Methods
    - Cryptographic Shredding
      - Destroys decryption keys, making data permanently unreadable
  - Example
    - AES-encrypted data on a hard drive becomes inaccessible when its key is securely shredded
- Summary
  - Secrets Management
    - Secure handling of sensitive information to prevent unauthorized access, including tokens, certificates, passwords, and cryptographic keys
  - Tokens

            - Temporary access credentials with short lifespans to limit misuse
        - Certificates
            - Identity-verifying documents used in secure communications
        - Passwords
            - Require secure storage (hashing) and periodic rotation for added security
        - Keys
            - Used in encryption, requiring careful handling and periodic rekeying
        - Rotation
            - Regularly updates credentials to reduce exposure risk
        - Deletion
            - Securely removes unneeded secrets, ensuring permanent data protection

- **Authentication and Authorization**
    - Authentication and Authorization
        - The processes that verify a user's identity and determine their access rights to resources within a system
        - Key components include attestation, SAML, OpenID, OAuth, and federation
    - Attestation
        - Definition
            - Verifies that a user, system, or device meets specific security standards and policies before granting access
        - Example

        - Ensuring that a user's security token complies with organizational policies before allowing access to a resource
    - How it Works
        - Uses digital signatures or hash values to confirm the integrity of a user's credentials
- Security Assertion Markup Language (SAML)
    - Definition
        - A standard that facilitates the exchange of authentication and authorization data, often used in Single Sign-On (SSO) environments
    - Example
        - An employee logs in once to the company's identity provider and gains access to multiple internal applications without re-entering credentials
    - How it Works
        - Uses XML-based tokens to pass authentication data securely between an identity provider and service providers, simplifying the login process
- OpenID
    - Definition
        - An authentication protocol that enables users to log in to multiple services using a single identity managed by an identity provider
    - Example
        - Logging into various websites using Google credentials, with Google acting as the identity provider
    - Benefits

      - Enhances user experience by reducing the need to remember multiple passwords, improving security
- OAuth (Open Authorization)
  - Definition
    - An authorization framework that allows third-party applications to access user data without handling authentication
  - Example
    - A third-party app accesses specific social media data to post photos without needing the user's credentials
  - How it Works
    - Issues tokens instead of passwords, limiting access to specific data and actions while protecting user credentials
- Federation
  - Definition
    - The practice of managing user identities across multiple domains, allowing users to access resources based on authentication from a trusted domain
  - Example
    - A student accesses resources at another university campus using their home campus credentials
  - How it Works
    - Often uses SAML to exchange authentication and authorization data between trusted domains, enabling efficient access management
- Summary
  - Authentication and Authorization

            - Processes that verify a user's identity and define their access rights within a system
        - Attestation
            - Confirms a user's credentials meet security standards, ensuring legitimacy
        - SAML
            - Enables SSO by securely passing authentication data between identity providers and service providers
        - OpenID
            - Simplifies logins by allowing users to access multiple services with one set of credentials managed by an identity provider
        - OAuth
            - Manages authorization by providing token-based access to specific data for third-party applications
        - Federation
            - Allows identity sharing across multiple domains, enabling users to access services across trusted domains with a single identity

- **Cloud IAM access and Trust Policies**
    - Cloud IAM Access and Trust Policies
        - Policies used in cloud environments to control and enforce user, group, and service access to cloud resources, defining actions and establishing trust boundaries between accounts and services
    - Cloud IAM Access Policies
        - Definition

        - Rules that manage which users, groups, and services have access to specific cloud resources and what actions they are permitted to perform
      - How it Works
        - Defines permissions for actions like reading, modifying, or deleting data within cloud environments, ensuring only authorized users can execute specific tasks
      - Examples of Access Policies in Major Cloud Platforms
        - AWS
          - Uses JSON documents associated with users, groups, or roles to define permissions
        - Azure
          - Uses Role-Based Access Control (RBAC) to assign access roles to users and groups
        - Google Cloud Platform (GCP)
          - Directly assigns permissions to users, groups, and services via IAM roles
      - Real-World Example
        - Granting a data analyst read-only access to a data warehouse, preventing modifications or deletions to maintain data security
    - Cloud IAM Trust Policies
      - Definition
        - Policies that define trusted relationships between cloud accounts, services, or external entities, specifying which entities are allowed to assume roles or access resources
      - How it Works

        - Used to control access between separate cloud accounts or services within the same cloud provider by establishing trust boundaries and permissions for cross-account or cross-service operations
    - Examples of Trust Policies in Major Cloud Platforms
        - AWS
            - Trust policies are part of IAM roles, enabling one account to assume roles in another for specific permissions
        - Azure
            - Establishes trust via service principals and managed identities for secure cross-service interactions
        - GCP
            - Configures trust boundaries between projects and services, allowing secure access between environments
    - Real-World Example
        - A Lambda function in one AWS account accesses resources in another account by assuming a role via a trust policy, allowing necessary permissions for cross-account tasks
- Summary
    - Cloud IAM Access and Trust Policies
        - Control access and define relationships within cloud environments, managing permissions for users, groups, and services
    - Access Policies
        - Specify roles and permissions, managing what actions users and services can perform within the cloud

        - Trust Policies
            - Define trust boundaries and relationships between cloud accounts or services, enabling secure cross-account or cross-service access
        - How They Work Together
            - Access policies set permissions, while trust policies enable secure sharing of these permissions across accounts and services

- **WiFi Authentication**
    - Wi-Fi Authentication
        - Methods used to manage and secure user access to wireless networks, including protocols like IEEE 802.1X, the Extensible Authentication Protocol (EAP), and the Simultaneous Authentication of Equals (SAE)
    - IEEE 802.1X
        - Definition
            - A network access control protocol that authenticates devices attempting to connect to a network, allowing access only to authorized users
        - How it Works
            - Uses an authentication server (e.g., RADIUS server) to verify user credentials
            - Begins the authentication process by initiating credential verification when a device (supplicant) tries to connect
        - Role of EAP
            - Integrates with 802.1X to offer flexible, secure credential exchanges
        - Example

        - An employee’s laptop requires credential verification via 802.1X and EAP before accessing the corporate Wi-Fi network
- Extensible Authentication Protocol (EAP)
    - Definition
        - A flexible framework supporting multiple authentication methods for secure communication between a supplicant and the network
    - Older Authentication Protocols
        - Password Authentication Protocol (PAP)
            - Transmitted credentials in plain text, vulnerable to interception
        - Challenge Handshake Authentication Protocol (CHAP)
            - Improved security by using a challenge-response mechanism
    - Common EAP Types
        - EAP-MD5 CHAP
            - Basic security with passwords, vulnerable to dictionary attacks
        - EAP-TLS
            - Uses digital certificates for high-security mutual authentication
        - PEAP and EAP-TTLS
            - Establish encrypted tunnels; secure but flexible options
        - EAP-FAST
            - Uses Protected Access Credentials, faster but slightly less secure
    - Preferred EAP Method

            - EAP-TLS for high security, or PEAP/EAP-TTLS for simplified implementations
    - Simultaneous Authentication of Equals (SAE) (Used in WPA3)
        - Definition
            - A password-based key exchange protocol in WPA3 that provides secure handshakes resistant to offline dictionary attacks
        - How it Works
            - Forward Secrecy
                - Ensures that past communications remain secure even if long-term keys are compromised
            - Dragonfly Algorithm
                - Uses elliptic curve cryptography for secure key exchanges, creating unique session keys for each session
        - Example
            - Each time a user connects to a WPA3 Wi-Fi network, SAE creates a unique session key, protecting data from interception and preserving security across sessions
    - Summary
        - Wi-Fi Authentication
            - Manages and secures network access, verifying devices and users before allowing network entry
            - IEEE 802.1X
                - Provides a framework for authenticating devices, using an authentication server to ensure only authorized access
            - Extensible Authentication Protocol (EAP)

            - Offers a variety of secure authentication methods, replacing outdated protocols like PAP and CHAP
        - Simultaneous Authentication of Equals (SAE)
            - Provides WPA3's secure key exchange, ensuring forward secrecy and unique session keys
    - Overall Goal
        - Robust Wi-Fi security by verifying identities, securely exchanging credentials, and protecting communications through adaptive, secure authentication

- **Access Control**
    - Access Control
        - Security measures that manage and enforce who can access specific resources within a system, based on predefined policies and user roles
        - Key components of access control include Single Sign-On (SSO), Kerberos, and Privileged Access Management (PAM)
    - Single Sign-On (SSO)
        - Definition
            - An authentication method allowing users to access multiple applications or services with one set of login credentials
        - Purpose
            - Simplifies user experience by reducing the need for multiple usernames and passwords and increases security by minimizing password-related breaches
        - Example

- An employee logs into the company portal and automatically gains access to email, HR systems, and other applications without separate logins

- Kerberos
  - Definition
    - A network authentication protocol that uses a ticket-based system to securely prove identities over a non-secure network
  - Purpose
    - Provides secure identity verification for users and services without transmitting passwords over the network
  - Key Concepts
    - Ticket-Granting Ticket (TGT)
      - Issued after initial authentication, allowing access to various services
    - Service Ticket
      - Presented by users to access specific services, validated by the service for authenticity
    - Mutual Authentication
      - Both the user and the service verify each other’s identity for added security
  - Example
    - A user in a Microsoft Active Directory environment accesses a shared file; Kerberos verifies their identity through a TGT and service ticket
- Privileged Access Management (PAM)
  - Definition

            - A security strategy to control and monitor access to critical systems by managing elevated user privileges
        - Purpose
            - Ensures that only authorized users gain elevated access to sensitive resources and reduces the risks associated with excessive permissions
        - Key Features
            - Just-in-Time Access
                - Grants elevated access only when needed for specific tasks
            - Continuous Monitoring
                - Tracks privileged activities to detect and respond to suspicious actions
            - Automated Security Controls
                - Includes features like password rotation, session management, and multi-factor authentication
        - Example
            - PAM solutions manage, audit, and monitor administrator accounts to prevent unauthorized access and ensure secure privileged actions
    - Summary
        - Access control secures system resources by regulating who can access what, based on policies and user roles
        - Key elements include
            - Single Sign-On (SSO)
                - Simplifies access by allowing users to authenticate once to access multiple services

        - Kerberos
            - A secure, ticket-based protocol for authenticating over non-secure networks
        - Privileged Access Management (PAM)
            - Manages elevated privileges to control access to critical systems and reduce security risks
    - Each method provides unique protection and usability benefits, ensuring a secure and efficient access management system for users and administrators

- **Conditional Access**
    - Conditional Access
        - A security approach that grants or denies access to resources based on specific conditions, such as user identity, device status, time, and location
    - Configuration
        - Definition
            - The process of setting up and customizing access policies to align with an organization's security needs
        - How it Works
            - Involves defining rules and parameters for granting or denying access based on roles, device compliance, and resource sensitivity
            - Includes setting up exceptions and exclusions for scenarios like emergency access to maintain business continuity
        - Example

            - Allowing access to a financial application only for finance department employees using company devices on a corporate network
        - Ongoing Process
            - Regular reviews and updates are needed to adapt to evolving security threats and business changes
    - User-to-Device Binding
        - Definition
            - A security measure that links a user's identity to a specific, approved device, ensuring access is only granted from trusted devices
        - Purpose
            - Prevent unauthorized access from unapproved or compromised devices
        - Implementation
            - Requires inventory management of approved devices and enforcement through tools like Mobile Device Management (MDM) for compliance
        - Example
            - Restricting access to internal systems exclusively from company-issued laptops, blocking access even if user credentials are stolen
    - Time-Based Conditions
        - Definition
            - Restrict access based on specific times, allowing access only within predefined time windows

        - Purpose
            - Helps prevent unauthorized access during off-hours when monitoring may be reduced
        - Example
            - Limiting access to internal resources like a dashboard only from 8 AM to 6 PM, automatically denying access outside these hours
        - Usage
            - Tailorable for different user groups (e.g., IT staff on-call); provides data on access patterns to help identify unusual activity
    - Geographic Location Conditions
        - Definition
            - Restrict access based on the physical location of the user, using IP addresses to verify if access is coming from an approved region
        - Purpose
            - Helps prevent unauthorized access from high-risk regions or areas outside the organization's normal zones
        - Example
            - Restricting access to data only within the United States, automatically blocking access attempts from foreign countries
        - Benefits
            - Useful for remote workforces; helps organizations meet regional compliance needs and detect potential threats based on access location
    - Summary
        - Conditional Access

            - Manages access to resources based on conditions that align security measures with organizational needs
            - Configuration
                - Defines access rules based on user roles, device compliance, and resource sensitivity; requires regular updates
            - User-to-Device Binding
                - Links a user's identity to specific, trusted devices, reducing the risk of unauthorized device access
            - Time-Based Conditions
                - Limits access to certain times, reducing off-hours risks and helping to prevent unauthorized activity
            - Geographic Location Conditions
                - Blocks access based on user location, protecting against unauthorized access from unapproved regions
        - Overall Goal
            - Strengthen security by using dynamic access policies that consider various contextual factors

- **Logging and Monitoring**
    - Logging and Monitoring
        - The processes of recording and analyzing access events and user activities to detect, investigate, and respond to potential security incidents
    - Logging
        - Definition

        - Capturing and recording detailed information about user actions, system changes, and access attempts within a system or network
    - Purpose
        - Provides a record of activity, crucial for audits, investigations, and compliance
    - Components
        - Log Collectors
            - Systems that centralize logs from multiple sources, simplifying management and enhancing visibility (e.g., Fluentd)
        - Security Information and Event Management (SIEM)
            - Aggregates logs from various sources and normalizes them, converting diverse log formats into a consistent structure to enable effective analysis
    - Requirements
        - Secure Storage and Retention
            - Logs must be securely stored and retained per regulatory standards
        - Accessibility
            - Logs should be easily accessible for audits and investigations
- Monitoring
    - Definition
        - Continuously reviewing and analyzing logs in real-time or scheduled intervals to detect unusual activities that may signal security threats

        - Goal
            - Identifies indicators of compromise (IOC) by detecting anomalies like repeated failed login attempts or unauthorized access
        - Challenges
            - The large volume of log data requires automation to detect abnormal patterns and generate alerts efficiently
        - Tools
            - SIEM Systems
                - Provide real-time alerts, dashboards, and data correlation to detect sophisticated attacks across systems
            - Intrusion Detection Systems (IDS)
                - Monitor network traffic for suspicious behaviors, generating alerts when unusual patterns emerge
            - Dedicated Log Monitoring Solutions
                - Tools like Graylog and SolarWinds support real-time network activity tracking and alerting
    - Summary
        - Logging
            - Captures and records detailed information about user actions, system changes, and access attempts, creating a record of network activity for investigation and compliance
        - Monitoring
            - Reviews logs actively to identify unauthorized or unusual activities, often using automated tools to detect indicators of compromise quickly
        - SIEM Systems

    - Essential for logging and monitoring, SIEMs aggregate and normalize log data and provide real-time alerts and dashboards for detecting threats across multiple systems

- Intrusion Detection Systems (IDS) and Log Monitoring Solutions
    - Provide additional layers of monitoring to detect and respond to network-based threats

Troubleshooting Network Infrastructure

Objective 3.3: Troubleshoot complex network infrastructure security issues

- **Observability**
  - Observability
    - The ability to monitor, understand, and diagnose the internal states and performance of a system using the data it generates
  - Monitoring
    - Definition
      - Continuously collecting data from various sources, such as network traffic, error rates, and system logs
    - Purpose
      - To provide a real-time view of system performance and detect abnormal behavior before issues escalate
    - Examples
      - Traffic Patterns
        - Monitoring for unusual spikes that could indicate potential security threats
      - Error Rates
        - Tracking application errors to spot performance bottlenecks or malfunctions
      - System Logs
        - Capturing logs from servers and applications to have a detailed activity trail for analysis

- Analysis
  - Definition
    - Processing collected data to gain insights into system behavior and identify deviations from normal operation
  - Purpose
    - To understand patterns within system data, highlight anomalies, and identify potential issues
  - Examples
    - Failed Login Attempts
      - Identifying an unusual increase, which may indicate a brute force attack
    - Traffic Spikes
      - Analyzing sudden increases in network traffic, which could signal a potential denial-of-service attack
    - Baseline Behavior
      - Establishing a baseline of normal activity to detect deviations that signal potential issues
- Diagnosis
  - Definition
    - Locating the exact source of an issue to enable targeted troubleshooting and efficient resolution
  - Purpose
    - To identify the root cause of problems, enabling faster and more precise troubleshooting
  - Examples
    - Server Downtime

                - Using observability tools to determine if the issue is due to hardware failure, misconfiguration, or a security breach
            - Application Slowness
                - Identifying whether the slowdown is due to code issues, database performance, or network bottlenecks
    - Summary
        - Observability
            - Enables organizations to monitor, analyze, and diagnose system performance, providing deep insights into internal system operations
        - Monitoring
            - Collects real-time data on system performance, capturing essential metrics like traffic patterns and error rates
        - Analysis
            - Processes collected data to identify patterns, detect anomalies, and understand system behavior
        - Diagnosis
            - Locates the root cause of issues for targeted troubleshooting, reducing time to resolution

- **Network Errors**
    - Network Errors
        - Issues disrupting the normal flow of data across a network, often due to misconfigurations, hardware malfunctions, or software bugs
    - Switching Errors
        - Definition

            - Errors that occur when data packets are misdirected or dropped within network switches
        - Common Types
            - Broadcast Storms
                - Cause
                    - Layer 2 network loop, often due to improper Spanning Tree Protocol (STP) configuration
                - Impact
                    - Floods the network, causing packet loss, device crashes, and network downtime
            - VLAN Mismatch
                - Cause
                    - Different VLAN settings on connected switches
                - Impact
                    - Traffic dropped or misrouted, isolating devices and disrupting communication
            - Duplex Mismatch
                - Cause
                    - Mismatched duplex settings (full vs. half-duplex) on a network link
                - Impact
                    - Leads to collisions, packet loss, and slow network performance
    - Routing Errors
        - Definition

- Errors in data transmission paths caused by misconfigurations or network instability
  - Common Types
    - Routing Loops
      - Cause
        - Misconfigured routing protocols or improper route advertisements
      - Impact
        - Packet gets stuck in a continuous loop, never reaching its destination
    - Incorrect Subnet Masks
      - Cause
        - Misconfigured subnet masks in IP addresses
      - Impact
        - Incorrect routing, causing connectivity issues within and between subnets
    - Route Flapping
      - Cause
        - Network route frequently changing state due to unstable links or faulty hardware
      - Impact
        - Network instability, increased latency, and dropped packets
- VPN and Tunnel Errors
  - Definition

            - Disruptions in secure communication tunnels due to misconfigurations, expired credentials, or encryption mismatches
        - Common Types
            - Incorrect Settings
                - Cause
                    - Misconfiguration of critical parameters such as IP addresses and DNS
                - Impact
                    - Failed or unreliable VPN connections, disrupting remote access
            - Expired Certificates
                - Cause
                    - Outdated certificates used for mutual authentication
                - Impact
                    - Authentication failures, breaking secure communication channels
            - Encryption Protocol Mismatches
                - Cause
                    - Incompatible or outdated encryption protocols between VPN endpoints
                - Impact
                    - Failure to establish a secure tunnel, exposing data to potential interception
    - Summary
        - Network Errors

            - Disrupt data flow and are typically caused by misconfigurations, hardware issues, or software bugs
        - Switching Errors
            - Include broadcast storms, VLAN mismatches, and duplex mismatches, causing misdirection or packet loss
        - Routing Errors
            - Include routing loops, incorrect subnet masks, and route flapping, leading to inefficient data transmission
        - VPN and Tunnel Errors
            - Caused by incorrect settings, expired certificates, or encryption protocol mismatches, resulting in failed secure connections

- **Network Misconfigurations**
    - Network Misconfigurations
        - Errors or incorrect settings in network devices that can lead to performance issues, security vulnerabilities, or connectivity failures
    - Insecure Routing
        - Definition
            - Occurs when routing protocols or configurations lack security measures, leaving the network vulnerable to attacks
        - Key Vulnerabilities
            - Lack of Authentication
                - Impact
                    - Allows attackers to inject false routing information, leading to route hijacking or route spoofing
                - Example

            - An attacker exploiting a BGP router lacking authentication to advertise false routes, redirecting traffic to unauthorized locations
    - Outdated Protocols
        - Impact
            - Protocols like RIP lack encryption and authentication, making routing data vulnerable to manipulation
        - Example
            - An attacker manipulating RIP routes to intercept or reroute sensitive data
    - Lack of Encryption
        - Impact
            - Allows attackers to intercept and modify routing data sent over public networks
        - Solution
            - Use IPSec to secure routing updates over untrusted networks, adding protection against redirection or interception
- Preventing Insecure Routing
    - Implement authentication and encryption in routing protocols (e.g., using OSPF or BGP with authentication)
    - Regularly review and update routing configurations, replacing outdated protocols with secure alternatives
    - Disable weak or deprecated protocols to reduce attack surfaces
- Configuration Drift

- Definition
  - The gradual and often undocumented changes in network configurations over time, leading to potential inconsistencies and security vulnerabilities
- Common Causes
  - Manual Configuration Changes
    - Impact
      - Undocumented adjustments accumulate, causing inconsistent settings across devices
    - Example
      - A manually adjusted ACL on one router is not applied across the network, resulting in inconsistent security policies
  - Stability and Performance Issues
    - Impact
      - Drifted configurations may cause network instability, performance degradation, or outages
    - Example
      - A server's load-balancing configuration drifts, resulting in some servers being overburdened while others are underutilized
- Combating Configuration Drift
  - Use automated configuration management tools (e.g., Cisco NSO, Ansible) to monitor and correct unauthorized changes
  - Perform regular configuration backups and audits to detect and resolve drift before it impacts performance

        - Maintain consistent configurations to ensure reliability, security, and efficiency
  - Summary
    - Network Misconfigurations
      - Can lead to security risks, connectivity issues, or performance degradation
    - Insecure Routing
      - Arises from lack of security in routing protocols, including insufficient authentication, use of outdated protocols, and absence of encryption
    - Configuration Drift
      - Occurs when settings gradually change over time, often due to undocumented manual changes, leading to inconsistencies and vulnerabilities
    - Mitigation
      - Requires regular reviews, secure protocols, automated management, and consistent documentation to ensure network stability and security

- **IPS/IDS Issues**
  - IPS and IDS Issues
    - Challenges in the deployment, configuration, and effectiveness of Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), including placement, lack of rules, and rule misconfigurations
  - Placement
    - Definition

        - Determines where the IDS or IPS is positioned within the network to maximize effectiveness
    - IDS Placement
        - Objective
            - Monitor network traffic passively without affecting flow
        - Placement
            - Typically on a switch mirror port or SPAN port to analyze traffic without interference
        - Risk
            - Poor placement may result in detection of irrelevant traffic, causing false positives
    - IPS Placement
        - Objective
            - Actively monitor and block suspicious traffic in real time
        - Placement
            - Must be in-line with traffic to take immediate action (e.g., dropping packets or terminating connections)
        - Risk
            - Ineffective placement can allow malicious traffic to bypass IPS protection
    - Optimized Placement
        - Involves understanding network traffic flow and identifying high-risk areas to ensure effective monitoring and prevention
- Lack of Rules
    - Definition

- Absence of sufficient policies or signatures that identify malicious behavior
- IDS
  - Impact
    - Incomplete rules limit detection capabilities, allowing advanced threats to go unnoticed
  - Example
    - Missing rules for advanced persistent threats (APTs) may result in undetected intrusions
- IPS
  - Impact
    - Insufficient rules reduce the ability to proactively block threats, allowing malicious traffic through
  - Example
    - Without updated rules, an IPS may miss new malware or attack vectors
- Mitigation
  - Regular rule updates to cover evolving threats and diverse attack patterns ensure effective detection and prevention

- Rule Misconfigurations
  - Definition
    - Incorrectly set rules that impact the balance between detecting threats and minimizing false alerts
  - IDS
    - Impact

            - Overly aggressive rules result in excessive false positives, overwhelming security teams
        - Example
            - Minor deviations triggering alerts may flood teams with harmless events like routine software updates
    - IPS
        - Impact
            - Overly strict configurations may block legitimate applications, disrupting business operations
        - Example
            - Blocking encrypted traffic essential for critical applications due to aggressive filtering
    - Signature Updates
        - Essential to detect new threats, as outdated signatures may fail to recognize advanced threats
    - Mitigation
        - Regular audits, automated rule updates, and configurations tailored to traffic patterns can prevent misconfigurations and ensure system effectiveness
- Summary
    - IPS and IDS Issues
        - Arise from improper placement, lack of rules, and misconfigured rules
    - Placement
        - Incorrect positioning of IDS and IPS can lead to missed threats or an overload of false alerts

        - Lack of Rules
            - Incomplete or outdated rule sets leave gaps in threat detection and prevention
        - Misconfigurations
            - Incorrect thresholds or outdated signatures may result in too many alerts or missed threats
        - Best Practices
            - Regular rule updates, proper placement, and consistent audits ensure optimal IPS and IDS performance

- **Alert Analysis**
    - Alert Analysis
        - The process of reviewing and interpreting security alerts to distinguish between genuine threats and benign activity
        - This includes identifying and managing false positives and false negatives
    - False Positives
        - Definition
            - Occurs when an Intrusion Prevention System (IPS) or Intrusion Detection System (IDS) incorrectly flags benign activity as a security threat
        - Impact
            - Can overwhelm security teams with unnecessary alerts, leading to wasted time and resources and potentially diverting focus from genuine threats
        - Example

        - An IDS incorrectly alerts on authorized internal network scans as threats
    - Management Strategies
        - Rule Tuning
            - Adjusting detection rules to reduce noise while maintaining threat visibility
        - Allowlisting
            - Creating rules to allow trusted IP addresses, ports, or traffic patterns
        - Threshold Adjustments
            - Setting thresholds to flag unusual patterns, such as high data transfer rates, without triggering for routine activities
        - Regular Updates
            - Keeping rules current and training analysts to distinguish between false positives and true positives
- False Negatives
    - Definition
        - Occurs when an IPS or IDS fails to detect an actual security threat, allowing malicious activity to go unnoticed
    - Impact
        - Creates a false sense of security, allowing attackers to operate undetected, leading to potential data theft, lateral movement, or system disruption
    - Example
        - An IDS or IPS fails to detect malware or lateral movement due to reliance on outdated threat signatures

        - Mitigation Strategies
            - Multi-Layered Detection
                - Combining signature-based detection, behavioral analysis, and anomaly detection
            - Behavioral Analysis
                - Identifying deviations from normal network activity, which can highlight suspicious activity even without specific threat signatures
            - Anomaly Detection
                - Using machine learning to flag irregular patterns outside of established baselines
            - Additional Tools
                - Using complementary security tools (e.g., firewalls, antivirus, endpoint detection) for layered defense
            - Threat Intelligence Integration
                - Incorporating real-time threat intelligence to enhance detection capabilities
    - Summary
        - Alert Analysis
            - Distinguishes real security threats from benign activity by managing false positives and false negatives
        - False Positives
            - Incorrectly flagging harmless activity; leads to alert fatigue and wasted resources
        - False Negatives

            - Missing real threats; poses a greater security risk by allowing undetected malicious activity
        - Best Practices
            - Reducing False Positives
                - Tuning rules, allowlisting trusted traffic, and setting appropriate alert thresholds
            - Detecting False Negatives
                - Using multi-layered detection techniques and integrating additional security tools for redundancy

- **DNS Security**
    - DNS Security
        - Measures to protect the Domain Name System from attacks and misconfigurations that could redirect or manipulate network traffic
        - Key DNS security concepts include Domain Name System Security Extensions (DNSSEC), Zone Transfers, DNS Poisoning, and Sinkholing
    - Domain Name System Security Extensions (DNSSEC)
        - Definition
            - Adds a layer of security to DNS by digitally signing DNS data to ensure integrity and authenticity
        - Purpose
            - Prevents cache poisoning by verifying that DNS responses are legitimate and from trusted sources
        - Example

        - When a user requests to visit a website, DNSSEC ensures the DNS data received is authentic, using public and private keys for verification
    - Implementation
        - Requires both DNS zones and DNS resolvers to support DNSSEC for complete protection
- Zone Transfers
    - Definition
        - A process to replicate DNS zone data across multiple servers, ensuring redundancy and reliability
    - Purpose
        - Keeps DNS data synchronized among authoritative servers, allowing for failover in case one server goes down
    - Vulnerability
        - If misconfigured, attackers can exploit zone transfers to access all DNS records, exposing sensitive information
    - Security Measures
        - Limit authorized servers for zone transfers and use Transaction Signatures (TSIG) to authenticate requests
- DNS Poisoning
    - Definition
        - Also known as cache poisoning, it involves injecting false DNS data into a resolver's cache, misdirecting users to malicious sites
    - Purpose of Attack
        - Redirects users to attacker-controlled sites, which may appear legitimate, to steal data or deliver malware

        - Defense Measures
            - DNSSEC
                - Validates DNS responses with digital signatures to prevent tampering
            - Patching and Updating
                - Keep DNS servers updated to close vulnerabilities that attackers might exploit
    - DNS Sinkholing
        - Definition
            - A defensive technique redirecting malicious traffic to a controlled sinkhole server instead of allowing it to reach a harmful destination
        - Purpose
            - Stops malware and botnet traffic by preventing communication with command-and-control servers
        - Example
            - When malware attempts to connect to a control server, the DNS server intercepts the request and sends it to the sinkhole instead, cutting off the connection
        - Requirements
            - The DNS server must stay updated with the latest threat intelligence, and sinkhole servers must be secured
    - Summary
        - DNS Security protects networks from DNS-based attacks and misconfigurations
        - Key elements include

                - DNSSEC
                    - Uses digital signatures to verify DNS data authenticity and prevent cache poisoning
                - Zone Transfers
                    - Synchronizes DNS data but must be secured to prevent unauthorized access
                - DNS Poisoning
                    - Redirects users by injecting false DNS data; defended by DNSSEC and server updates
                - DNS Sinkholing
                    - Redirects malicious traffic to a safe server to prevent malware communication
            - These strategies work together to enhance DNS integrity, protect user traffic, and maintain secure network operations

- **Email Security**
    - Email Security
        - Protecting email communications from threats like phishing, spoofing, and unauthorized access through protocols such as SPF, DKIM, DMARC, and S/MIME
    - Sender Policy Framework (SPF)
        - Definition
            - An email validation protocol that specifies which mail servers are authorized to send emails on behalf of a domain
        - Purpose

            - Prevents email spoofing by allowing domain owners to create an allowlist of approved mail servers
        - Mechanism
            - Uses DNS TXT records to list allowed IP addresses for sending email on behalf of a domain
        - Example
            - If an unauthorized server attempts to send an email pretending to be from the domain, the email can be flagged or blocked
    - DomainKeys Identified Mail (DKIM)
        - Definition
            - Adds a digital signature to outgoing emails, allowing recipients to verify message integrity and the sender's authenticity
        - Purpose
            - Prevents email tampering by confirming that the email content has not been altered
        - Mechanism
            - The sender’s private key creates a signature, while the public key is stored in DNS for recipient verification
        - Example
            - Validating that an email has not been altered after it was sent from the domain
    - Domain-based Message Authentication, Reporting, and Conformance (DMARC)
        - Definition
            - Builds on SPF and DKIM by providing a policy for handling unauthenticated emails
        - Purpose

        - Protects against phishing and spoofing by enforcing actions on emails that fail SPF or DKIM checks.
    - Mechanism
        - Domain owners publish policies that tell receiving servers whether to reject, quarantine, or flag emails that fail authentication checks
    - Example
        - Instructs mail servers to reject emails failing SPF/DKIM checks and provides feedback on handling
- Secure/Multipurpose Internet Mail Extensions (S/MIME)
    - Definition
        - A standard that adds encryption and digital signatures to individual email messages
    - Purpose
        - Provides confidentiality, integrity, authentication, and non-repudiation at the message level
    - Mechanism
        - Uses public key cryptography to encrypt emails, requiring recipients to use their private keys for decryption
    - Example
        - Encrypted emails between users where only the intended recipient can read the content
- Summary
    - Email Security
        - Protects communications by preventing phishing, spoofing, and unauthorized access
    - SPF

        - Verifies authorized mail servers to prevent spoofing
    - DKIM
        - Adds a digital signature to ensure message integrity
    - DMARC
        - Combines SPF and DKIM checks to enforce handling policies for unauthenticated emails
    - S/MIME
        - Provides encryption and digital signatures, securing individual messages and ensuring privacy

- **Network Issues**
    - Network Issues
        - Problems that disrupt data flow in networks, impacting connectivity, performance, and security. Common network issues include NACL misconfigurations, resource exhaustion, and Distributed Denial of Service (DDoS) attacks
    - Network Access Control List (NACL) Issues
        - Definition
            - Problems in network traffic control when NACL rules are misconfigured, either blocking legitimate traffic or allowing harmful traffic
        - Blocking Legitimate Traffic
            - Misconfigured rules can prevent users from accessing critical systems, leading to productivity loss
        - Permissive Settings

            - Overly permissive NACLs allow unauthorized traffic, increasing the risk of attacks
        - Mitigation
            - Regular auditing and fine-tuning of NACLs help ensure balanced security and accessibility
    - Resource Exhaustion
        - Definition
            - Occurs when critical network resources like bandwidth, memory, or processing power are depleted
        - Causes
            - Excessive demand during peak hours, inefficient resource management, and unbalanced workloads
        - Example
            - A server without load balancing might crash under excessive traffic, disrupting essential services
        - Mitigation
            - Proactive monitoring, load balancing, and scaling resources help prevent depletion
    - Distributed Denial of Service (DDoS) Attacks
        - Definition
            - An attack that overwhelms a network with traffic from multiple sources, exhausting resources and making services unavailable to legitimate users
        - Impact
            - Causes downtime, revenue loss, and reputational damage, often targeting vulnerable networks

        - Example
            - A large-scale attack on an e-commerce site could prevent customers from accessing services, resulting in financial losses
        - Mitigation
            - Use of DDoS mitigation services, firewalls, and distributed resources across data centers can help protect against these attacks
    - Summary
        - Network Issues
            - Disrupt connectivity, performance, and security in enterprise networks
        - NACL Issues
            - Misconfigured rules can block or permit incorrect traffic, leading to security vulnerabilities or disruptions
        - Resource Exhaustion
            - Depletion of critical resources due to high demand or inefficient management
        - DDoS Attacks
            - Flood of traffic from multiple sources targeting network resources, resulting in service unavailability

- **Cryptographic Issues**
    - Cryptographic Issues
        - Problems in encryption protocol application, affecting the secure transmission of data and overall system integrity

        - Common cryptographic issues include TLS errors, cipher mismatches, and improper cryptographic implementation
    - Transport Layer Security (TLS) Errors
        - Definition
            - Errors in establishing a secure connection between a client and a server, often due to protocol incompatibilities or certificate issues
        - Protocol Incompatibility
            - Occurs when a client and server support different protocol versions (e.g., TLS 1.2 vs. TLS 1.0), preventing secure connection establishment
        - Certificate Issues
            - Problems with certificates (e.g., expired, invalid domain, broken trust chain) disrupt the TLS handshake, affecting secure connections
        - Mitigation
            - Regularly update protocols to modern versions (e.g., TLS 1.2 or 1.3), monitor certificates, and ensure protocol compatibility
    - Cipher Mismatch
        - Definition
            - Occurs when the client and server cannot agree on a common encryption algorithm, leading to failed or insecure connections
        - Components of Cipher Suite
            - Includes key exchange, digital signature, encryption algorithm, and hashing algorithm. If incompatible, the handshake fails
        - Example

            - If a server supports only strong ciphers (e.g., AES-GCM) but the client only supports weak ones (e.g., DES-CBC3), a secure connection cannot be established
        - Mitigation
            - Configure servers to prioritize strong encryption methods, disable outdated ciphers, and regularly scan for weak ciphers to ensure up-to-date cryptographic configurations
    - Issues with Cryptographic Implementations
        - Definition
            - Vulnerabilities from improper application of cryptographic techniques, such as weak key management or misconfigured encryption
        - Weak Key Management
            - Using static or outdated keys weakens encryption and can be exploited by attackers
            - Proper key rotation and generation of unique session keys help maintain security
        - Outdated Algorithms
            - Use of deprecated algorithms (e.g., MD5 for hashing) exposes systems to attacks, such as collision attacks
        - Mitigation
            - Use strong algorithms like AES, remove deprecated ones, and ensure proper key management practices like unique, regularly rotated keys for secure connections
    - Summary
        - Cryptographic Issues

                - Affect secure data transmission and system integrity
            - TLS Errors
                - Result from protocol incompatibilities and certificate issues, preventing secure connections
            - Cipher Mismatch
                - Occurs when incompatible encryption algorithms lead to failed connections, requiring up-to-date configurations
            - Cryptographic Implementation Issues
                - Arise from weak encryption application, outdated algorithms, and poor key management, leading to security vulnerabilities

- **PKI Issues**
    - PKI Issues
        - Problems affecting the security, trust, and reliability of Public Key Infrastructure (PKI), impacting the proper management of encrypted communications and digital certificates
        - Common issues include CA misconfigurations, expired or improperly issued certificates, challenges in certificate revocation, and improper key management
    - Certificate Authority (CA) Misconfigurations
        - Definition
            - Errors in configuring a CA can lead to the issuance of certificates to unauthorized entities
        - Security Risks

            - Misconfigurations can allow unauthorized users to impersonate legitimate parties, leading to potential security breaches and unauthorized data access
        - Example
            - Attackers with access to a compromised CA can issue fraudulent certificates for trusted sites, facilitating on-path attacks and data interception
        - Mitigation
            - Secure CA private keys, configure CA infrastructure properly, use hardware security modules (HSMs), and conduct regular audits
    - Expired or Improperly Issued Certificates
        - Definition
            - Certificates must be valid and accurately represent the identity of their holders
        - Consequences
            - Expired Certificates
                - Prevent users from establishing secure connections, affecting user trust and accessibility
            - Improperly Issued Certificates
                - Allow attackers to impersonate legitimate entities, leading to phishing and fraud
        - Example
            - Domain squatting can exploit improper issuance by using similar domain names (e.g., "rnysecurebank.com") to trick users into trusting fake websites
        - Mitigation

        - Strict validation procedures, regularly update and monitor certificate validity, and conduct rigorous identity checks before certificate issuance
- Challenges in Certificate Revocation
    - Definition
        - Revocation ensures certificates no longer in use are marked as invalid but can face propagation delays
    - Issues
        - Propagation Delay
            - Delay in updating Certificate Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP) responses can allow unauthorized access
        - Lack of Real-Time Status
            - Systems that don't check for revoked certificates may accept compromised certificates as valid
    - Mitigation
        - Implement systems with real-time status checking (OCSP), regularly update and distribute CRLs, and prioritize timely revocation processes
- Improper Key Management
    - Definition
        - Inadequate protection and distribution of private and public keys can lead to unauthorized access
    - Risks
        - Key Theft

            - If private keys are not securely stored, attackers can use them to decrypt data or impersonate the key owner
        - Improper Distribution
            - If keys are shared insecurely (e.g., unencrypted email), they can be intercepted by unauthorized parties
        - Mitigation
            - Use HSMs or secure storage for private keys, ensure secure distribution channels, and enforce regular key rotation policies
- Summary
    - PKI Issues
        - Undermine the security and trustworthiness of encrypted communications and digital certificates
    - CA Misconfigurations
        - Allow unauthorized certificate issuance, risking impersonation and unauthorized access
    - Expired/Improperly Issued Certificates
        - Lead to failed secure connections or allow attackers to impersonate legitimate entities
    - Certificate Revocation Challenges
        - Cause delays in identifying invalid certificates, risking unauthorized access
    - Improper Key Management
        - Risks key theft and improper distribution, compromising encrypted data and secure communications

Cloud Security

Objective 2.5: Securely implement cloud capabilities in an enterprise environment

- **Cloud Implementation**
    - Cloud Implementation
        - The process of deploying and configuring cloud services with security measures to protect data and ensure compliance
        - Key components of cloud implementation include cloud service adoption and applying preventive, detective, and proactive cloud control strategies
    - Cloud Service Adoption
        - Definition
            - The integration of cloud platforms and services into an organization's existing infrastructure to leverage scalable resources, increased storage, and flexible computing power
        - Implementation Strategy
            - Deciding which services to move to the cloud and how to integrate them with existing systems, such as using a hybrid cloud approach where sensitive data remains on-premises while less critical operations move to the cloud
        - Security Considerations
            - Establishing access controls, data encryption, and compliance measures due to the shared nature of cloud environments
        - Regulatory Compliance

            - Ensuring cloud platforms meet industry-specific regulatory requirements like HIPAA or GDPR, requiring thorough risk assessments for secure cloud adoption
- Cloud Control Strategies
    - Preventative Controls
        - Definition
            - First line of defense, designed to prevent security incidents
        - Examples
            - Identity and access management (IAM) systems and multi-factor authentication (MFA) help restrict access to only authorized users
    - Detective Controls
        - Definition
            - Monitor systems to detect and respond to security incidents or breaches as they occur
        - Examples
            - Real-time logging and monitoring solutions such as AWS CloudTrail and Microsoft Azure Monitor detect abnormal patterns and alert security teams to respond quickly
    - Proactive Controls
        - Definition
            - Anticipate and address potential security risks before they manifest
        - Examples
            - Regular security assessments, including vulnerability scans, penetration testing, and threat intelligence services, help

            identify and mitigate potential weaknesses in the cloud environment

- Summary
    - Cloud Implementation
        - Integrates and configures cloud services with security measures for data protection and regulatory compliance
    - Cloud Service Adoption
        - Integrates cloud platforms into existing infrastructure, balancing benefits like scalability with security needs, including regulatory compliance
    - Cloud Control Strategies
        - Uses preventative controls to stop incidents, detective controls to monitor for breaches, and proactive controls to address threats in advance, creating a secure cloud adoption environment

- **Cloud Management**
    - Cloud Management
        - The process of overseeing and securing cloud resources with clearly defined responsibilities between the cloud provider and the customer
        - Key aspects of cloud management include the shared responsibility model and management of encryption keys and licenses in both cloud and customer-managed environments
    - Shared Responsibility Model
        - Definition
            - A model that delineates security responsibilities between the cloud service provider (CSP) and the client

        - Types of Controls
            - Inherited Controls
                - Fully managed by the CSP (e.g., physical infrastructure and environmental controls)
            - Shared Controls
                - Responsibilities shared between CSP and client (e.g., patch management where CSP patches infrastructure, and the client patches operating systems)
            - Customer-Specific Controls
                - Solely the client's responsibility (e.g., configuring security zones and setting application communication protocols)
        - Responsibility Division
            - CSP manages physical hardware, regions, availability zones, and core services; clients manage their data protection, system configuration, and identity access permissions
    - Cloud vs. Customer-Managed Resources
        - Definition
            - In cloud environments, the CSP manages core infrastructure, while customers manage critical resources like encryption keys and licenses
        - Encryption Key Management
            - Cloud Provider's Role
                - Maintains infrastructure for secure key storage (e.g., AWS Key Management Service (KMS) and Azure Key Vault)
            - Customer's Role

            - Generates, manages, and rotates keys properly. Mismanagement (e.g., failing to rotate keys) can expose sensitive data
        - License Management
            - Cloud License Bundling
                - Some licenses are bundled with cloud subscriptions, reducing administrative workload
            - Customer-Managed Licensing
                - Requires tracking, renewal, and compliance of licenses, often needing dedicated staff to ensure timely renewals and legal compliance
    - Summary
        - Cloud Management
            - Involves securing cloud resources with a clear division of responsibilities between CSP and customer
        - Shared Responsibility Model
            - CSP handles physical infrastructure; clients manage data encryption, security configurations, and access
        - Cloud vs. Customer-Managed Resources
            - CSP often manages key storage and licensing, but customers are responsible for encryption key management and compliance

- **Connectivity and Integration**
    - Connectivity and Integration

        - Ensuring secure and seamless connections between customer systems and cloud services, focusing on secure connections, data flow, and management of unauthorized applications
    - Customer-to-Cloud Connectivity
        - Definition
            - Secure connection between a customer's network and the cloud provider
        - Secure Connection Options
            - VPNs
                - Encrypted, internet-based connections that protect data in transit
            - Dedicated Lines (e.g., AWS Direct Connect)
                - Private, high-speed connections with increased reliability and reduced latency
        - Use Case Example
            - A financial organization uses a private line to transfer sensitive client data securely between its network and the cloud
        - Performance Considerations
            - Redundant connections and failover strategies ensure uptime and high performance
    - Cloud Service Integration
        - Definition
            - Linking different cloud services and platforms securely to enable data flow and interoperability
        - Key Security Measures
            - Encryption

      - Protects data both at rest and in transit across public, private, or hybrid cloud environments
    - Access Control Policies
      - Restrict access to authorized services and users only
  - Integration Platforms
    - Tools like MuleSoft and Microsoft Azure Logic Apps manage secure data flow and compatibility between services
  - Example
    - A business integrates Google Cloud and Salesforce through secure APIs for real-time data sharing in CRM
- Shadow IT Detection
  - Definition
    - Identifying and managing unauthorized cloud services or applications used by employees without IT team knowledge
  - Key Risks
    - Data Leakage
      - Unapproved services may lack encryption or access controls, increasing exposure risks
    - Regulatory Non-Compliance
      - Unauthorized services can lead to breaches in data privacy regulations like HIPAA or GDPR
  - Detection Tools
    - Solutions like Microsoft Cloud App Security or Cisco Cloudlock monitor activity and flag unapproved applications
  - Mitigation

        - Block unauthorized services or transition employees to approved alternatives to maintain compliance and reduce security risks
  - Summary
    - Connectivity and Integration
      - Establishes secure connections, enables seamless integration between cloud services, and manages unauthorized application use
    - Customer-to-Cloud Connectivity
      - Provides secure, reliable connections for data transmission
    - Cloud Service Integration
      - Ensures secure and efficient data flow across multiple platforms
    - Shadow IT Detection
      - Identifies unauthorized applications, reducing security risks and ensuring regulatory compliance

- **Cloud Security Considerations**
  - Cloud Security Considerations
    - Addressing risks and vulnerabilities to protect data and resources within cloud environments
    - Key considerations include insecure storage resources, data exposure, data leakage, and data remanence
  - Insecure Storage Resources
    - Definition
      - Cloud storage lacking proper security configurations, such as encryption or access controls
    - Common Storage Types

        - Buckets (AWS) and Blobs (Azure) used to store data in different cloud environments
    - Misconfigurations
        - Issues arise with default settings (e.g., default read/write access) and improper configuration of access control lists (ACLs), IAM authorizations, origin settings, and cross-origin resource sharing (CORS)
    - Mitigation
        - Regularly update access controls, check CORS settings, and enforce strong security policies for secure data storage
- Data Exposure
    - Definition
        - Sensitive information unintentionally made accessible to unauthorized users, often due to misconfigurations
    - Causes
        - Incorrect access permissions on cloud storage (e.g., public access), weak IAM policies, and unrevoked access
    - Example
        - An improperly configured database set to "public" exposes customer data to anyone with the URL
    - Mitigation
        - Encrypt sensitive data at rest and in transit, regularly audit permissions, and use automated security tools like AWS Config or Azure Security Center to detect misconfigurations
- Data Leakage
    - Definition

        - Unauthorized transfer of sensitive data to external or untrusted parties
    - Examples
        - Employees using unapproved cloud services, insecure email communications, and external storage (e.g., personal Dropbox)
    - Risk
        - Data leakage often goes unnoticed, increasing the risk of theft or unauthorized access
    - Prevention
        - Implement Data Loss Prevention (DLP) tools, monitor employee data movement, and enforce strict policies regarding personal device and cloud service usage
- Data Remanence
    - Definition
        - Residual data that remains on storage devices after deletion, posing a security risk if improperly sanitized
    - Cloud Concern
        - Remanence can occur when virtual machines, containers, or volumes are deleted but not fully wiped
    - Example
        - A cloud provider reuses hardware without fully wiping data, exposing residual information to future clients
    - Mitigation
        - Use cryptographic erasure or physical destruction, and verify that the cloud provider follows data destruction standards to prevent unauthorized recovery of residual data

- Summary
  - Cloud Security Considerations address risks to data and resources within cloud environments
  - Insecure Storage Resources
    - Risks arise when storage containers lack proper configurations, like access controls
  - Data Exposure
    - Occurs when sensitive information is accidentally accessible due to misconfigurations
  - Data Leakage
    - Refers to unauthorized data transmission to external entities, often unnoticed and at risk of theft
  - Data Remanence
    - Involves residual data remaining on devices after deletion, requiring secure erasure to mitigate recovery risks

- **API Security**
  - API Security
    - Measures to protect APIs from unauthorized access, misuse, and attacks, ensuring secure interactions between applications and cloud services
    - Key concepts include authorization, rate limiting, and logging
  - Authorization
    - Definition
      - Ensures that only authenticated users and applications have access to specific API endpoints
    - Implementation

        - Typically through token-based systems like OAuth or JSON Web Tokens (JWT), which verify user identity and permissions for each API request
    - Role-Based Access Control (RBAC)
        - Allows fine-grained control, such as permitting only managers to approve requests while other employees can only submit them
    - Example
        - In a healthcare application, a doctor can access patient records, while patients can only view their own data
    - Benefit
        - Protects sensitive data by preventing unauthorized access and actions
- Rate Limiting
    - Definition
        - Controls the number of API requests a user or application can make within a specific timeframe
    - Function
        - Prevents abuse, overuse, and denial-of-service (DoS) attacks, ensuring fair usage and stable performance
    - Implementation
        - Each user or application is assigned a request limit per minute or hour; exceeding this limit results in request rejection
    - Example
        - An online service may allow 100 requests per minute per user to prevent service degradation or abuse
    - Benefit

- Protects APIs from excessive traffic, ensuring system stability and availability

- Logging
  - Definition
    - Records every API interaction, creating an audit trail for monitoring usage, detecting threats, and troubleshooting
  - Details Captured
    - User identity, action, timestamp, and result of each interaction
  - Compliance
    - Helps organizations meet regulatory standards, like HIPAA, by maintaining a record of all interactions with sensitive data
  - Example
    - Logs of access to patient records in a healthcare API ensure only authorized access, supporting compliance
  - Benefit
    - Enables detection of suspicious activity, compliance auditing, and troubleshooting
- Summary
  - API Security
    - Protects against unauthorized access and misuse, ensuring secure application and cloud service interactions
  - Authorization
    - Allows only authenticated users access to permitted API endpoints, protecting sensitive data
  - Rate Limiting

                - Limits request frequency to prevent abuse, ensuring fair usage and stable API performance
            - Logging
                - Tracks API interactions, aiding in compliance, threat detection, and troubleshooting

- **Cloud Access Security Broker (CASB)**
    - Cloud Access Security Broker (CASB)
        - A security solution that acts as a control point between cloud service users and cloud applications, enforcing security policies and protecting data across cloud environments
        - CASB implementations include API-based and proxy-based solutions
    - API-Based CASB
        - Definition
            - Connects directly with cloud services via an Application Programming Interface (API), monitoring and controlling data flow without routing user traffic
        - Function
            - Enforces security policies and provides visibility by directly communicating with the cloud provider's API
        - Example
            - A CASB can update cloud provider access settings, blocking users as soon as they're removed from the network (e.g., preventing Jason from logging in after his access is revoked)
        - Limitation

        - Dependent on the API capabilities offered by the cloud provider, which may limit control over granular policies (e.g., geographic restrictions)
    - Use Case
        - Ideal for cloud-native services requiring minimal user device modifications
- Proxy-Based CASB
    - Definition
        - Intercepts and inspects traffic between users and cloud services, enforcing security policies in real-time by routing user traffic through a proxy server
    - Forward Proxy
        - Positioned at the user's network edge, inspecting all outgoing traffic before reaching cloud services
        - Example
            - A parent monitors home network traffic for website restrictions
        - Limitation
            - Inspects all network traffic, which can cause bottlenecks and may allow users to bypass proxy restrictions
    - Reverse Proxy
        - Positioned between users and specific cloud services, inspecting only traffic directed at supported cloud applications, enhancing performance by reducing unnecessary inspections
        - Limitation

            - Only compatible with cloud services supporting reverse proxy configurations
        - Use Case
            - Suited for environments requiring advanced security configurations like blocking specific applications or managing data access by geographic location
    - Summary
        - CASB
            - Manages and secures access to cloud services, enforcing data protection and security policies
        - API-Based CASB
            - Integrates with cloud services via API, providing seamless control over data flow without impacting user experience
        - Proxy-Based CASB
            - Intercepts and inspects user traffic through a forward or reverse proxy, offering greater control but potentially impacting performance

- **Development and Deployment**
    - Development and Deployment
        - The process of creating, configuring, and releasing cloud-based applications and infrastructure while integrating security best practices throughout
        - Key tools and concepts include Terraform, Ansible, and Package Monitoring
    - Terraform

        - Definition
            - An Infrastructure as Code (IaC) tool that automates the provisioning and management of cloud infrastructure
        - Purpose
            - Ensures consistency and security across environments by managing resources (servers, databases, networks) through code
        - Example
            - A development team uses Terraform to define a secure virtual private cloud (VPC) with subnets, security groups, and access control lists
        - Features
            - Multi-Cloud Compatibility
                - Manages infrastructure across AWS, Azure, GCP, and other cloud providers
            - State Management
                - Tracks real-time infrastructure state to detect and resolve any configuration drift
    - Ansible
        - Definition
            - An automation tool for configuration management, application deployment, and task automation across cloud environments
        - Purpose
            - Focuses on configuring and managing applications on provisioned infrastructure, ensuring consistency and reducing configuration drift
        - Example

- After setting up a virtual machine with Terraform, Ansible can install security patches, configure firewalls, and deploy applications using a playbook

■ Features

- Playbooks
  - YAML files that define automation tasks
- Agentless Architecture
  - Uses SSH for communication, making integration lightweight and secure

○ Package Monitoring

■ Definition

- The process of tracking and analyzing the security and integrity of software packages and dependencies used in development

■ Purpose

- Prevents vulnerabilities by identifying outdated or compromised packages before they are used in deployment

■ Example

- A team uses open-source libraries but employs package monitoring to detect any vulnerabilities in dependencies

■ Tools

- OWASP Dependency-Check
  - Scans project dependencies for vulnerabilities across various programming languages
- npm audit
  - Specifically scans JavaScript and Node.js dependencies for security issues

            - Integration
                - Often incorporated into continuous integration pipelines to alert teams when vulnerabilities are detected
        - Summary
            - Development and deployment of cloud-based applications require robust security measures
            - Essential tools and practices include
                - Terraform
                    - Manages cloud infrastructure with IaC, ensuring secure and consistent resource configuration across environments
                - Ansible
                    - Automates configuration and application management, using playbooks to ensure consistent settings and reduce drift
                - Package Monitoring
                    - Uses tools like OWASP Dependency-Check and npm audit to identify and mitigate vulnerabilities in software dependencies
            - Together, these tools and practices maintain security, consistency, and efficiency in the development and deployment of cloud applications

- **CI/CD Pipeline**
    - Continuous Integration/Continuous Deployment (CI/CD) Pipeline
        - An automated process that integrates, tests, and deploys code changes securely and efficiently throughout the software development lifecycle, ensuring consistent security measures

- Source Code Integration
  - Definition
    - The stage where developers regularly merge code into a shared repository
  - Function
    - Automated processes check code for syntax errors and run unit tests to catch errors early
  - Analogy
    - Like the initial assembly station in a car factory where parts are checked for fit before moving forward
- Automated Testing
  - Definition
    - Rigorous testing phase where code undergoes security scans, performance testing, and functional checks
  - Function
    - Ensures code stability, security, and proper functionality by identifying issues before deployment
  - Analogy
    - Testing a car’s systems (engine, brakes) on the assembly line to verify safety and performance
- Build and Deployment
  - Definition
    - Stage where the code is built, packaged, and deployed to staging or production environments
  - Function

            - Code is deployed automatically to reduce human error and ensure it meets security and performance standards
        - Analogy
            - Final assembly and quality checks before a car is sent to dealerships for customer use
    - Monitoring and Feedback
        - Definition
            - Continuous monitoring of deployed code to ensure functionality and security in a real-world environment
        - Function
            - Detects and addresses bugs or performance issues, ensuring continuous stability post-deployment
        - Analogy
            - Safety inspections of cars after they leave the factory to verify ongoing performance
    - Summary
        - CI/CD Pipeline
            - Automates integration, testing, and deployment to streamline secure software delivery
        - Source Code Integration
            - Code is regularly merged and checked for syntax and functionality errors
        - Automated Testing
            - Security and performance checks ensure code stability before deployment
        - Build and Deployment

            - Code is packaged and deployed automatically, meeting all benchmarks
        - Monitoring and Feedback
            - Post-deployment monitoring ensures ongoing security and functionality in production

- **Container Management**
    - Container Management
        - Overseeing the deployment, operation, and security of containers to ensure applications run securely and efficiently across cloud environments
    - Container Security
        - Definition
            - Protecting containerized applications and their underlying infrastructure from vulnerabilities
        - Image Security
            - Securing container images by using trusted repositories, scanning for vulnerabilities, and updating images regularly
        - Access Control
            - Using role-based access control (RBAC) to restrict access to containers based on user roles
        - Monitoring
            - Implementing tools like Falco and Sysdig to detect unauthorized access or abnormal behavior within containers in real time
    - Container Orchestration
        - Definition

        - Automating the deployment, scaling, and management of containers across multiple servers
    - Automated Scaling
        - Tools like Kubernetes automatically scale containers based on demand (e.g., increased traffic)
    - Configuration Consistency
        - Enforcing security policies (e.g., non-root privileges) to maintain secure and consistent container configurations
    - Update Management
        - Rolling out security patches without downtime using orchestration tools
    - High Availability
        - Managing failover and recovery to ensure continuous availability of containers, even during host failures
- Summary
    - Container Management
        - Ensures secure, efficient deployment and operation of containerized applications
    - Container Security
        - Involves securing images, controlling access, and continuous threat monitoring
    - Container Orchestration
        - Automates deployment, scaling, patching, and failover processes to maintain high performance and availability

- **Serverless Computing**
  - Serverless Computing
    - Running application code without managing the underlying infrastructure, with the cloud provider handling the execution environment and associated resources
  - Workloads
    - Definition
      - Specific tasks or operations that serverless functions execute, triggered automatically (e.g., HTTP requests, database updates)
    - Efficiency
      - Workloads execute only when needed, reducing idle time and costs by paying only for the actual processing time
    - Scalability
      - Automatically scales to handle surges in demand without over-provisioning resources, returning to normal levels when demand drops
    - Automation & Reliability
      - Cloud providers ensure stable execution, rerouting workloads as needed to maintain uptime
  - Functions
    - Definition
      - Independent units of work performing a single task, supporting modular and flexible application design
    - Statelessness

        - Functions don’t retain data between executions, making them efficient for quick, repeatable tasks like data processing or image resizing
    - Microservices
        - Functions work together in a microservices architecture, allowing independent scaling and updating of each component
    - Examples
        - FaaS offerings like AWS Lambda, Google Cloud Functions, and Azure Functions support automated, streamlined function deployment
- Resources
    - Definition
        - Underlying compute, storage, and networking managed by the cloud provider, dynamically allocated to serverless functions
    - Dynamic Allocation
        - Resources are used only when functions are triggered, saving costs and avoiding over-provisioning
    - High Availability
        - Cloud providers replicate serverless functions across data centers, ensuring availability and fault tolerance
    - Security
        - Providers handle security controls, such as encryption and access restrictions, to protect data and ensure resource safety
- Summary
    - Serverless Computing

        - Allows applications to run without direct infrastructure management, relying on cloud providers to handle the execution environment
    - Workloads
        - Defined tasks that trigger serverless functions to run as needed, ensuring cost-effective and scalable operations
    - Functions
        - Independent, stateless units performing single tasks, facilitating modular and flexible applications
    - Resources
        - Compute, storage, and networking managed by the provider, offering dynamic allocation, high availability, and built-in security

Specialized System Security

Objective 3.5: Secure specialized and legacy systems against threats

- **Specialized Systems**
    - Specialized Systems
        - Purpose-built technologies like Systems-on-a-Chip (SoCs), embedded systems, Internet of Things (IoT) devices, and wireless technologies, each requiring unique security measures due to their specific functions and operational constraints
    - System-on-a-Chip (SoC)
        - Definition
            - An integrated circuit containing all components of a computer system (e.g., processor, memory, storage, USB controllers, and wireless radios)
        - Purpose
            - Provides high performance in a compact, low-power format, suitable for small devices
        - Example
            - Used in devices like smart TVs, streaming sticks, and smartphones
        - Benefit
            - Simplifies device design by embedding all necessary computing components on a single chip, enhancing energy efficiency
    - Embedded Systems
        - Definition

        - Computing systems within larger devices performing dedicated functions
    - Applications
        - Found in everyday items such as washing machines, medical devices, and industrial machinery
    - Purpose
        - Executes specific, limited tasks reliably, focusing on low resource use and stability
    - Example
        - Microcontrollers in cars managing engine control, or in smart thermostats adjusting temperature
- Internet of Things (IoT)
    - Definition
        - Networked devices that interact, send, and receive data through the internet
    - Examples
        - Smart lights, thermostats, industrial sensors, and healthcare equipment
    - Purpose
        - Allows for real-time monitoring, automation, and control of connected devices
    - Impact
        - Improves efficiency and scalability in homes, industries, and healthcare by automating tasks and collecting data
- Wireless Technologies (including Radio Frequency, RF)
    - Definition

        - Communication methods that transmit data via radio waves, enabling wireless connections
    - Applications
        - RF (radio frequency) signals, Wi-Fi, Bluetooth, Zigbee, and Z-Wave are common wireless protocols
    - RF Details
        - RF uses different frequency ranges, such as 2.4 GHz for Wi-Fi and 900 MHz for Z-Wave, with each protocol optimized for various ranges and interference management
    - Purpose
        - Supports seamless communication in IoT and embedded systems without physical wiring
- Summary
    - Specialized Systems
        - Unique technologies requiring specific security protocols
    - SoC
        - Integrates essential computing components onto a single chip, optimizing power and space for compact devices
    - Embedded Systems
        - Embedded within larger devices to perform dedicated tasks efficiently
    - IoT
        - Connects devices to the internet, enabling real-time data exchange across various industries
    - Wireless Technologies (RF)

- Allow wireless communication, crucial for IoT and embedded systems, providing flexible and efficient connectivity solutions

- **Operational Technology (OT)**
    - Operational Technology (OT)
        - Systems used to monitor and control physical processes within critical infrastructure, such as HVAC systems, Industrial Control Systems (ICS), and Supervisory Control and Data Acquisition (SCADA) systems
        - These systems are essential for the continuous operation of industries and require strong security to protect against disruptions and unauthorized access
    - Heating, Ventilation, and Air Conditioning (HVAC)
        - Definition
            - Systems controlling climate in buildings, including temperature, airflow, and air quality
        - Components
            - Sensors and Controllers
                - Adjust heating, cooling, and ventilation automatically
            - Programmable Logic Controller (PLC)
                - Controls HVAC functions based on sensor data
            - Human-Machine Interface (HMI)
                - Allows operators to monitor and adjust HVAC settings
        - Purpose
            - Ensures safe environmental conditions for people and equipment, such as maintaining stable temperatures in data centers
        - Example

            - In data centers, HVAC systems prevent overheating of servers and network infrastructure by continuously regulating temperature and airflow
    - Industrial Control Systems (ICS)
        - Definition
            - Systems managing industrial processes like power generation or water distribution
        - Features
            - Programmable Logic Controllers (PLCs)
                - Automate tasks by processing input from sensors
            - Distributed Control System (DCS)
                - Interconnects multiple ICS for centralized control over larger operations
            - Data Historian
                - Collects and stores data for monitoring and security tracking
        - Example
            - On a U.S. Navy warship, ICS manages everything from power generation to water treatment, supporting the ship's self-contained operations
        - Security Note
            - ICS systems prioritize availability and may lack modern security features, making them vulnerable to cyber threats, especially if connected to the internet
    - Supervisory Control and Data Acquisition (SCADA)
        - Definition

            - A subset of ICS used for remote monitoring and control over wide areas, commonly in infrastructure like power plants or water treatment facilities
        - Components
            - Remote Sensors and PLCs
                - Monitor and control industrial processes across multiple locations
            - Central Control System
                - Processes data from field devices for analysis and control
        - Communication Methods
            - Cellular, satellite, or VPN connections allow SCADA to manage remote sites
        - Example
            - Smart meters in electric grids transmit data back to a SCADA system, enabling utilities to monitor energy use, detect outages, and adjust resources
        - Purpose
            - Enables centralized, remote monitoring and control, enhancing efficiency and real-time responsiveness in large-scale industrial applications
    - Summary
        - Operational Technology (OT)
            - Essential systems controlling physical processes in critical industries like power and water
        - HVAC

            - Controls environmental conditions, protecting both people and sensitive equipment, such as servers in data centers
        - ICS
            - Manages and automates industrial processes but can be vulnerable to cyber threats due to outdated security features
        - SCADA
            - Used for remote monitoring and control across wide areas, integrating multiple sites for efficient and centralized management

- **Characteristics of Specialized/Legacy Systems**
    - Specialized and Legacy Systems
        - Technologies with outdated features that make them challenging to secure
        - Characteristics include being obsolete, unsupported, unable to secure, or highly constrained, each adding unique security and operational risks
    - Obsolete Systems
        - Definition
            - Systems designed with outdated technology standards, lacking modern capabilities
        - Characteristics
            - Lack cloud connectivity, real-time analytics, or integration with current technologies
            - Slower and less efficient, often requiring more manual intervention and prone to failure

        - Lack protections against modern cyber threats, such as data encryption or security patches
    - Example
        - A hospital using an outdated patient record system that cannot connect with newer devices, requiring manual data transfers that increase error risks and delay care
- Unsupported Systems
    - Definition
        - Systems that no longer receive security updates, patches, or vendor support
    - Characteristics
        - Unaddressed security vulnerabilities due to lack of support
        - Reliance on in-house troubleshooting or third-party services, which can be costly
        - Non-compliance risks, especially in regulated industries like healthcare or finance
    - Example
        - A bank using an unsupported financial platform, risking operational issues due to lack of technical expertise and potential legal fines for non-compliance
- Unable to Secure Systems
    - Definition
        - Systems that, due to design or age, cannot be protected with modern security solutions
    - Characteristics

        - Lack of foundational security features such as encryption or secure access controls
        - Dependence on outdated protocols or systems that disrupt operations when upgraded
        - Often used in critical industries, where risks extend to individual safety and privacy
    - Example
        - A legacy SCADA system in an energy plant, initially built without internet connectivity in mind, now highly vulnerable in a networked environment
- Highly Constrained Systems
    - Definition
        - Systems with limited resources such as processing power, memory, or connectivity, unable to support modern security features
    - Characteristics
        - Focus on efficiency over security, often to avoid delays in critical processes
        - High cost or complexity in upgrading, leading to reliance on compensating security measures like network isolation
    - Example
        - An embedded system in an older industrial robot with limited processing capacity, making it unable to support diagnostic updates without full system replacement
- Summary

        - Specialized and Legacy Systems are outdated, often insecure technologies that pose risks when they are obsolete, unsupported, unable to secure, or highly constrained
        - Obsolete Systems lack integration with modern technologies, slowing operations
        - Unsupported Systems do not receive updates, creating security and compliance risks
        - Unable to Secure Systems are unprotectable by modern security measures, often found in critical infrastructure
        - Highly Constrained Systems have resource limitations that prevent security upgrades, often requiring compensatory isolation measures

- **Security Practices**
    - Security Practices
        - Techniques like segmentation, hardening, and monitoring to protect specialized systems from threats by isolating segments, reducing vulnerabilities, and ensuring real-time threat detection
    - Segmentation
        - Definition
            - Dividing a network into isolated sections to contain potential breaches and protect critical systems
        - Functionality
            - Limits access to sensitive systems by isolating different parts of the network
            - Uses VLANs and firewalls to separate and manage network traffic, particularly useful for industrial control systems

        - Example
            - Separating corporate email and operational systems in a factory so attackers breaching the corporate network cannot reach manufacturing controls
        - Tools
            - Cisco's Adaptive Security Appliance, Palo Alto's Next-Generation Firewalls
    - Hardening
        - Definition
            - Enhancing system security by reducing vulnerabilities through limiting services and applying security patches
        - Functionality
            - Disables unnecessary features (e.g., remote access on a local-only database) to minimize potential entry points
            - Applies security patches promptly to protect against known vulnerabilities
        - Example
            - Disabling FTP on a web server if it is not required, reducing the server's attack surface
        - Tools
            - Microsoft's Security Compliance Toolkit, Lynis
    - Monitoring
        - Definition
            - Observing system activity continuously to detect and respond to security incidents in real-time
        - Functionality

            - Uses SIEM systems and intrusion detection tools to track and analyze log data for unusual activity
            - Alerts administrators to potential threats (e.g., unauthorized access attempts) for quick response
        - Example
            - A retail business monitoring its payment system to detect unauthorized access attempts to prevent data breaches
        - Tools
            - Splunk, Graylog, Snort, Suricata
    - Summary
        - Security Practices
            - Use layered defenses to protect systems from threats
        - Segmentation
            - Divides the network, limiting attack spread by isolating sensitive systems
        - Hardening
            - Reduces vulnerabilities by disabling unneeded services and applying updates
        - Monitoring
            - Provides real-time tracking, alerting to threats immediately for rapid response

- **Data Management**
    - Data Management
        - Organizing, protecting, and analyzing data to maintain its integrity, confidentiality, and availability within specialized systems

        - Key components include aggregation and data analytics
    - Aggregation
        - Definition
            - The process of collecting and combining data from various sources to create a single, comprehensive dataset for analysis
        - Functionality
            - Unifies fragmented data from different systems to gain a holistic view of patterns, trends, or anomalies
            - Useful for analyzing data from multiple sources (e.g., customer purchases, app activity, and marketing campaigns) to uncover insights
        - Technical Requirements
            - Requires systems capable of collecting data in real-time or at intervals
            - Includes normalization to ensure data from different sources aligns in format and structure before storage
    - Data Analytics
        - Definition
            - Analyzing aggregated data to detect security incidents, identify trends, and inform decisions
        - Functionality
            - Detects security threats and suspicious activities using analytics tools
            - Typically facilitated by a Security Information and Event Management (SIEM) system, which normalizes, indexes, and analyzes data for threat detection

- Data Processing Pipeline
  - Modern pipelines often use stream processing for near real-time data analysis
  - Indexing and Log Curation
    - Essential for efficient data searches and focusing on relevant information to avoid data overload
- Security
  - Database Activity Monitoring (DAM) tools ensure database security by detecting unauthorized access and monitoring database activity in real-time
  - Methods
    - Interception-Based
      - Monitors communication between client and server
    - Memory-Based
      - Captures SQL statements as executed, offering in-depth visibility
    - Log-Based
      - Monitors transactions based on database server logs, useful for auditing and compliance

- Summary
  - Data Management
    - Ensures data remains secure and organized
  - Aggregation
    - Combines data from multiple sources, providing a complete dataset for analysis

        - Data Analytics
            - Examines aggregated data for security threats and insights; uses tools like SIEM and DAM to manage and protect data in real-time
        - Database Activity Monitoring (DAM)
            - Protects databases from unauthorized access, contributing to overall data integrity and compliance

- **Compliance and Regulatory Considerations**
    - Compliance and Regulatory Considerations
        - Adhering to laws, standards, and guidelines specific to industries to ensure systems operate safely, securely, and within legal requirements
    - Regulatory Considerations
        - Definition
            - Ensure businesses and systems operate according to industry laws and guidelines to promote legality and safety
        - Examples
            - HIPAA
                - Protects patient data in healthcare.
            - GDPR
                - Safeguards user privacy in the European Union
        - Importance
            - Builds trust with customers and partners
            - Ensures business continuity and risk management
            - Non-compliance can lead to fines, legal issues, and reputational damage
        - Benefits

        - Attracts clients and partners valuing secure practices
        - Provides a competitive advantage
- Environmental Considerations
    - Definition
        - Ensure systems operate safely within their physical environments
    - Focus Areas
        - Data Centers
            - Must maintain optimal temperature (64 to 81 degrees Fahrenheit) and humidity (40 to 60 percent) to protect sensitive equipment
        - Power Fluctuations
            - Systems must avoid disruptions from environmental changes or electromagnetic interference
    - Examples
        - Cleanrooms
            - Designed for semiconductor production to limit dust and maintain humidity
    - Consequences of Neglect
        - Potential system failures, downtime, data loss, and loss of client trust
- Safety Considerations
    - Definition
        - Ensure systems are designed to protect operators, users, and the public
    - Standards
        - IEC 61508

            - Covers functional safety in electronic systems
        - ISO 26262
            - Focuses on safety for electrical and electronic systems in the automotive industry
    - Importance
        - Prevents accidents and reduces liability. Ensures reliable performance of systems
    - Examples
        - Automobiles
            - Safety standards for systems like airbags and braking systems
    - Outcome
        - Enhances operational reliability and safety for all users
- Summary
    - Compliance and Regulatory Considerations
        - Ensure systems adhere to laws and standards, enhancing security and legality
    - Regulatory Considerations
        - Vital for data protection and fostering trust with stakeholders, avoiding penalties and promoting responsible practices
    - Environmental Considerations
        - Focus on maintaining optimal conditions for system performance, crucial for preventing failures and maintaining service quality
    - Safety Considerations
        - Prioritize user and public safety, ensuring systems function reliably and comply with safety regulations to prevent accidents

- **Critical Services Challenges**
    - Critical Services Challenges
        - Security issues faced by essential infrastructure sectors (utilities, transportation, healthcare) in protecting against disruptions and threats that can have significant societal impacts
    - Utilities
        - Definition
            - Essential infrastructure including power grids, water systems, and gas pipelines that require robust cybersecurity to prevent disruptions
        - Challenges
            - Protecting against cyberattacks that can cause widespread outages or water contamination
            - Safeguarding industrial control systems that monitor and manage essential services
        - Example
            - Remote hijacking of operator workstations led to power outages for hundreds of thousands during winter
        - Security Measures
            - Implementation of firewalls, encryption, and regular software updates
            - Conducting regular security audits and employee training for rapid response to attacks
    - Transportation
        - Definition

        - Sector involving critical infrastructure such as rail networks, air traffic control systems, and public transit
    - Challenges
        - Reliance on digital systems for monitoring and communication, making them vulnerable to cyberattacks
        - Potential for delays, accidents, and endangering passenger safety
    - Example
        - The 2017 NotPetya ransomware attack on Maersk disrupted global shipping operations, costing an estimated $300 million and delaying goods worldwide
    - Security Measures
        - Investment in network segmentation, intrusion detection systems, and strong access controls
- Healthcare
    - Definition
        - Sector responsible for managing sensitive patient data and critical medical devices and hospital networks
    - Challenges
        - Vulnerability to cyberattacks that can compromise patient care and personal health information
    - Example
        - The WannaCry ransomware attack in 2017 affected healthcare systems globally, including the UK's NHS, causing the cancellation of approximately 19,000 medical appointments
    - Security Measures

            - Prioritization of securing medical devices, encrypting patient data, and implementing multi-factor authentication
            - Regular updates and vulnerability patching to prevent exploitation of system weaknesses
    - Summary
        - Critical Services Challenges
            - Encompass significant cybersecurity threats faced by utilities, transportation, and healthcare sectors
        - Utilities
            - Must protect essential services from cyber threats to prevent outages and contamination
        - Transportation
            - Relies on secure digital infrastructure to maintain safety and operational efficiency
        - Healthcare
            - Highly vulnerable, requiring robust measures to protect sensitive data and ensure uninterrupted patient care

- **Commercial and Government Challenges**
    - Commercial and Government Challenges
        - Security issues related to protecting sensitive operations and data in sectors like manufacturing, finance, and government/defense against sophisticated threats and vulnerabilities
    - Manufacturing
        - Definition

                - Sector focused on production processes involving industrial control systems (ICS) and intellectual property
            - Challenges:
                - Protecting ICS to maintain operational continuity and prevent disruptions to the supply chain
                - Safeguarding intellectual property such as product designs and trade secrets from cyber espionage
            - Example
                - The 2019 LockerGoga ransomware attack on Norsk Hydro disrupted operations, forcing a shift to manual processes and resulting in over $40 million in losses
            - Security Measures
                - Deployment of strong encryption methods and access controls to protect intellectual property
                - Securing connected devices and sensors through updated firmware and monitoring for unusual traffic patterns
        - Financial Sector
            - Definition
                - Sector responsible for processing financial transactions and managing customer data
            - Challenges
                - Securing transaction systems to prevent financial loss and maintain customer trust
                - Protecting sensitive customer data against fraud and breaches while complying with strict regulations
            - Example

        - The 2016 Bangladesh Central Bank attack involved exploiting vulnerabilities in the SWIFT payment system, resulting in attempted theft of nearly $1 billion
    - Security Measures
        - Adherence to regulations such as the General Data Protection Regulation (GDPR) and Payment Card Industry Data Security Standard (PCI DSS)
        - Implementing strong security protocols and monitoring systems to detect suspicious transactions
- Government/Defense
    - Definition
        - Sector focused on national security, classified information, and military operations
    - Challenges
        - Protecting sensitive government data from state-sponsored cyberattacks and espionage
        - Securing communication networks critical for military operations and national defense
    - Example
        - The 2020 SolarWinds cyberattack allowed hackers to infiltrate multiple U.S. government agencies, exposing sensitive information and national security secrets
    - Security Measures
        - Investment in cybersecurity strategies to protect military assets and communications

- Implementation of strict protocols for managing third-party software to prevent supply chain vulnerabilities

- Summary
  - Commercial and Government Challenges
    - Encompass significant cybersecurity threats faced by manufacturing, financial, and government/defense sectors
  - Manufacturing
    - Requires protection of ICS and intellectual property to maintain business operations and competitive edge
  - Financial Sector
    - Focuses on securing transaction systems and customer data while adhering to regulatory requirements to prevent fraud
  - Government/Defense
    - Challenged by state-sponsored attacks targeting classified information and critical infrastructure, necessitating robust cybersecurity measures

Automated Security Operations

Objective 3.6: Use automation to secure the enterprise

- **Vulnerability Scanning and Reporting**
    - Vulnerability Scanning and Reporting
        - The systematic process of identifying and documenting security weaknesses in systems and networks, using tools like Tenable.io and QualysGuard
        - This allows for the timely remediation of vulnerabilities, improving security posture and supporting compliance efforts
    - Vulnerability Scanning
        - Definition
            - An automated process to detect security weaknesses in systems and networks before they can be exploited
        - Purpose
            - Helps organizations detect issues such as outdated software, misconfigurations, and exposed services that may present security risks
        - Example
            - Universal Chauvet performs weekly scans with Tenable.io, identifying and categorizing vulnerabilities like outdated TLS protocols on web servers
        - Process
            - Scanning
                - Automated tools probe for known vulnerabilities

- Detection
  - Finds issues like unpatched software or weak configurations
- Report Generation
  - Organizes vulnerabilities based on severity, providing actionable insights

- Reporting
  - Definition
    - The process of generating structured, detailed reports based on vulnerability scan results, often including prioritization and remediation recommendations
  - Purpose
    - Assists IT teams in understanding security risks, prioritizing response, and planning remediation efforts
  - Example
    - Tenable.io's report for Universal Chauvet highlights a critical issue with outdated TLS, advising an update to TLS 1.2 or TLS 1.3 to improve security
  - Features
    - Severity Levels
      - Ranges from low to critical, based on potential impact and urgency
    - Actionable Recommendations
      - Offers specific guidance on remediation steps for each vulnerability
    - Follow-Up Scans

        - Confirms vulnerabilities have been resolved after remediation
- Risk Prioritization
    - Definition
        - The process of ranking vulnerabilities by severity to allocate resources effectively
    - Purpose
        - Ensures critical vulnerabilities are addressed promptly while lower-severity issues are handled as resources allow
    - Example:
        - Universal Chauvet prioritizes addressing a critical vulnerability on a public-facing server over a low-severity issue on an internal system
    - Benefits
        - Optimizes resource use, focuses attention on high-risk areas, and enhances overall security management
- Compliance and Documentation
    - Definition
        - Using scan reports to support regulatory compliance, risk assessments, and documentation for audits
    - Purpose
        - Demonstrates active security efforts to regulators and auditors, meeting compliance requirements
    - Example

            - Universal Chauvet's reports help meet PCI DSS and GDPR standards, providing evidence of ongoing vulnerability management
        - Uses
            - Regulatory Compliance
                - Helps meet industry standards by documenting security practices
            - Audit Preparation
                - Shows auditors a record of active vulnerability monitoring
            - Risk Assessment
                - Assists leadership in evaluating security posture and making informed decisions
    - Summary
        - Vulnerability scanning and reporting is essential for maintaining system security by identifying and categorizing security weaknesses before they can be exploited
        - The scanning process detects vulnerabilities and generates reports that prioritize remediation efforts based on severity, while also supporting compliance, risk management, and ongoing security evaluations

- **Scripting**
    - Scripting
        - Writing code that automates repetitive tasks and processes in security management, enhancing efficiency and consistency
    - Pseudocode
        - Definition

            - A high-level outline of a script’s logic and steps without focusing on syntax
        - Purpose
            - Serves as a blueprint for planning scripts, allowing visualization of the overall structure
        - Example
            - Outlining steps to scan network traffic for anomalies before actual coding
        - Importance
            - Helps break down complex tasks into manageable steps
            - Facilitates communication with team members unfamiliar with specific languages
            - Reduces errors during actual coding by clarifying logic in advance
        - Function
            - Acts as a translator between human logic and machine execution, ensuring structured code flow
    - Data Structures
        - Definition
            - Methods of organizing and storing data in scripts to facilitate efficient processing
        - Common Types
            - Arrays/Lists
                - Store sequences of data (e.g., IP addresses, log entries)
            - Dictionaries
                - Store data as key-value pairs for efficient retrieval
        - Usage

        - Enable scripts to access, update, or iterate over data efficiently. Suitable for specific tasks based on the nature of the data
    - Example
        - Using a dictionary to associate IP addresses with their statuses
- Control Structures
    - Definition
        - Constructs that guide the logical flow of a script, similar to traffic signals
    - Types
        - Loops
            - Repeat actions based on conditions (e.g., “for” or “while” loops)
        - Conditional Statements
            - Make decisions based on data (e.g., “if,” “else”)
    - Importance
        - Allow scripts to adapt to varying conditions and automate responses
        - Help in processing large datasets or recurring tasks efficiently
    - Example
        - Using a loop to scan login attempts for suspicious activity. Implementing a conditional statement to clear temporary files if disk usage exceeds 90%
- Summary
    - Scripting
        - Enhances automated security operations by streamlining repetitive tasks for better efficiency

        - Pseudocode
            - Aids in planning by outlining the script's logic without technical details
        - Data Structures
            - Organize and manage information, ensuring scripts can process data effectively
        - Control Structures
            - Direct the script's flow, enabling it to make decisions and repeat tasks automatically

- **Bash**
    - Bash (Bourne Again Shell)
        - A Unix shell and command language for writing scripts to automate system management and security tasks
    - Bash Data Structures
        - Variables
            - Store single values without the need for keywords
        - Arrays
            - Store multiple values in a single variable, accessed with zero-based indexing
        - Strings
            - Represent sequences of characters and can be manipulated with special operators
    - Bash Control Structures
        - Conditional Statements
            - if, elif, and else

            - Evaluate conditions to execute specific code blocks
    - Loops
        - for loop
            - Iterates over lists or arrays to automate repetitive tasks
        - while loop
            - Executes as long as a condition is true
- Case Statements
    - Useful for multiple condition evaluations, serving as an alternative to nested if statements
- Summary
    - Bash automates system tasks with data structures (variables, arrays, strings) and control structures (loops, conditional statements)
    - Data Structures store data for organized access and processing
    - Control Structures manage script flow, enabling repeated actions and decision-making

- **PowerShell**
    - PowerShell
        - A powerful scripting language and command-line shell designed for automating tasks and managing configurations in Windows environments
    - PowerShell Data Structures
        - Variables
            - Used to store single values (e.g., file paths, user details)
            - Assigned with $variableName = value, where no spaces are allowed around =
        - Arrays

            - Store lists of items, accessed by zero-based indexing
        - Hash Tables
            - Store key-value pairs for organized data access
        - Objects
            - Represent structured data with properties and methods
            - PowerShell cmdlets often output objects, making data easy to filter and manipulate
    - PowerShell Control Structures
        - Conditional Statements
            - if-else
                - Used to execute code blocks based on true or false conditions
        - Loops
            - for loop
                - Executes a block of code multiple times
            - foreach loop
                - Iterates over each element in an array or collection
        - Switch Statement
            - Matches a variable to multiple possible values, simplifying complex conditions
    - Summary
        - PowerShell combines data structures (variables, arrays, hash tables, and objects) with control structures (if-else, for, foreach, and switch) to automate complex tasks and manage configurations
        - Data structures organize information, while control structures enable decision-making and task repetition

        - Cmdlets, such as Get-EventLog and Write-Host, interact with system data, making PowerShell ideal for Windows system administration and automation

- **Python**
    - Python
        - A versatile programming language widely used for scripting, automation, and data manipulation across various platforms and operating systems
    - Python Data Structures
        - Lists
            - Store ordered collections of items (can contain various data types)
            - Accessed by index position (zero-based indexing)
        - Tuples
            - Similar to lists but immutable (values cannot be changed once set)
            - Used for data that should remain constant
        - Dictionaries
            - Store key-value pairs for efficient data retrieval
            - Useful for mapping unique keys to specific values
        - Sets
            - Store unique elements in an unordered collection (no duplicates). Automatically removes duplicate values
    - Python Control Structures
        - Conditional Statements
            - if, elif, else statements guide decision-making based on conditions
        - Loops
            - for loop

            - Iterates over a sequence like a list or string
        - while loop
            - Continues as long as a specified condition is true
    - Break and Continue
        - break
            - Exits a loop prematurely when a condition is met
        - continue
            - Skips the current iteration and moves to the next
- Summary
    - Python is widely used for automation due to its flexibility, robust data structures (lists, tuples, dictionaries, and sets), and intuitive control structures (if-else, for, while, break, and continue)
    - Data structures allow for efficient organization and manipulation of data, supporting a range of applications from log analysis to security assessments
    - Control structures help Python scripts automate complex tasks by enabling decision-making and handling repetitive actions
    - Its platform compatibility and ease of use make Python an essential tool for automating tasks, handling data, and executing scripts across different environments

- **Cron/Scheduled Tasks**
    - Cron and Scheduled Tasks
        - Tools for automating the execution of scripts and commands at predefined times on Linux and Windows operating systems, respectively
    - Cron (Linux)

- Definition
    - A time-based job scheduler on Linux systems that allows users to automate repetitive tasks by specifying the exact time and frequency of execution
- Controlled by
    - crontab file, where users define tasks to be scheduled
- Crontab Format
    - Uses five fields to set time and frequency
        - Minute (0-59)
        - Hour (0-23)
        - Day of the month (1-31)
        - Month (1-12)
        - Day of the week (0-7, where both 0 and 7 represent Sunday)
    - Example
        - 20 20 * * 1-5 /scripts/backup-server (runs a script every weekday at 8:20 p.m.)
- Uses
    - Scheduling system maintenance tasks, such as
        - Running security scans
        - Updating system logs
        - Performing backups
    - Automating security tasks, such as integrity checks to detect unauthorized file modifications
- Benefits
    - Reduces need for manual intervention

        - Ensures consistent execution of critical tasks, improving security and stability
- Scheduled Tasks (Windows)
    - Definition
        - A time-based job scheduler in Windows systems that automates tasks by setting exact time and frequency for execution
    - Managed by
        - Windows Task Scheduler
    - Scheduling Options
        - Can set tasks to run
            - Daily
            - Weekly
            - Monthly
        - Example
            - Scheduling system backups every day at a specified time
    - Uses
        - Automating maintenance tasks, such as
            - Running PowerShell scripts
            - System backups
            - Security scans and software updates
    - PowerShell Integration
        - Can automate workflows such as
            - Scanning logs for errors
            - Sending alerts for potential issues
            - Backing up files automatically
    - Benefits

        - Ensures critical tasks run consistently
        - Reduces administrative workload
        - Enhances system security and stability by minimizing the chance for human error
  - Summary
    - Cron and Scheduled Tasks are tools for automating system maintenance and management tasks on Linux and Windows systems, respectively
    - Cron
      - Schedules tasks on Linux, with flexibility in time and frequency settings using crontab, making it ideal for tasks like security scans and log updates
    - Scheduled Tasks
      - Schedules tasks on Windows through Task Scheduler, useful for tasks like backups, updates, and maintenance scripts, often integrated with PowerShell for automation
    - Benefits
      - Both tools reduce manual intervention, minimize human error, maintain security, and ensure consistent execution of system-critical tasks

- **Workflow Automation**
  - Workflow Automation
    - A method of streamlining and automating multi-step security processes to improve efficiency, consistency, and accuracy in threat detection, incident response, and reporting, minimizing the need for manual intervention

- Benefits of Workflow Automation
  - Speed
    - Workflow automation executes tasks instantly, which is crucial in security events where rapid response can prevent further damage
    - Example
      - Upon detecting unusual activity, an automated workflow can isolate affected systems, notify the security team, and generate reports—all within seconds
  - Consistency
    - Automated workflows follow standardized instructions every time, eliminating inconsistencies due to human error or variations in response
    - Example
      - An automated response checklist ensures that each step in a security alert (e.g., system isolation, threat scanning) is completed accurately in sequence
  - Efficiency in Repetitive Tasks
    - Repetitive tasks like logging, scanning, and report generation are automated, freeing security teams to focus on complex issues
    - Example
      - Routine event logging and threat scanning run automatically, reducing the manual workload on the team
- Identifying Tasks for Automation
  - Ideal Candidates
    - Repetitive Tasks

            - Tasks that are performed frequently and require consistent execution, such as logging security events
        - Time-Sensitive Tasks
            - Tasks where immediate action is needed, such as isolating compromised systems
        - Tasks Prone to Human Error
            - Processes where manual actions could introduce errors, like data entry in threat analysis
        - Predictable Tasks
            - Actions that follow the same steps each time, making them reliable for automation
    - Non-Ideal Candidates
        - Complex tasks requiring human judgment or adaptability, which may not benefit from automation
- Workflow Automation Tools and Applications
    - Security Information and Event Management (SIEM)
        - Tools like Splunk can automatically analyze threat data, alert teams to potential threats, and generate comprehensive reports
    - Endpoint Detection and Response (EDR)
        - Tools like CrowdStrike automatically detect and block suspicious activity, log each action, and create detailed reports
    - Intrusion Detection Systems (IDS)
        - Automatically monitor network traffic for suspicious patterns, notifying teams of potential threats without manual inspection
- Comprehensive Documentation

        - Automated workflows log each action taken, including details of blocked activities, isolated systems, and reports generated
        - Benefit
            - This documentation provides a complete record for post-event analysis, enabling teams to refine and improve processes over time
    - Summary
        - Workflow automation streamlines multi-step security processes by enabling tasks to run automatically, reducing response time and minimizing errors
        - It’s particularly effective for repetitive, time-sensitive tasks, allowing security teams to focus on more complex issues
        - Automation tools (e.g., SIEM, EDR) enable rapid threat detection and response, ensuring a consistent and documented approach to managing security events

Integrated Security and Automation

Objective 3.6: Use automation to secure the enterprise

- **Configuration Files**
  - Configuration Files
    - Essential for defining settings and parameters that guide automated processes and security controls within an environment
    - Common file formats include YAML, XML, JSON, and TOML
  - YAML (Yet Another Markup Language)
    - Description
      - Known for readability, using indentation rather than brackets to define structure
    - Use Case
      - Ideal for complex configurations in cloud deployment (e.g., defining resources, services, security groups)
  - XML (eXtensible Markup Language)
    - Description
      - Structured with nested tags to clearly represent hierarchy, suitable for detailed configurations
    - Use Case
      - Frequently used in web services for security policies and access control settings
  - JSON (JavaScript Object Notation)
    - Description

        - Lightweight, using key-value pairs and arrays for simple, readable configurations
    - Use Case
        - Common for API configurations where quick data processing is needed
- TOML (Tom's Obvious, Minimal Language)
    - Description
        - Simple, bracket-based structure, ideal for small to medium configurations
    - Use Case
        - Used in tools prioritizing simplicity, like monitoring setups
- Summary
    - YAML
        - Readable, indentation-based format, ideal for complex cloud configurations
    - XML
        - Structured with nested tags, suited for detailed enterprise security policies
    - JSON
        - Lightweight, key-value format, efficient for APIs and data interchange
    - TOML
        - Simple bracketed layout, best for smaller configurations like monitoring setups

- **Automated Patching**
  - Automated Patching
    - The process of automatically applying software updates and security patches to systems and applications without manual intervention
    - This approach leverages scheduling and triggering mechanisms to ensure timely updates based on criteria like vulnerability severity or a predefined maintenance window
  - Automated Patching Process
    - Definition
      - Applying software updates (new features or improvements) and patches (security fixes) to systems automatically, without manual intervention
    - Scheduling and Triggering
      - Automates updates based on a schedule or triggers updates based on criteria like vulnerability severity
    - Example
      - Deploying critical security updates overnight to avoid business-hour disruptions
  - Patch Management Program
    - Purpose
      - Secures workstations and servers by adopting a consistent patching strategy rather than planning each patch as an isolated event
    - Patch Management Tools
      - Example

            - Microsoft SCCM (System Center Configuration Manager) automates patching across networks, reducing manual effort
        - Selective Manual Patching
            - Some patches may still be applied manually for sensitive systems to prevent unexpected disruptions
- Risks of Delayed Patching
    - Reverse Engineering
        - Hackers often reverse-engineer patches to exploit unpatched vulnerabilities quickly
    - Examples of Patches
        - Hotfix/Critical Update
            - Immediate fixes for actively exploited vulnerabilities
        - Updates
            - Usually add features but can sometimes introduce new vulnerabilities
- Service Packs and Continuous Updates
    - Service Packs
        - Traditionally cumulative bundles of updates, now replaced by continuous updates in some systems
    - Windows 10 Update Model
        - Delivers monthly cumulative updates and bi-annual feature updates to ensure up-to-date security and functionality
- Establishing a Patch Management Program
    - Responsibility

                - Designate a team or individual to track vendor-released security patches and firmware updates
            - Patching Mechanisms
                - Automated Patching
                    - Preferred method, though manual updates may be necessary for certain systems
                - Prioritization and Testing
                    - Patch Priority Levels
                        - Urgent, important, non-critical
                    - Staging Environment
                        - Test all patches in a non-production environment to minimize potential disruptions
            - Periodic Implementation
                - Example
                    - Schedule non-critical patches bi-weekly for consistent, manageable updates
        - Summary
            - Automated Patching
                - Essential for maintaining system security by applying updates without manual intervention
            - Difference between Updates and Patches
                - Updates add functionality, while patches fix security vulnerabilities; both are necessary
            - Scheduling and Triggering
                - Uses maintenance windows and vulnerability severity to apply patches at optimal times

        - Patch Management Program
            - Prioritizes and tests patches in a staging environment before production to prevent disruptions

- **Dynamic Security Controls**
    - Dynamic Security Controls
        - Adaptive security measures that automatically adjust settings in response to changing conditions or detected threats, providing continuous protection
        - Key components include event-based triggers and auto-containment
    - Event-Based Triggers
        - Definition
            - Act as "alarm bells" that monitor for specific conditions (e.g., detected malware, unusual network patterns) and activate security responses upon detection
        - Examples of Detection
            - Unauthorized login attempts
            - Unusual data transfers
            - Rapid location changes in user logins
        - Function
            - Tools such as Intrusion Detection Systems (IDS) and Security Information and Event Management (SIEM) platforms monitor and alert on suspicious activity
            - Automated responses may include multi-factor authentication, account access blocks, or re-authentication requirements
        - Adaptability

        - Event-based triggers update with new threat intelligence, recognizing newly discovered malware signatures and quickly adapting to new threat patterns
    - Automation Benefits
        - Automation tools like orchestration platforms streamline responses, triggering pre-set workflows to respond to threats in real time without relying solely on human intervention
- Auto-Containment
    - Definition
        - Automatically isolates potentially harmful activities or entities from the network to prevent infection spread
    - Analogy
        - Like a hospital quarantine, where suspicious files or devices are isolated to avoid spreading risk
    - Example
        - A device sending unusual data to an unknown external source is automatically isolated, restricting its network interaction
        - Isolated devices can still be analyzed by security teams to assess if the behavior was a false positive or a true threat
    - Operational Benefits
        - Auto-containment is ideal in high-traffic environments, isolating threats quickly without interrupting broader network productivity
        - Minimizes the risk of network-wide infection, allowing secure analysis and remediation without impacting other users
- Summary
    - Dynamic Security Controls

            - Automated adjustments to security settings based on real-time threat detection and behavioral changes
        - Event-Based Triggers
            - Automatically activate responses to detected conditions, like malware signatures or suspicious logins, providing swift defense against potential threats
        - Auto-Containment
            - Isolates suspicious files or devices to prevent infection spread, allowing for secure investigation while maintaining network stability

- **Security Orchestration, Automation, and Response (SOAR)**
    - SOAR Platforms
        - Tools and processes designed to streamline and automate security operations, enabling organizations to efficiently detect, respond to, and mitigate threats
        - Key components include playbooks and runbooks
    - Playbooks
        - Definition
            - Predefined workflows in incident response plans, outlining specific actions to take during various types of security incidents
        - Function
            - Serve as step-by-step guides, detailing standardized response actions for incidents like DDoS attacks, malware infections, phishing, and data exfiltration

            - Act as checklists, ensuring consistent and timely response actions by security teams
        - Examples
            - Phishing Playbook
                - A 10-page guide based on the NIST incident response process, including phases like preparation, detection, analysis, and recovery
                - Preparation
                    - Form the core team, review timelines, and conduct interviews
                - Detection
                    - Categorize incidents and initiate response actions
                - Analysis
                    - Define risk factors and triage accordingly
            - Resources
                - Incident Response Consortium
                    - Templates available for customization
                - Microsoft Technical Playbooks
                    - Provides technical guidance for responses to phishing, password attacks, and more
        - Benefits
            - Enhance response speed and consistency by providing structured workflows
            - Modern SOAR platforms use AI and machine learning to improve response accuracy and adapt dynamically to emerging threats
    - Runbooks

- Definition
  - Automated implementations of playbooks that perform incident response tasks with periodic checkpoints for human intervention
- Function
  - Runbooks automate routine response tasks, freeing analysts to focus on complex issues
  - Automate steps like isolating affected workstations, running scans, and re-imaging devices, with pauses for human review as needed
- Examples of Implementation
  - Phishing Response Runbook
    - Automatically deletes malicious emails, isolates affected workstations, performs scans, and prompts analyst for re-imaging confirmation
  - Ransomware Response Runbook
    - Quickly isolates infected systems, identifies stakeholders, and retains encryption keys for forensic analysis
  - Data Exfiltration Runbook
    - Analyzes compromised data sources, performs forensic analysis, and detects lateral movement across the network
- Common Threats for Runbooks
  - Ransomware
    - Involves steps to isolate, retain encryption keys, and quickly disconnect infected systems
  - Data Exfiltration
    - Guides actions to detect and mitigate data leaks from SQL injections, compromised accounts, and other methods

            - Social Engineering
                - Detects and addresses phishing attempts, identifies affected users, and includes steps for password resets and re-imaging
    - Summary
        - SOAR Platforms
            - Provide automated and orchestrated responses to security incidents through tools like playbooks and runbooks
        - Playbooks
            - Detailed workflows that guide teams through incident responses with predefined steps for specific threat types
        - Runbooks
            - Automate the execution of playbook actions, incorporating checkpoints for analyst input and freeing resources for complex threat analysis

- **Cloud Automation**
    - Cloud Automation
        - The use of tools and processes to manage, deploy, and secure cloud resources and applications efficiently
    - Containerization
        - Definition
            - A virtualization method where the host operating system (OS) creates isolated environments (containers) for each application, ensuring consistent operation across different OS environments
        - Isolation Level

            - Containers operate at the OS level, unlike traditional virtualization, which isolates resources at the hardware level
        - Containerization Engine
            - Uses engines (e.g., Docker, Kubernetes) instead of hypervisors (found in traditional virtualization) to create and manage containers
        - How It Works
            - Containers share the host OS and isolate only applications and dependencies within each container, making them lightweight and efficient
            - Applications within containers are logically separated, enhancing security
            - Containers include all dependencies needed for an application, ensuring it runs consistently on any host
        - Benefits
            - Increased resource efficiency compared to virtual machines. Enhanced security through isolation
            - Consistency in application performance across multiple environments
    - Infrastructure as Code (IaC)
        - Definition
            - A practice where infrastructure components (e.g., networks, servers) are defined and managed as code, enabling automated, consistent, and repeatable deployments
        - Languages and Tools

            - Uses languages such as JSON and YAML, with tools like Terraform, AWS CloudFormation, and Ansible
        - How IaC Differs from Traditional Configuration
            - Traditional configuration requires manual setup, which is time-consuming and prone to error
            - IaC automates infrastructure setup and management, turning it into an efficient, consistent, and version-controlled process
        - How It Works
            - Infrastructure setups are defined in configuration files
            - IaC tools interpret the configurations and automatically deploy resources as defined
            - Supports version control, allowing teams to track changes, roll back configurations, and manage resources predictably
        - Benefits: Consistent infrastructure setups across environments. Reduced manual work, minimizing errors. Allows for quick deployment and updates, improving operational efficiency.
    - Summary
        - Cloud Automation
            - Automates the deployment and management of cloud resources to ensure efficiency, consistency, and security
        - Containerization
            - Packages applications with dependencies into isolated units (containers), ensuring consistent performance and security across environments
        - Infrastructure as Code (IaC)

            - Defines infrastructure setups as code, enabling fast, repeatable, and consistent deployments with minimal manual intervention
        - Both concepts enhance operational control, reduce errors, and improve the consistency of cloud resources across various environments

- **Cloud APIs/Software Development Kits (SDKs)**
    - Cloud APIs and SDKs
        - Tools enabling developers to interact with cloud services programmatically
        - They facilitate the automation and integration of security controls within applications, often through functions like webhooks, which trigger actions based on specific events
    - Cloud APIs
        - Purpose
            - Allows developers to interact directly with cloud services using HTTP-based requests, enabling the automation of tasks like virtual machine creation, database management, and workflow automation
        - Features
            - Webhooks
                - User-defined callbacks that listen for specific events and initiate automated actions in real time
            - Event Notifications
                - APIs can send event details to specified URLs, such as when a file is uploaded or modified
        - Example

            - A cloud storage API webhook notifies a designated URL whenever a file changes, sending data such as the file name and modification time
            - This helps maintain visibility and initiate timely responses
        - Analogy
            - Think of webhooks in Cloud APIs as a “digital dispatcher” for cloud events, much like a fire alarm instantly alerts emergency responders
            - Webhooks notify other services as soon as an event occurs, allowing quick responses
    - Software Development Kits (SDKs)
        - Purpose
            - SDKs offer tools and libraries that make integrating cloud functionalities into applications simpler, handling API complexities and providing language-specific functions
        - Features
            - Webhook Management
                - SDKs often include functions to register and manage webhooks, reducing the need for manual setup
            - Ease of Use
                - SDKs streamline the setup and management of webhooks and other functionalities, allowing developers to focus on core tasks
        - Example

            - In an e-commerce app, an SDK can register a webhook to alert customers via email when an order is shipped, handling background requests and error-checking
        - Analogy
            - Using SDKs for webhooks is like setting up automated calendar reminders
            - SDK functions respond to specified events without additional setup, simplifying development
    - Summary
        - Cloud APIs
            - Provide direct access to cloud services, allowing automation and management of resources through HTTP requests
            - Webhooks enable real-time communication between services by triggering actions in response to specific events
        - SDKs
            - Simplify cloud service integration, especially webhook setup, by providing tools to register, verify, and manage notifications in a streamlined manner
        - Overall Role
            - Both Cloud APIs and SDKs streamline workflows, automate tasks, and ensure responsive interactions within cloud environments

- **Vulnerability Management**
    - Vulnerability Management
        - The continuous process of identifying, assessing, and mitigating security vulnerabilities within an organization's systems and applications

        - It involves using standardized tools within the SCAP (Security Content Automation Protocol) framework, such as CPE, CVE, and CVSS, to streamline identification and prioritization
    - Common Platform Enumeration (CPE)
        - Purpose
            - Provides a standardized identifier for IT platforms, including hardware, operating systems, and applications
        - Role in SCAP
            - Enables consistent classification of assets, aiding in the application of the SCAP framework across networks
        - Example
            - A vulnerability management tool uses CPE identifiers to cross-reference affected assets and vulnerabilities, helping security teams pinpoint systems needing updates
        - Analogy
            - Think of CPE as an "ID tag" for IT assets, like a barcode identifying products, providing precise details about hardware, OS, or applications in an environment
    - Common Vulnerabilities and Exposures (CVE)
        - Purpose
            - Assigns unique identifiers to known vulnerabilities, making it easier to access information on each vulnerability
        - Role in SCAP
            - Acts as a universal catalog for cybersecurity teams to track, manage, and share information on vulnerabilities
        - Example

            - CVE-2017-0144, associated with the EternalBlue vulnerability in Windows, highlights a known issue exploited by WannaCry ransomware
            - Security teams use CVE databases to understand vulnerability details and implement necessary patches
        - Analogy
            - The CVE database is like a "universal reference library" for vulnerabilities, similar to a medical database cataloging diseases and treatments for effective diagnosis and management
    - Common Vulnerability Scoring System (CVSS)
        - Purpose
            - Provides a standardized numerical severity score (0.0 to 10.0) to assess the risk level of vulnerabilities
        - Role in SCAP
            - Helps prioritize vulnerabilities based on severity, allowing organizations to allocate resources effectively
        - Example
            - A vulnerability with a score of 9.0+ (critical) requires immediate attention, whereas a score of 4.5 (medium) can be scheduled for later remediation
        - Analogy
            - CVSS serves as a "risk thermometer" for vulnerabilities, much like a triage system in hospitals, where the most severe cases are prioritized first
    - Summary
        - Vulnerability Management

            - Ongoing process of identifying, assessing, and addressing security weaknesses
        - CPE
            - Standardized identifiers for IT assets, aiding in asset classification and vulnerability management
        - CVE
            - Unique identifiers for known vulnerabilities, enabling quick access to information and remediation guidance
        - CVSS
            - Severity scoring for prioritizing vulnerabilities, ensuring critical issues are addressed first to maintain security

- **Security Content Automation Protocol (SCAP)**
    - Security Content Automation Protocol (SCAP)
        - A framework that standardizes the format and automated exchange of information related to security vulnerabilities, configurations, and compliance
        - SCAP components include the Open Vulnerability and Assessment Language (OVAL) and the eXtensible Configuration Checklist Description Format (XCCDF)
    - Open Vulnerability and Assessment Language (OVAL)
        - Purpose
            - Provides a standardized XML format for describing system security states, including configurations and vulnerabilities
        - Role in SCAP

        - Acts as a “universal language” that enables security tools to interpret vulnerability and configuration data consistently, ensuring uniform assessments across diverse tools
    - Example
        - OVAL is integrated into tools like Tenable.io and Qualys to automatically scan systems for vulnerabilities by encoding criteria for each known vulnerability
    - Analogy
        - OVAL is like a set of "universal traffic signs" in cybersecurity, ensuring all tools interpret vulnerability conditions the same way
- eXtensible Configuration Checklist Description Format (XCCDF)
    - Purpose
        - Provides a machine-readable format for creating, defining, and auditing security configuration policies
    - Role in SCAP
        - Facilitates automated compliance checks by enabling tools to scan configurations against best practices or regulatory standards
    - Example
        - XCCDF checklists might specify secure password policies or software configuration requirements, which SCAP-compatible tools can validate to ensure consistent compliance across systems
    - Analogy
        - XCCDF acts as a “recipe” for secure configurations, listing the necessary configuration details to meet compliance requirements
- Summary
    - SCAP

        - A standardized framework for exchanging information on vulnerabilities, configurations, and compliance
    - OVAL
        - Describes system vulnerabilities and configuration states in a standardized format, allowing consistent vulnerability assessments
    - XCCDF
        - Defines security checklists and policies in a machine-readable format, enabling automated compliance checks and streamlined configuration validation

Artificial Intelligence

Objectives:

- 1.5 - Summarize the information security challenges associated with artificial intelligence adoption
- 3.6 - Use automation to secure the enterprise

- **Generative AI**
  - Generative Artificial Intelligence (AI)
    - AI systems that create new content by learning patterns from existing data, enabling them to generate text, images, code, music, and more
    - Key applications include code assistance and automated documentation
  - AI Content Creation
    - Definition
      - The ability of AI to create content by learning patterns and structures from extensive datasets, such as text, images, or code
    - Model Example
      - Generative Pre-trained Transformer (GPT) models, which understand context and language structure through tokenization and attention layers
    - Applications
      - Used to generate coherent, contextually relevant content for business, educational, and creative purposes
    - Analogy

- GPT models process words like parts of a puzzle, where context defines meaning, allowing AI to interpret words accurately, such as understanding "bank" as a riverbank when associated with "fish"

- Code Assist
  - Purpose
    - Assists developers in writing code by offering real-time suggestions and auto-completing syntax, reducing repetitive work and enhancing productivity
  - Tools
    - GitHub Copilot and Tabnine provide code suggestions based on coding standards and best practices
  - Benefits
    - Speeds up coding, improves consistency across projects, and helps less experienced developers learn new practices
  - Analogy
    - Code assist is like an "auto-pilot" for developers, streamlining routine coding tasks and allowing them to focus on problem-solving
- AI-Generated Documentation
  - Purpose
    - Automatically generates technical documents based on existing code, including user guides and release notes, to ensure up-to-date, consistent documentation
  - Tools

            - GitBook and Document360 analyze codebases to create and update documentation aligned with the latest changes
        - Benefits
            - Reduces the burden on developers to manually update documentation, ensuring accuracy and accessibility for users and support teams
        - Analogy
            - AI-generated documentation is like a “smart assistant” that keeps technical documents current, making it easier for teams to access reliable information
    - Summary
        - Generative AI
            - Creates new content by learning from large datasets, enabling applications across content creation, code assistance, and documentation
        - AI Content Creation
            - Uses models like GPT to generate coherent, contextually appropriate content
        - Code Assist
            - Offers real-time code suggestions, enhancing productivity and quality of development
        - AI-Generated Documentation
            - Automatically produces and updates technical documentation, easing the documentation workload for developers

- **Ethical and Governance Considerations**
  - Ethical and Governance Considerations
    - Frameworks and guidelines to ensure that AI development and usage are responsible, aligning with societal values and organizational standards
    - This includes ethical governance of AI and organizational policies for AI use
  - Ethical Governance of AI Purpose
    - Establishes structures to ensure AI usage aligns with societal values, focusing on fairness, transparency, and accountability
    - Example of Success
      - The European Union's AI Act, which classifies AI risk levels, mandates bias testing, and requires documentation and human oversight for high-risk applications like facial recognition
    - Example of Failure
      - The COMPAS algorithm used for recidivism prediction in the U.S., which lacked bias testing and transparency
      - It led to racial disparities due to unaddressed biases in its data, underscoring the importance of governance in high-stakes decisions
    - Key Components
      - Transparency
        - Ensures AI decisions are explainable, with information on data sources and potential biases available for stakeholders
      - Accountability

            - Establishes responsibility for AI outcomes, with protocols to investigate and correct errors, enabling trustworthy and ethical AI deployment
- Organizational Policies on the Use of AI
    - Purpose
        - Defines company-specific rules for responsible AI use, ensuring alignment with ethical standards and legal obligations
    - Example of Success
        - Microsoft's AI ethics principles, which involve expert reviews for high-impact systems to ensure responsible deployment
    - Example of Failure
        - Amazon's AI hiring tool, which exhibited gender bias, demonstrating the need for fairness checks in AI tools to prevent replicating discriminatory patterns
    - Key Components
        - Transparency
            - Clearly defined boundaries help employees understand acceptable AI applications, ensuring ethical data collection and usage
        - Training and Education
            - Provides employees with knowledge on AI ethics, covering data privacy, algorithmic bias, and transparency
        - Guidelines and Continuous Training
            - Reinforces accountability, ensuring AI is used responsibly and aligning with organizational values through ongoing education

- Summary
  - Ethical Governance of AI
    - Establishes frameworks that promote fairness, transparency, and accountability, reducing unintended harm and building public trust
  - Organizational Policies on AI Use
    - Provides specific guidelines and training within companies to ensure AI aligns with ethical standards and legal obligations
  - Transparency and Accountability
    - Essential across both ethical governance and organizational policies, allowing stakeholders to understand AI decision-making and manage risks effectively

- **Legal and Privacy Risks**
  - Legal and Privacy Risks
    - The need for AI systems to comply with laws and regulations that protect individual rights and personal data
    - Key considerations include potential misuse of AI and the distinction between explainable and non-explainable AI models
  - Potential Misuse of AI
    - Purpose
      - Identifies ways AI can be used inappropriately, leading to discrimination, privacy breaches, or harmful consequences
    - Example in Predictive Policing

        - Algorithms analyzing historical crime data may reinforce biases if trained on biased data, leading to over-policing in certain communities and potential discrimination
    - Example in Healthcare
        - Models trained on non-diverse data may not generalize well, potentially resulting in misdiagnoses or inadequate treatment for certain patient demographics, highlighting the need for inclusive datasets
    - Key Concern
        - Misuse of AI can lead to biased outcomes that harm individuals or communities, emphasizing the importance of regulatory oversight and diverse data to prevent such consequences
- Explainable vs. Non-Explainable AI Models
    - Explainable Models
        - Provide clear reasons for decisions, increasing transparency and trust
        - Example
            - A decision tree for loan approvals may show how income or credit history influenced approval, helping meet transparency requirements
        - Importance
            - Helps organizations meet legal transparency requirements, allowing users to understand and, if necessary, challenge AI decisions
    - Non-Explainable (Black-Box) Models

        - Complex, high-performance models like deep learning algorithms, which are difficult to interpret
        - Example
            - In credit scoring, a non-explainable model might deny loans based on hidden patterns, making it difficult to understand the rationale and potentially conflicting with anti-discrimination laws
        - Challenge
            - Balancing accuracy and interpretability, especially in sectors with regulatory requirements like finance and healthcare, where transparency is critical
    - Key Takeaway
        - Explainable models allow for greater legal compliance and trust by offering interpretability, while non-explainable models pose challenges for transparency and accountability
- Summary
    - Potential Misuse of AI
        - Occurs when systems reinforce biases, particularly in sensitive areas like policing and healthcare
        - This underscores the need for regulatory oversight and diverse data
    - Explainable Models
        - Offer clear decision-making processes, aiding in legal compliance and user trust, particularly valuable in regulated sectors
    - Non-Explainable Models

            - Provide high accuracy but limited interpretability, raising legal and ethical concerns
            - Balancing accuracy with transparency is key to responsible AI governance

- **Threats to the Model**
    - Threats to the AI Model
        - Potential attacks or vulnerabilities that can compromise the security, integrity, or confidentiality of AI systems
        - These threats create risks of incorrect decisions, unauthorized access to data, and other harmful outcomes
    - Prompt Injection
        - Definition
            - Manipulating input prompts to alter the model's behavior
        - Example
            - In a chatbot, an attacker crafts a prompt to trick the bot into providing confidential information
        - Mitigation
            - Use prompt filtering and conduct regular model validation
    - Unsecured Output Handling
        - Definition
            - Improper management of AI-generated outputs, leading to potential leaks or execution of harmful commands
        - Example
            - An AI assistant that inadvertently reveals sensitive data through unprotected outputs

        - Mitigation
            - Secure protocols and output monitoring to prevent data leaks
- Training Data Poisoning
    - Definition
        - Attackers manipulate training data to distort model predictions or functionality
    - Example
        - In content moderation, poisoned data could lead a model to approve harmful content or censor safe posts
    - Mitigation
        - Regularly audit and secure training data
- Model Denial-of-Service (DoS)
    - Definition
        - Flooding the model with excessive requests to render it unusable
    - Example
        - Attackers slow down or crash an image recognition model with a high volume of requests
    - Mitigation
        - Use rate limiting and monitor for unusual activity
- Supply Chain Vulnerabilities
    - Definition
        - Risks from compromised third-party components or libraries in AI development
    - Example
        - Using a pre-trained model from an untrusted source that contains embedded malicious code

    - Mitigation
      - Vet and verify third-party components to ensure security
- Model Theft
  - Definition
    - Unauthorized extraction or duplication of a model's intellectual property
  - Example
    - Attackers replicate a proprietary model by querying it for predictions and copying the responses
  - Mitigation
    - Secure models through encryption and access controls
- Model Inversion
  - Definition
    - Attackers reconstruct sensitive data from the model's outputs, exposing personal or proprietary information
  - Example
    - By analyzing an AI model's responses, attackers reverse-engineer training data demographics
  - Mitigation
    - Use differential privacy to obscure individual data points within the broader dataset
- Summary
  - Prompt Injection
    - Manipulation of input prompts; mitigated with prompt filtering
  - Unsecured Output Handling

            - Poor output management leading to data leaks; mitigated with secure protocols
        - Training Data Poisoning
            - Manipulation of training data to alter behavior; mitigated by auditing data
        - Model Denial-of-Service (DoS)
            - Overloading model access; mitigated with rate limiting
        - Supply Chain Vulnerabilities
            - Risks from third-party components; mitigated by verifying sources
        - Model Theft
            - Extraction of model IP; mitigated with encryption and access controls
        - Model Inversion
            - Reconstruction of sensitive data from model outputs; mitigated by applying differential privacy

- **AI-enabled Attacks**
    - AI-enabled Attacks
        - Attacks that use artificial intelligence to enhance the sophistication, scale, and effectiveness of malicious activities, making them harder to detect and defend against
    - Insecure Plugin Design
        - Definition
            - AI plugins are add-ons that extend the capabilities of existing software with artificial intelligence, but insecure plugins can introduce vulnerabilities

        - Example
            - A plugin with weak authentication may allow attackers to gain unauthorized access to sensitive systems
        - Mitigation
            - Use security tools like Burp Suite, ZAP, Snyk, and Dependabot to scan plugins for vulnerabilities and dependency issues, and implement frequent plugin updates
    - AI Pipeline Injectors
        - Definition
            - Injecting malicious data or code into an AI model’s training or deployment pipeline to alter its behavior
        - Example
            - In fraud detection, attackers may label fraudulent transactions as normal, leading the AI model to misclassify actual fraud indicators
        - Mitigation
            - Implement anomaly detection, rigorous data validation, and regular model audits with tools like TensorFlow Privacy, Adversarial Robustness Toolbox, and Shield AI
    - Automated Exploit Generation
        - Definition
            - AI-driven process that identifies and exploits vulnerabilities automatically, accelerating the attack process
        - Example
            - An AI system could scan a network for unpatched software and generate specific code to exploit it instantly
        - Mitigation

            - Proactive patching and automated vulnerability detection with tools like Tenable.io and Qualys to stay ahead of potential exploits
    - Summary
        - Insecure Plugin Design
            - Weak or insecure AI plugins can expose systems to unauthorized access, especially without robust authentication or validation checks
        - AI Pipeline Injectors
            - Malicious data in training or deployment pipelines can lead to compromised AI models that produce incorrect or biased outputs
        - Automated Exploit Generation
            - AI systems can rapidly identify and exploit vulnerabilities, highlighting the need for constant vigilance in patching and security practices

- **AI Manipulation Attacks**
    - AI Manipulation Attacks
        - Intentional alteration or deception of AI systems to produce harmful or misleading outcomes
    - Social Engineering
        - Definition
            - Exploiting human trust through AI-driven interactions to deceive individuals into sharing sensitive information or performing actions
        - How It Works

        - Attackers use AI chatbots or fake social media profiles that mimic real contacts
        - Conversations feel natural due to advanced AI language models, increasing persuasiveness
    - Examples
        - Chatbots impersonating known contacts to collect personal data
        - AI-driven phishing messages tailored to user behavior, enhancing trust and response rates
    - Common Platforms
        - Social media accounts impersonating friends, family, or authority figures
        - Emails and direct messaging apps with automated, personalized phishing attempts
    - Mitigation
        - Verify authenticity of online interactions
        - Use multi-factor authentication (MFA) for accounts
        - Implement user training to recognize phishing and impersonation attempts
- Deepfakes through Digital Media and Interactivity
    - Definition
        - Hyper-realistic videos or audio files that mimic real people, designed to mislead or harm
    - How It Works
        - AI generates fabricated content mimicking public figures or known individuals

        - Interactive deepfakes allow real-time manipulation (e.g., live video calls with fake voices or faces)
    - Examples
        - Fake video of a CEO endorsing a fraudulent scheme, spreading misinformation
        - Real-time video conferencing where attackers impersonate participants using face-swapping and voice-cloning technology
    - Risks
        - Damage to reputation
        - Spread of misinformation
        - Increased fraud during business meetings or public broadcasts
    - Mitigation
        - Implement advanced verification tools (e.g., biometric authentication)
        - Use AI-based detection tools to analyze video and audio content
        - Raise awareness and train employees on recognizing signs of deepfake technology
- Summary
    - AI Manipulation Attacks use advanced technologies to exploit trust, deceive individuals, and spread misinformation
    - Social Engineering
        - Uses AI chatbots or fake profiles to create convincing interactions
        - Focuses on gathering sensitive information or influencing actions
    - Deepfakes
        - Creates false but realistic video or audio content

            - Delivered through digital platforms or live interactions for maximum deception
        - Defense Strategies
            - Verify authenticity in communications. Use advanced AI detection tools for content verification
            - Train individuals and organizations to recognize manipulation techniques
        - By understanding these threats and implementing proper defenses, organizations can reduce risks associated with AI-driven deception

- **AI Usage Risks**
    - AI Usage Risks
        - Potential dangers and unintended consequences that arise from the implementation and reliance on artificial intelligence systems, which can impact decisions, data privacy, and security
    - Overreliance on AI
        - Definition
            - Placing excessive trust in AI output without verifying its accuracy or considering additional contextual factors
        - Example
            - In finance, analysts relying solely on AI recommendations may overlook critical market trends, leading to suboptimal investment decisions
        - Mitigation
            - Combine AI insights with human expertise to ensure decisions are both data-informed and contextually grounded

- Sensitive Information Disclosure To the Model
  - Definition
    - Risk of exposing private, confidential, or proprietary data by inputting it into AI systems without robust privacy controls
  - Example
    - A customer service AI that receives sensitive data without encryption or access controls may inadvertently expose it, leading to compliance risks
  - Mitigation
    - Implement data anonymization, secure access protocols, and clear data usage guidelines to protect sensitive information
- Sensitive Information Disclosure From the Model
  - Definition
    - Unintentional release of private data that an AI model has learned during its training or operational use
  - Example
    - An AI assistant in customer service might inadvertently reveal past interactions, like account balances or transactions, to unauthorized users
  - Mitigation
    - Use differential privacy techniques and strict data handling protocols to prevent sensitive information from being disclosed
- Excessive Agency of the AI
  - Definition

        - Allowing AI systems too much autonomy, which can lead to unsupervised decisions that may misalign with organizational or ethical standards
    - Example
        - Self-driving cars making all driving decisions could encounter situations they weren't trained for, risking harmful consequences
    - Mitigation
        - Implement human-in-the-loop (HITL) systems, structured workflows, and decision auditing to ensure human oversight in critical decision-making processes
- Summary
    - Overreliance on AI
        - Excessive trust in AI can lead to decisions that ignore critical context; combining AI with human judgment mitigates this risk
    - Sensitive Information Disclosure To the Model
        - Inputting sensitive data without privacy controls can expose organizations to compliance risks; anonymization and access controls are essential
    - Sensitive Information Disclosure From the Model
        - AI models may inadvertently release private data they encountered during training; differential privacy helps prevent these disclosures
    - Excessive Agency of the AI
        - AI autonomy without supervision can lead to misaligned decisions; oversight mechanisms ensure AI actions align with human values and safety standards

- **AI Bots**
  - AI Bots
    - Automated software programs that interact with users or systems to perform specific tasks, often as digital assistants or workers
    - Key components for secure operation include Access and Permissions, Guardrails, Data Loss Prevention (DLP), and Disclosure of AI usage
  - Access and Permissions
    - Definition
      - Controls that limit what data and systems an AI bot can access, ensuring it only interacts with authorized information
    - Example
      - A healthcare bot with access only to general medical info to answer questions, but no access to specific patient records
    - Benefit
      - Protects data integrity and user privacy by restricting bot access within defined roles
  - Guardrails
    - Definition
      - Predefined boundaries that restrict an AI bot's actions, keeping it within safe operational limits
    - Example
      - In an online store, a recommendation bot is limited to suggesting in-stock products only.
    - Benefit
      - Ensures the bot provides accurate and reliable information, maintaining user satisfaction and safety

- Data Loss Prevention (DLP)
  - Definition
    - Security measures, like encryption and access policies, to prevent bots from leaking or misusing sensitive data
  - Example
    - A banking chatbot restricts sharing financial details without verifying the user's identity
  - Benefit
    - Safeguards sensitive data, protecting user trust and ensuring compliance with privacy regulations
- Disclosure of AI Usage
  - Definition
    - Informing users that they are interacting with an AI bot, fostering transparency and setting user expectations
  - Example
    - A customer service chatbot stating, "I am an AI assistant here to help you," at the start of a conversation
  - Benefit
    - Builds trust and clarity in interactions, as users know they're communicating with AI, not a human
- Summary
  - Access and Permissions
    - Restrict bots to authorized data and tasks, preventing unauthorized access or actions.
  - Guardrails

- Set operational limits, guiding bots to perform their tasks accurately and safely

- Data Loss Prevention (DLP)
    - Prevents bots from accessing or leaking sensitive data, protecting user privacy
- Disclosure of AI Usage
    - Informs users they’re interacting with AI, promoting transparency and trust

Vulnerabilities and Attacks

Objectives:

- 3.4 - Implement hardware security technologies and techniques
- 4.2 - Analyze vulnerabilities and attacks and recommend solutions to reduce the attack surface

- **Injection Vulnerabilities**
    - Injection vulnerabilities
        - Occur when attackers insert malicious code through input fields or interfaces, enabling unauthorized actions or access
    - Injection
        - Definition
            - Attackers execute arbitrary commands on the host operating system via an application's interface
        - Example
            - Malicious input
                - diontraining.com; rm -rf /important/data
                - This input runs the intended ping command and a destructive deletion command
        - Impact
            - System compromise, data deletion, privilege escalation
        - Mitigation
            - Strict input validation (e.g., allowing only IPs or domains). Whitelisting acceptable commands

- Code Injection
  - Definition
    - Malicious code is inserted into an application and executed with its permissions
  - Impact
    - Application-level compromise, unauthorized actions, data exposure
  - Mitigation
    - Avoid evaluating user inputs directly (e.g., avoid eval())
    - Validate and sanitize all inputs
- Cross-Site Scripting (XSS)
  - Definition
    - Injected scripts execute in users' browsers
  - Types
    - Reflected XSS
      - Script injected into URL or form and reflected back
      - Example
        - <script>alert('You have been hacked!')</script>
    - Stored XSS
      - Malicious script stored on the server and executed when the page loads
      - Example
        - Script in a comment field affecting all users
    - DOM-Based XSS
      - Client-side script modifies the DOM to execute
      - Example

            - Malicious link modifies document.write
        - Impact
            - Session hijacking, data theft, redirection to malicious sites
        - Mitigation
            - Validate and encode inputs
            - Use Content Security Policies (CSPs)
            - Avoid unsafe JavaScript methods (e.g., document.write)
    - Cross-Site Request Forgery (CSRF)
        - Definition
            - Tricks users into performing actions on a trusted site while authenticated
        - Example
            - A crafted link transfers money on behalf of the user without their consent
        - Impact
            - Unauthorized actions, data integrity compromise
        - Mitigation
            - Use anti-CSRF tokens
            - Require re-authentication for sensitive actions
    - Server-Side Request Forgery (SSRF)
        - Definition
            - Forces a server to make unauthorized requests to internal/external resources
        - Example
            - Request to internal metadata server in cloud environments to access sensitive data

    - Impact
        - Access to internal systems, data exfiltration, resource manipulation
    - Mitigation
        - Validate and restrict allowable destinations for server requests. Enforce strict access controls
- Deserialization
    - Definition
        - Exploits the process of reconstructing data, injecting malicious content
    - Example
        - Injecting harmful objects during deserialization that execute unauthorized code
    - Impact
        - Remote code execution, privilege escalation, data corruption
    - Mitigation
        - Avoid deserializing untrusted data. Use secure serialization formats. Restrict object types allowed during deserialization
- Summary
    - Injection vulnerabilities exploit weaknesses in input validation or processing to compromise systems
    - Effective mitigation includes strict input validation, avoiding dynamic code execution, and implementing security best practices to detect and block malicious activities

- **Memory-related Vulnerabilities**
    - Memory-related Vulnerabilities
        - Flaws in how an application manages memory, potentially allowing attackers to execute malicious code, cause crashes, or leak sensitive information
        - Common vulnerabilities include Deprecated Functions, Unsafe Memory Utilization, Overflows, Race Conditions, and Time of Check to Time of Use (TOCTOU)
    - Deprecated Functions
        - Definition
            - Older, insecure programming functions that are still in use but lack modern security features
        - Example
            - The gets() function in C, which does not check buffer size, making it vulnerable to buffer overflow attacks
        - Mitigation
            - Replace deprecated functions with secure alternatives, like using fgets() instead of gets() for buffer size control
    - Unsafe Memory Utilization
        - Definition
            - Poor memory management practices that can lead to issues like buffer overflows and memory leaks
        - Example
            - Memory leaks from unreleased allocations cause systems to slow or crash over time
        - Protection Techniques

        - ASLR (Address Space Layout Randomization)
            - Randomizes memory address locations to disrupt attack patterns
        - DEP (Data Execution Prevention)
            - Prevents executable code from running in specific memory areas that should only store data
- Overflows
    - Types
        - Buffer Overflow
            - Definition
                - Occurs when data exceeds its allocated memory buffer, potentially altering or corrupting adjacent memory
            - Example
                - Overflowing a buffer with excess data, which can lead to “stack smashing” and allow code execution
        - Integer Overflow
            - Definition
                - Happens when a mathematical operation produces a result too large for its storage variable, causing “wrap-around”
            - Example
                - Adding two large numbers in a two-digit storage results in a negative number
    - Mitigation

        - Use boundary checks, choose appropriate variable sizes, and validate all inputs
- Race Conditions
    - Definition
        - Occur when the timing of operations impacts the outcome, leading to unpredictable results if exploited
    - Example
        - The Dirty COW vulnerability allowed attackers to escalate privileges on Linux systems by exploiting race conditions
    - Mitigation
        - Implement locking mechanisms to synchronize access to shared resources, ensuring events execute in the intended sequence
- Time of Check to Time of Use (TOCTOU)
    - Definition
        - A specific race condition where the state of data is altered between the time it's checked and the time it's used
    - Example
        - An attacker replaces a checked file with another before access is granted, gaining unauthorized access
    - Mitigation
        - Use locks and mutexes to control resource access, preventing state changes between checks and execution
- Summary
    - Deprecated Functions
        - Outdated functions lacking security features; replace with modern alternatives to avoid risks like buffer overflows

        - Unsafe Memory Utilization
            - Poor memory handling leads to leaks and vulnerabilities; tools like ASLR and DEP can reduce exploit potential
        - Overflows
            - Memory management issues (buffer and integer) allow data overflow, leading to possible data corruption or code execution
        - Race Conditions
            - Timing vulnerabilities that allow attackers to manipulate process outcomes; synchronization prevents unpredictable behavior
        - TOCTOU
            - Exploits delays between data checks and usage; locks ensure data integrity and prevent unauthorized changes

- **Configuration Vulnerabilities**
    - Configuration Vulnerabilities
        - Security weaknesses arising from improper system, application, or network configurations, potentially exposing systems to exploitation
    - Directory Service Misconfiguration
        - Definition
            - Improper setup or management of directory services (e.g., Active Directory), often resulting in unauthorized access or privilege escalation
        - Example
            - An organization grants broad permissions across user groups, allowing lower-privileged users unintended access to sensitive resources

        - Mitigation
            - Enforce least-privilege access, routinely audit directory configurations, and use tools like BloodHound to monitor directory structures
    - Insecure Configuration
        - Definition
            - Improperly set system parameters, often due to reliance on default settings or insufficient understanding of security configurations
        - Example
            - An open SSH port with weak credentials on a web server, which attackers can exploit for unauthorized access
        - Case Study
            - The Capital One breach in 2019 involved insecure AWS settings, allowing attackers to access 100 million customer records
        - Mitigation
            - Conduct regular configuration audits, harden system configurations, and avoid reliance on default settings
    - Embedded Secrets
        - Definition
            - Sensitive credentials (e.g., passwords, API keys) hard-coded directly into source code or configuration files
        - Example
            - In 2021, Twilio accidentally exposed API keys in a public GitHub repository, which attackers used to access internal systems
        - Mitigation

- Use secret management tools like HashiCorp Vault or AWS Secrets Manager, and employ automated scanning tools to detect embedded secrets in code

- Outdated and Unpatched Software and Libraries
  - Definition
    - Software or libraries that lack current patches, making them vulnerable to known exploits
  - Case Study
    - The 2021 Accellion data breach resulted from an unpatched version of Accellion's File Transfer Appliance, exposing sensitive client data
  - Mitigation
    - Implement automated patch management, regularly update software, and conduct comprehensive vulnerability scans
- End-of-Life (EOL) Software
  - Definition
    - Software that no longer receives vendor support or security updates, leaving it highly susceptible to attacks
  - Example
    - Windows 10 will reach end-of-life in October 2025, with users who remain on it facing increased risks from unpatched vulnerabilities
  - Mitigation
    - Plan for timely migration, isolate EOL systems from critical networks, and apply compensating controls until migration is complete

- Summary
  - Directory Service Misconfiguration
    - Poorly managed directory services can lead to unauthorized access; mitigate through least-privilege policies and auditing tools
  - Insecure Configuration
    - Default or improperly set configurations expose systems to risk; address with regular audits and secure setups
  - Embedded Secrets
    - Hard-coded credentials in code are high-risk; manage with secure storage solutions and automated scans
  - Outdated/Unpatched Software and Libraries
    - Vulnerable to known exploits; secure by maintaining regular updates and automated patching
  - End-of-Life Software
    - Unsupported systems remain exposed to new threats; plan migrations and apply interim controls to maintain security

- **Authorization Vulnerabilities**
  - Authorization Vulnerabilities
    - Flaws in access control management that allow unauthorized users or processes to perform actions beyond intended permissions
  - Confused Deputy
    - Definition
      - Occurs when an attacker tricks a higher-privilege program or service (the "deputy") into performing unauthorized actions on their behalf

        - Example
            - An attacker manipulates a proxy server to access restricted files by crafting a request that the proxy, with higher privileges, mistakenly honors
        - Mitigation
            - Implement robust access control checks at every authorization step, using tokens or certificates to validate user permissions directly and prevent reliance on intermediaries
    - Weak Ciphers
        - Definition
            - Vulnerabilities from using outdated or insecure cryptographic algorithms, leading to ineffective protection of sensitive data
        - Examples
            - Hashing Algorithms
                - MD5, SHA1, both prone to collision attacks
            - Encryption Ciphers
                - DES (short key vulnerable to brute-force attacks), RC4 (predictable patterns susceptible to attacks)
        - Case Study
            - A legacy banking app using DES encryption risks brute-force decryption of stored customer data, exposing passwords and account details
        - Mitigation
            - Regularly update encryption standards, use strong algorithms like AES256 and SHA256, implement secure transport layers (TLS), and

conduct periodic encryption audits to ensure secure cipher standards

- Vulnerable Third Parties
    - Definition
        - Authorization risks stemming from reliance on external services, APIs, or partners that could be compromised
    - Example
        - A web application uses a third-party payment processor; if compromised, attackers could intercept or alter transaction data, steal credit card details, or inject malicious code
    - Mitigation
        - Conduct thorough security evaluations of third-party services, limit access permissions, use secure APIs, and continually monitor compliance with security standards
- Summary
    - Confused Deputy
        - A privileged program performs unauthorized actions on behalf of an attacker, bypassing standard authorization; mitigated by direct permission checks and restricted intermediary permissions
    - Weak Ciphers
        - Outdated encryption methods (e.g., MD5, DES) leave data vulnerable to interception; addressed through updated encryption standards, strong algorithms, and regular protocol audits
    - Vulnerable Third Parties

            - Compromised third-party services can allow unauthorized access; reduced through rigorous third-party vetting, limited access, and continuous monitoring for security compliance

- **Malicious Code Attacks**
    - Malicious Code Attacks
        - Insertion of harmful software or code into a system to disrupt, steal, or manipulate data and operations
    - Implants
        - Definition
            - Persistent backdoors or unauthorized code/hardware stealthily embedded in a system, enabling attackers to gain continuous, often undetected access
        - Example
            - A software implant within a server injects code that sends sensitive data to an attacker every time a legitimate function runs
        - Mitigation
            - Code Reviews
                - Rigorous analysis of external or unverified code to identify unauthorized code
            - Endpoint Monitoring Tools
                - Use tools like OSSEC, CrowdStrike, and Carbon Black to detect unusual system behavior
            - Integrity Checks
                - Regular checks on critical files and configurations to spot unauthorized modifications

- Poisoning
    - Definition
        - The manipulation of a system's data or environment to alter its behavior, causing incorrect outcomes or degraded performance
    - Example
        - In a machine learning model, an attacker injects mislabeled spam messages as legitimate data, which reduces the model's accuracy over time
    - Mitigation
        - Data Access Control
            - Ensure only trusted sources can modify training data
        - Data Validation Checks
            - Detect abnormal data patterns to prevent mislabeling
        - Performance Monitoring
            - Regularly monitor the model's accuracy for sudden shifts, which may indicate poisoning
- Summary
    - Implants
        - Unauthorized code/hardware embedded in systems, providing attackers continuous access for data theft or manipulation; addressed through code reviews, monitoring tools, and integrity checks
    - Poisoning
        - Attackers tamper with data or environments to degrade model accuracy or mislead systems; mitigated by controlling data access, validating data, and monitoring performance

- **Hardware and Firmware Attacks**
    - Hardware and Firmware Attacks
        - Exploitation of vulnerabilities in physical components or embedded software to gain deep access, control, or disrupt system operations
    - Firmware Tampering
        - Definition
            - Altering the embedded firmware that controls hardware components, allowing attackers persistent, hard-to-detect control
        - Example
            - Attackers modify network card firmware to intercept data packets, maintaining ongoing access to sensitive network data
        - Mitigation
            - Firmware Updates
                - Use only verified sources for updates and limit physical access to sensitive devices
            - Integrity Checks
                - Implement cryptographic signatures on firmware updates to detect unauthorized changes
    - BIOS/UEFI Attacks
        - Definition
            - Targeting low-level firmware responsible for initializing and loading the operating system, enabling control before OS loads
        - Example
            - Injecting malicious code into the BIOS/UEFI firmware, allowing attackers to intercept data or disable security as soon as the device powers on

- Mitigation
  - Secure Boot
    - Ensures only signed, trusted firmware loads during startup
  - BIOS/UEFI Updates
    - Regular patches from official manufacturers to fix vulnerabilities
  - Tamper-Evident Hardware
    - Physical security measures to prevent unauthorized access to hardware

- USB-based Attacks
  - Definition
    - Using compromised USB devices to introduce malicious code directly into a system, bypassing traditional security
  - Example
    - Using tools like Rubber Ducky to install keyloggers or modify system settings upon connection
  - Mitigation
    - Policy Enforcement
      - Strict rules prohibiting unauthorized USB devices
    - Disable Autorun
      - Disabling automatic execution features on USB ports
    - USB Security Software
      - Scan and authenticate USB devices before access
    - Physical Protection
      - In high-security environments, use data-only cables and physically block unused ports

- Summary
  - Firmware Tampering
    - Attacking hardware components' embedded firmware, granting persistent, hard-to-detect control; mitigated through verified updates and integrity checks
  - BIOS/UEFI Attacks
    - Targeting boot firmware to gain control before OS loads; addressed with Secure Boot, regular updates, and tamper-evident hardware
  - USB-based Attacks
    - Using malicious USB devices to bypass security and inject code; mitigated with strict USB policies, disabling autorun, and physical port protection

- **Memory-based Attacks**
  - Memory-based Attacks
    - Exploitation of vulnerabilities in a system's memory management to execute malicious code, manipulate data, or cause system crashes
  - Memory Attacks
    - Definition
      - Targeting system memory to exploit vulnerabilities in memory allocation, such as buffer overflows or null pointer dereferences
    - Examples
      - Buffer Overflow
        - Attacker inputs data exceeding a buffer's capacity, overwriting memory to inject malicious code

            - Null Pointer Dereference
                - Program attempts to access a null memory address, causing crashes or denial of service
        - Mitigation
            - Secure Coding
                - Employ input validation, bounds checking, and memory-safe libraries
            - Security Mechanisms
                - Use Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) to prevent attackers from predicting memory addresses and executing unauthorized code
    - Shimming Attacks
        - Definition
            - Inserting a malicious "shim" layer between an application and the operating system to intercept and manipulate system calls
        - Examples
            - User Authentication Interception: Shim captures login credentials by intercepting system calls, redirecting them to an attacker.
            - API Hooking: Attacker inserts a shim to modify API functions, allowing unauthorized access or privilege escalation.
        - Mitigation
            - Code Integrity
                - Enforce policies allowing only trusted, signed code to load on the system
            - Application Allowlisting

                - Use secure boot and allow only verified applications to interact with system processes
            - Monitoring Tools
                - Deploy Endpoint Detection and Response (EDR) and Security Information and Event Management (SIEM) to detect unusual system calls or privilege escalations
    - Summary
        - Memory Attacks
            - Exploit vulnerabilities in memory management, such as buffer overflows and null pointer dereferences, allowing attackers to inject malicious code or crash systems
        - Shimming Attacks
            - Insert a malicious code layer between applications and the OS to intercept and manipulate system calls for unauthorized access, data interception, or privilege escalation

- **Electro-magnetic Attacks**
    - Electro-magnetic Attacks
        - Malicious use of electromagnetic interference (EMI) or electromagnetic pulses (EMP) to disrupt, damage, or manipulate electronic hardware and data
    - Electromagnetic Interference (EMI) Attacks
        - Definition
            - Malicious emissions of electromagnetic signals designed to interfere with the normal operation of electronic devices
        - Example

                - An attacker generates a strong electromagnetic field near a pacemaker, causing potential malfunctions
                - Similar disruptions could affect CPUs or memory in computers, leading to data corruption or crashes
            - Mitigation
                - Shielding
                    - Use copper or aluminum enclosures to block or absorb EMI
                - Shielded Rooms
                    - Create isolated zones for critical equipment to protect from external electromagnetic sources
                - Regular Testing
                    - Test for EMI resilience according to standards like FCC guidelines
        - Electromagnetic Pulse (EMP) Attacks
            - Definition
                - Use of a powerful burst of electromagnetic energy to disable or destroy electronic devices through voltage surges
            - Example
                - EMP devices disable network infrastructure, such as servers and routers, by creating voltage spikes that damage essential components, causing system failures in critical environments
            - Mitigation
                - Faraday Cages
                    - Enclosures made from conductive materials to block electromagnetic pulses

            - Surge Protectors
                - Devices to regulate voltage spikes and prevent circuit damage
            - EMP Resilience Standards
                - Follow standards like MIL-STD-188-125 for critical infrastructure systems
    - Summary
        - EMI Attacks
            - Involve the deliberate emission of electromagnetic signals that disrupt device operations, causing temporary faults, data corruption, or system crashes
        - EMP Attacks
            - Deliver a powerful electromagnetic burst that can disable or destroy electronic circuits, targeting the device's infrastructure and components with intense energy spikes

Detection and Mitigation

Objectives:

- 3.4 - Implement hardware security technologies and techniques
- 4.2 - Analyze vulnerabilities and attacks and recommend solutions to reduce the attack surface

- **Tamper Detection and Countermeasures**
    - Tamper Detection and Countermeasures
        - Mechanisms and actions designed to identify and respond to unauthorized attempts to alter or manipulate a system or device
    - Tamper Detection
        - Definition
            - Process of identifying unauthorized access or alterations in systems or devices using technologies like physical seals, sensors, and cryptographic checks
        - Examples
            - ATMs
                - Equipped with tamper-evident seals and sensors that trigger alerts or disable access if unauthorized entry is attempted
            - Digital Signatures/Hash Functions
                - Cryptographic tools that verify the integrity of data, raising alerts if tampering is detected

- Mitigation
  - Layered Detection
    - Use sensors in critical areas, tamper-evident materials, and real-time monitoring
  - Regular Audits
    - Conduct periodic maintenance and integrity checks
  - Cryptographic Validation
    - Digital signatures and hash functions for detecting unauthorized firmware or software changes

- Countermeasures
  - Definition
    - Tools or actions that prevent tampering or minimize its impact, such as physical barriers, encryption, or system shutdowns
  - Examples
    - Secure Data Facilities
      - Utilize locks, biometric access, and camouflage to control physical access
    - Encryption
      - Ensures data remains unreadable without a decryption key even if accessed by unauthorized entities
  - Mitigation
    - Physical & Digital Controls
      - Reinforced locks, biometric access, encryption, and access logging
    - Security Audits & Updates

            - Routine security checks and software updates to counter evolving threats
    - Summary
        - Tamper Detection
            - Uses tools like seals, sensors, and cryptographic checks to alert when unauthorized access or alterations are attempted
        - Countermeasures
            - Employ physical barriers, encryption, and system controls to prevent or limit tampering damage

- **Design Mitigations**
    - Design Mitigations
        - Security measures incorporated during the system design phase to prevent or reduce the impact of vulnerabilities in IT systems
    - Security Design Patterns
        - Definition
            - Established practices that guide secure system architecture, such as input validation, least privilege, secure defaults, error handling, and logging
        - Examples
            - Input Validation
                - Verifying user inputs to prevent harmful data, like SQL injections, from compromising systems
            - Least Privilege
                - Restricting user and application permissions to only what is necessary, limiting damage if a breach occurs

        - Secure Defaults
            - Configuring applications with safe settings by default to reduce misconfiguration risks
    - Mitigation
        - Apply security design patterns throughout development to reinforce resilience against vulnerabilities
        - Implement consistent security practices across the IT system for a strong security foundation
- Defense-in-Depth
    - Definition
        - A layered security approach with multiple protective measures around critical assets, providing redundancy
    - Examples
        - Database Security
            - Firewalls, encryption, access controls, and continuous monitoring protect sensitive information within databases
        - Perimeter and Internal Security Layers
            - Combining network segmentation, intrusion detection, and encryption to defend critical systems
    - Mitigation
        - Combine various security measures to safeguard against different threats
        - Regularly update and test layers to ensure ongoing effectiveness and system resilience
- Dependency Management
    - Definition

            - Monitoring and updating third-party libraries and components to prevent security flaws from outdated or vulnerable dependencies
        - Examples
            - Regular Audits
                - Checking third-party libraries for security patches and updates
            - Automated Vulnerability Scanning
                - Using tools to scan dependencies for known vulnerabilities
        - Mitigation
            - Proactively monitor and assess security risks in third-party tools and libraries
            - Use automated tools for vulnerability detection and integrate checks into the development pipeline to catch issues early
    - Summary
        - Security Design Patterns
            - Best practices, such as input validation and least privilege, that guide secure application architecture
        - Defense-in-Depth
            - A layered approach that protects critical assets by building multiple, redundant security measures
        - Dependency Management
            - Ensures third-party libraries and components are secure by selecting, monitoring, and updating them regularly

- **Validation Mitigations**
    - Validation Mitigations

        - Security measures that ensure data entering or leaving a system is validated and encoded, preventing unauthorized or harmful content from being processed or displayed
    - Input Validation
        - Definition
            - The process of verifying and sanitizing user-provided or external data before server processing to ensure it meets expected formats and data types
        - Purpose
            - Prevents attacks like SQL injection, cross-site scripting (XSS), and buffer overflows by filtering out dangerous or unexpected data
        - Analogy
            - Airport Security Checks
                - Just as passengers and their luggage are screened for restricted items, input validation checks data for disallowed characters, formats, or malicious patterns before processing
        - Example
            - A website form expecting a username only allows alphanumeric characters, blocking any attempt to include HTML or SQL code
        - Mitigation
            - Use both client-side and server-side validation to catch invalid data early
            - Employ allowlisting for acceptable inputs (safer than denylisting)
            - Utilize OWASP resources, regular expressions, and validation libraries to enforce data type constraints and length checks

- Output Encoding
  - Definition
    - A security measure that converts data into a safe format before it's displayed or processed, ensuring that it cannot be interpreted as executable code
  - Purpose
    - Protects against malicious scripts or content, especially from user-provided data, by rendering potentially harmful characters as plain text
  - Analogy
    - Website Comment Section
      - Output encoding ensures special characters in user comments are displayed as text, preventing scripts from executing
  - Example
    - A user enters a comment with the characters < and >, which are converted into HTML-safe representations to avoid script execution
  - Mitigation
    - Encode data as close to the point of display or use as possible
    - Use security libraries or frameworks with built-in encoding functions to simplify implementation
    - Combine with input validation for a multi-layered defense against malicious content
- Summary
  - Input Validation

            - Verifies and cleanses incoming data to ensure it fits expected patterns and is free of harmful content
        - Output Encoding
            - Safely formats data before it's displayed or processed, neutralizing potential threats by preventing executable code

- **Safe Functions**
    - Safe Functions
        - Programming functions that operate securely and reliably by managing data and resources in a way that prevents conflicts, memory issues, and inconsistencies
    - Atomic Functions
        - Definition
            - Functions that complete operations as single, indivisible actions, ensuring they either fully execute or not at all
        - Purpose
            - Ensures data consistency and stability, especially in concurrent environments where multiple threads may access the same data
        - Analogy
            - Bank Transfer System
                - In a banking transfer, an atomic function ensures that either the full amount is transferred, or nothing is, preventing partial or inconsistent states
        - Example
            - A money transfer system uses atomic functions to prevent one account from being debited without crediting the other

        - Mitigation
            - Use atomic operations or mutex locks to control access to shared resources
            - Implement atomic functions in programming with atomic variables or synchronization techniques, like mutexes, to ensure exclusive access during operation
    - Memory-safe Functions
        - Definition
            - Functions that securely manage memory to avoid issues like buffer overflows, memory leaks, and segmentation faults
        - Purpose
            - Protects applications from memory-related vulnerabilities by ensuring allocated memory is correctly used and safely released
        - Analogy
            - Efficient Data Handling
                - An application handling large data files uses memory-safe functions to keep data within memory boundaries, preventing crashes or unintended data access
        - Example
            - Memory-safe functions ensure that each data chunk fits within memory limits, avoiding buffer overflow and unintended access
        - Mitigation
            - Use languages emphasizing memory safety (e.g., Rust or Java) with automatic memory management

            - Apply libraries for safe memory allocation in C/C++, like smart pointers, which automate memory allocation and deallocation to prevent memory leaks
    - Thread-safe Functions
        - Definition
            - Functions that operate correctly when multiple threads access shared data simultaneously, avoiding race conditions and data corruption
        - Purpose
            - Ensures reliability in concurrent environments by preventing data conflicts during simultaneous access
        - Analogy
            - Online Shopping Cart
                - Thread-safe functions in a shopping cart prevent users' changes from conflicting, maintaining accurate cart totals even with simultaneous updates
        - Example
            - Functions handling cart updates use thread safety to prevent data conflicts, ensuring accurate totals for each user
        - Mitigation
            - Use locking mechanisms like mutexes for controlled resource access
            - Apply atomic variables or thread-safe data structures to manage concurrent modifications
    - Summary
        - Atomic Functions

            - Complete operations as indivisible steps, ensuring all-or-nothing execution for data consistency
        - Memory-safe Functions
            - Securely manage memory to prevent vulnerabilities like buffer overflows and memory leaks
        - Thread-safe Functions
            - Operate reliably in concurrent environments by preventing data conflicts and ensuring consistency

- **Access Control Mitigations**
    - Access Control Mitigations
        - Security measures that restrict access to system resources and functionalities, ensuring only authorized users or processes can interact with critical assets
    - Least Privilege
        - Definition
            - Ensures users or processes are granted only the minimum access needed to perform their tasks
        - Purpose
            - Minimizes unauthorized access or misuse by restricting permissions to essential functions, reducing potential damage if an account is compromised
        - Analogy
            - Payroll Permissions in Accounting Software

            - Only payroll staff can access sensitive payroll data; other employees, like customer service staff, are limited to data required for their roles
    - Example
        - An accounting software limits access to payroll data to payroll staff, protecting it from unauthorized access
    - Implementation
        - Regularly review access permissions and adjust them based on role changes
        - Use tools like Access Review Automation, Identity and Access Management (IAM), and Privileged Access Management (PAM) to enforce least privilege
- Least Function or Functionality
    - Definition
        - Limits system functions or features to only those required for essential operations, reducing potential vulnerabilities
    - Purpose
        - Minimizes the attack surface by limiting access to only necessary system components, preventing exploitation of unused features
    - Analogy
        - Web Application Access
            - Only admin users can access the admin panel, while regular users are restricted to the standard interface
    - Example

        - A web application restricts access to the admin panel, keeping regular users confined to the user interface, thus reducing the risk of unauthorized access
    - Implementation
        - Identify essential functions, disable or remove unnecessary features (e.g., unused APIs, debugging tools)
        - Regularly evaluate and update permissions to maintain a minimal functionality profile, protecting against potential exploitation
- Allow Listing
    - Definition
        - Limits access to a list of approved applications, IP addresses, or entities, blocking anything not explicitly permitted
    - Purpose
        - Ensures only trusted resources can access the system, effectively preventing unauthorized applications or users from interacting with sensitive assets
    - Analogy
        - Secure Network Access
            - Only trusted IP addresses are permitted to connect to a corporate network, blocking unauthorized attempts by default
    - Example
        - In a secure corporate network, only trusted IPs are allowed to connect; attempts from unlisted IPs are automatically blocked
    - Implementation

            - Maintain an updated list of trusted applications, IP addresses, and devices
            - Use tools like Microsoft Active Directory, AppLocker, and Defender Application Control to streamline allow listing management, adjusting the list as needed
    - Summary
        - Least Privilege
            - Grants minimum access required for tasks, reducing unauthorized data exposure
        - Least Function or Functionality
            - Limits available system features to necessary ones, minimizing entry points for attackers
        - Allow Listing
            - Restricts access to a predefined list of approved entities, ensuring only trusted resources can interact with critical assets

- **Confidentiality Management**
    - Confidentiality Management
        - Involves protecting sensitive information from unauthorized access or disclosure by using structured methods and tools
    - Indexing
        - Definition
            - Organizes and structures sensitive data to allow efficient retrieval without exposing underlying content
        - Implementation

        - Creates an index structure with metadata or unique identifiers pointing to data locations
        - Uses database tools like B-trees or hash indexes
    - Example
        - Index includes non-sensitive attributes like timestamps or keywords for search queries
        - Query searches the index first, and only authorized users access the full dataset
    - Enhancements
        - Combine indexing with encryption to ensure data remains secure even when indexed
- Key Rotation
    - Definition
        - Regularly updates encryption keys or sensitive credentials to minimize risks from key compromise or reuse
    - Implementation
        - Use secrets management tools like AWS Key Management Service to automate rotation
        - Enforce strict policies for key handling and secure storage
    - Importance
        - Ensures outdated keys cannot be exploited
        - Reduces risks from prolonged key exposure or re-use
- Encryption
    - Definition
        - Converts sensitive data into unreadable ciphertext, requiring a decryption key for access

        - Use Cases
            - Data at Rest
                - Encrypt stored data to prevent unauthorized access
            - Data in Transit
                - Protect data during transfer between systems or networks
        - Standards
            - Use strong encryption protocols like AES-256
            - Manage decryption keys securely via key management systems
        - Best Practices
            - Apply encryption consistently across storage and transmission stages
            - Regularly update encryption algorithms to remain secure against evolving threats
    - Code Signing
        - Definition
            - Ensures the authenticity and integrity of software through digital signatures, verifying that it has not been tampered with
        - Implementation
            - Use trusted code-signing certificates
            - Store private signing keys securely (e.g., in hardware security modules or secure vaults)
        - Example
            - A software update is digitally signed to verify it originated from the developer and remains unaltered
        - Best Practices
            - Conduct regular audits of code-signing activities

                - Monitor for unauthorized signing attempts to detect tampering early
        - Summary
            - Confidentiality management safeguards sensitive information through structured approaches
                - Indexing
                    - Enables efficient data retrieval without exposing sensitive content
                - Key Rotation
                    - Regularly updates encryption credentials to mitigate risks of compromise
                - Encryption
                    - Protects data by converting it into a secure format for storage and transmission
                - Code Signing
                    - Verifies the authenticity and integrity of software, ensuring it has not been altered
            - These practices collectively form a robust strategy for protecting sensitive data and maintaining organizational security

- **Update Management**
    - Update Management
        - The process of regularly updating and patching all components within a system to maintain security, stability, and compatibility
    - Firmware
        - Definition

        - Embedded software within hardware devices that controls and facilitates hardware operations
    - Purpose
        - Ensures secure communication between hardware and the operating system, addressing vulnerabilities that may expose the system to unauthorized access
    - Example
        - Dell Command Update tool automates firmware updates for Dell devices, streamlining firmware maintenance
    - Risks of Poor Management
        - Vulnerabilities remain unpatched, leading to security risks and potential hardware failures
- System Images
    - Definition
        - Comprehensive backups that capture the entire state of a system (OS, applications, settings, files) at a specific time
    - Purpose
        - Provides a consistent baseline configuration for efficient, secure deployment
    - Example
        - Tools like Microsoft SCCM or VMware vSphere manage and update system images efficiently
    - Risks of Poor Management
        - Outdated images increase security risks by deploying vulnerable software across multiple devices
- Hypervisors

    - Definition
        - Software that enables virtualization by hosting and managing virtual machines (VMs)
    - Purpose
        - Ensures security, stability, and compatibility across virtual environments
    - Example
        - VMware vCenter and Microsoft Hyper-V Manager centralize hypervisor updates with minimal impact on running VMs
    - Risks of Poor Management
        - Unpatched hypervisors can lead to VM escapes, allowing attackers to access other VMs or the host system
- Operating Systems
    - Definition
        - The core software that coordinates hardware and applications
    - Purpose
        - Regular updates improve security, stability, and compatibility with applications and hardware
    - Example
        - Windows Update, yum (Linux), and macOS Software Update provide automated OS updates
        - Microsoft SCCM enables centralized OS patch management for large environments
    - Risks of Poor Management
        - Outdated OSs are vulnerable to malware and exploits, causing potential data loss and compatibility issues

- Software
  - Definition
    - Applications and utilities running on the OS, each requiring separate updates
  - Purpose
    - Keeps applications functional, secure, and compatible with other system components
  - Example
    - Regularly updating web browsers, productivity tools, and other applications helps maintain system integrity
  - Risks of Poor Management
    - Vulnerable applications can be exploited, leading to unauthorized access and potential data breaches
- Summary
  - Firmware
    - Foundational code in hardware; regular updates prevent security risks
  - System Images
    - Backups for deployment; keeping them updated ensures consistency and security across devices
  - Hypervisors
    - Manage virtual machines; updates prevent VM escapes and strengthen virtual environment security
  - Operating Systems
    - Core software of the computing environment; frequent updates protect against vulnerabilities and ensure stability

        - Software
            - Individual applications; separate updates prevent exploits and maintain compatibility

- **Fail-Safe Mechanisms**
    - Fail-safe Mechanisms
        - Systems designed to maintain a secure or safe operational state in the event of a failure or security breach, by defaulting to either locked or accessible modes based on priorities of security or safety
    - Fail Secure
        - Definition
            - Mechanisms that lock down access or shut down functions during a failure to maintain security, preventing unauthorized access at all costs
        - Purpose
            - To prioritize security and data protection, even if it causes temporary service disruption
        - Examples
            - Server Protection
                - A server with sensitive data shuts down access during an anomaly to prevent data breaches
            - Bank Database
                - A bank's database detects unusual activity and locks access, protecting financial data while preventing temporary user access
        - Risks

            - Temporary denial of access to legitimate users, impacting availability but ensuring data protection
        - Analogy
            - Like a vault door that locks during a power outage, protecting valuables even if access is temporarily denied
        - Use Cases for IT Systems
            - Ideal for critical data systems requiring high confidentiality (e.g., financial and healthcare data systems)
    - Fail Safe
        - Definition
            - Mechanisms that maintain safe, continuous operation during failures, prioritizing functionality over strict security
        - Purpose
            - To keep systems operational, emphasizing safety and accessibility, even if it means relaxing some security measures
        - Examples
            - Emergency Door Mechanism
                - During a power outage, an emergency exit door unlocks to ensure people can exit safely
        - Network Failover
            - If a primary internet connection fails, traffic reroutes through a secondary link with lower security to maintain business continuity
        - Risks
            - Some reduction in security as functionality is prioritized to ensure safe and continuous operations
        - Analogy

        - Like an emergency exit door that unlocks in a power outage, allowing safe egress even if it slightly reduces security
    - Use Cases for IT Systems
        - Suitable for business continuity situations where safe operation must continue (e.g., alternate network routing)
- Summary
    - Fail Secure
        - Emphasizes security, locking down access to protect sensitive information during failures
    - Fail Safe
        - Emphasizes safety and continuous functionality, allowing systems to operate during failures with relaxed security

Threat Modeling Considerations

Objectives:

- 1.4 - Perform threat modeling activities
- 3.2 - Analyze requirements to enhance the security of endpoints and servers

- **Threat Actor Motivation**
  - Threat Actor Motivation
    - The underlying reasons or goals that drive attackers to target specific organizations or systems, including political, financial, social, or personal motives
  - Geopolitical
    - Definition
      - Driven by nation-states to disrupt or influence the political or economic stability of other countries
    - Objectives
      - Target critical infrastructure, financial systems, and government networks to create instability. Gain a strategic advantage over competing nations
    - Examples
      - APT Groups
        - Fancy Bear (linked to Russia) has reportedly attempted to influence elections by targeting and leaking sensitive information
    - Response Strategy

            - Use threat intelligence and enhanced monitoring to detect long-term, covert threats
            - Focus on critical infrastructure protection to guard against national-scale attacks
    - Espionage
        - Definition
            - Motivated by the desire to steal sensitive information, such as intellectual property, trade secrets, or state intelligence
        - Objectives
            - Gain a competitive advantage by accessing proprietary information
            - Remain undetected within systems to gather intelligence
        - Examples
            - SolarWinds Breach
                - Allegedly by APT 29 (Cozy Bear) using backdoor software (SUNBURST) for long-term, stealthy access to government and corporate data
        - Response Strategy
            - Strengthen data access controls and monitor for unusual patterns to detect unauthorized access
            - Implement regular security checks on software and updates to prevent breaches
    - Financial
        - Definition
            - Driven by monetary gain, often achieved through ransomware, fraud, or data theft

        - Objectives
            - Obtain quick payoffs by extorting money or selling stolen data
        - Examples
            - DarkSide Ransomware
                - Known for targeting businesses like Colonial Pipeline, disrupting operations and demanding ransom for decryption
        - Response Strategy
            - Maintain data backups and endpoint protection to reduce exposure
            - Train employees on phishing awareness and monitor for suspicious financial activities
    - Activism
        - Definition
            - Also known as hacktivism, focused on promoting a political or social cause
        - Objectives
            - Disrupt services, leak information, or cause reputational damage to organizations perceived as opposing their cause
        - Examples
            - Anonymous (Operation Tunisia)
                - Supported Arab Spring protests by attacking government websites and helping citizens bypass censorship
        - Response Strategy
            - Monitor public sentiment and anticipate attacks based on political or social events

        - Prepare for potential disruptions and secure data against leaks
- Notoriety
    - Definition
        - Driven by the desire for attention, status, or recognition within hacker communities
    - Objectives
        - Cause disruptions for bragging rights, often using unsophisticated tools
    - Examples
        - Lapsus$ Group: Known for targeting high-profile companies like Microsoft and Nvidia, often using basic tactics for visibility
    - Response Strategy
        - Secure public-facing systems and monitor for traffic spikes to detect and mitigate disruptive attacks
        - Focus on system hardening and public monitoring to guard against visibility-driven disruptions
- Summary
    - Geopolitical
        - Nation-states aiming to destabilize or influence other countries for strategic gain
    - Espionage
        - Stealing sensitive data for competitive or intelligence advantages
    - Financial
        - Profit-driven attacks like ransomware or fraud
    - Activism
        - Hacktivism targeting organizations for political or social causes

        - Notoriety
            - Thrill-seekers looking to gain recognition through disruptive actions

- **Threat Actor Resources**
    - Threat Actor Resources
        - Assets that enable attackers to plan, execute, and sustain operations, including time and money
        - These resources contribute to the sophistication, persistence, and scale of an attack
    - Time
        - Definition
            - Time allows attackers, especially well-funded ones like nation-state actors, to conduct prolonged, methodical campaigns
        - Importance
            - Enables careful planning and execution over extended periods, sometimes lasting months or years
            - Allows attackers to develop custom strategies and remain undetected
        - Example
            - SolarWinds Breach
                - Attackers allegedly spent over a year developing and deploying a backdoor, allowing covert access to government and corporate networks worldwide
        - Response Strategy

        - Use behavioral analysis and continuous monitoring to detect unusual patterns over time
        - Implement long-term detection strategies like anomaly detection, rather than relying solely on single alerts
- Money
    - Definition
        - Financial resources provide attackers access to advanced tools, skilled personnel, and high-level exploits
    - Importance
        - Enables the purchase of zero-day exploits, premium tools, and infrastructure like command and control (C2) servers
        - Allows hiring of skilled attackers and even recruiting insiders
    - Example
        - Lazarus Group (North Korea)
            - Used significant funding for the 2016 Bangladesh Bank heist, resulting in $81 million stolen through a sophisticated scheme
    - Response Strategy
        - Focus on securing high-value assets and multi-layered defenses
        - Invest in robust incident response capabilities and specialized malware detection to counter sophisticated attacks
- Summary
    - Time
        - Provides the flexibility to conduct long-term, stealthy campaigns, especially for nation-state actors who prioritize prolonged data collection and vulnerability research

            - Money
                - Enables access to advanced resources, skilled personnel, and high-end technology, enhancing the impact and sophistication of an attack

- **Threat Actor Capabilities**
    - Threat Actor Capabilities
        - Skills, tools, and expertise that attackers use to identify and exploit vulnerabilities in target systems
        - Key capabilities include knowledge, vulnerability creation, exploit creation, and supply chain access
    - Knowledge
        - Definition: Understanding of a target's systems, defenses, and vulnerabilities.
        - Acquisition
            - Through reconnaissance or insider information
            - Techniques like Open-Source Intelligence (OSINT) and network scanning
            - Passive Reconnaissance
                - Gathering public information without interacting directly (e.g., using OSINT)
            - Active Reconnaissance
                - Directly scanning target's interfaces to identify vulnerabilities
        - Example

            - Using tools like Nmap for network scanning or social media for employee information
        - Sophistication
            - Basic tools for quick scans vs. custom scripts for stealthier operations
    - Vulnerability Creation
        - Definition
            - Introducing new weaknesses within a system, such as through code tampering or system misconfigurations
        - Execution
            - Requires technical skill in programming and configuration
            - Can involve tampering with open-source projects, misconfigurations, or social engineering
        - Example
            - Embedding malicious code within an open-source library widely used by the target organization
        - Approach
            - Often long-term, aiming for vulnerabilities that go unnoticed but exploitable in the future
    - Exploit Creation
        - Definition
            - Development of tools or techniques to take advantage of known vulnerabilities
        - Execution
            - Includes basic scripts or sophisticated payloads for zero-day exploits

        - Requires programming knowledge and understanding of the software being targeted
    - Example
        - Crafting a buffer overflow exploit to target software used by the organization
    - Sophistication
        - Simple scripts for known vulnerabilities vs. months of research for creating unique, zero-day exploits
- Supply Chain Access
    - Definition
        - Gaining indirect access to a target by compromising third-party vendors or service providers
    - Execution
        - Involves extensive planning and targeting of trusted vendors. Commonly through spear-phishing, social engineering, or direct exploitation of vendors
    - Example
        - SolarWinds breach
            - Attackers compromised the Orion software platform to distribute malware to numerous clients
    - Sophistication
        - Targets third-party software updates or services to bypass security, potentially affecting multiple organizations
- Summary
    - Knowledge

                - Threat actors gather intelligence through reconnaissance to map a target's vulnerabilities
            - Vulnerability Creation
                - Skilled attackers introduce weaknesses into systems via tampering, social engineering, or configuration changes
            - Exploit Creation
                - Attackers create tools to exploit vulnerabilities, with advanced actors developing zero-day exploits
            - Supply Chain Access
                - Advanced attackers compromise vendors to indirectly access a target, leveraging trusted relationships

- **Attack Patterns**
    - Attack Patterns
        - Tactics and techniques that threat actors use to exploit vulnerabilities in a system, commonly modeled in threat analysis
        - Key patterns include Injection Attacks, Authentication and Authorization Attacks, and On-Path Attacks
    - Injection Attacks
        - Definition
            - Inserting malicious code or commands into a system through vulnerable input points, manipulating the application's behavior
        - Types
            - SQL Injection
                - Injecting SQL statements to gain unauthorized database access

- Command Injection
  - Executing system commands to control system functions
- Code Injection
  - Inserting code directly into the application's codebase
- Cross-Site Scripting (XSS)
  - Injecting user-side scripts to manipulate content or steal session data

- Example
  - In a login form, entering " ' OR 1=1;--" to trick the system into granting access by bypassing authentication
- Mitigation
  - Input Validation and Sanitization
    - Ensuring all inputs conform to expected formats
  - Parameterized Queries (Prepared Statements)
    - Separates code from data, especially effective against SQL injection
  - Web Application Firewalls (WAFs)
    - Detect and block suspicious requests

- Authentication and Authorization Attacks
  - Definition
    - Exploiting weaknesses in access control mechanisms to bypass login systems or gain unauthorized access
  - Types
    - Password Reset Exploits
      - Manipulating the password reset process to take over accounts

        - Session Hijacking
            - Taking control of active user sessions
        - Exploiting Weak Access Control
            - Leveraging flaws in access control lists (ACLs) or role-based access control (RBAC) for privilege escalation
    - Example
        - An attacker bypasses a password reset process that relies on easily accessible information, like a user's email or knowledge-based questions
    - Mitigation
        - Multi-Factor Authentication (MFA)
            - For both login and password reset processes
        - Secure Session Management
            - Enforcing secure session ID generation and management
        - RBAC and ACL Configuration
            - Setting appropriate permissions to prevent unauthorized privilege escalation
- On-Path Attacks (Formerly Man-in-the-Middle Attacks)
    - Definition
        - Attacker intercepts and potentially alters communications between two parties, leading to data theft or content injection
    - Types
        - Packet Sniffing
            - Capturing network traffic
        - Session Hijacking

        - Taking control of an active session between a user and application
    - Example
        - On public Wi-Fi, an attacker intercepts unencrypted bank login credentials and uses them to access the victim's account
    - Mitigation
        - Encryption Protocols (TLS)
            - Secures communications to prevent interception
        - Certificate Authentication
            - Verifies the authenticity of communicating parties
        - Anomaly Detection Systems
            - Detects unusual traffic patterns to identify on-path attacks
- Summary
    - Injection Attacks
        - Exploit vulnerabilities in input fields, allowing attackers to insert code that manipulates system behavior
        - Mitigated by input validation, parameterized queries, and WAFs
    - Authentication and Authorization Attacks
        - Target weak access controls, enabling attackers to bypass login systems or gain unauthorized privileges
        - Mitigated with MFA, secure session management, and proper role-based permissions
    - On-Path Attacks
        - Involve intercepting communications to steal or modify data between parties, commonly occurring over unprotected networks

            - Mitigated by TLS encryption, certificates, and anomaly detection systems

- **Threat Actor Methods**
    - Threat Actor Methods
        - Specific strategies and approaches attackers use to exploit vulnerabilities in a system to achieve their objectives
        - Key methods include Abuse Cases, Antipatterns, and Attack Trees or Graphs
    - Abuse Cases
        - Definition
            - Scenarios where attackers manipulate legitimate features in a system to gain unauthorized access or cause harm
        - Examples
            - Fake Reviews
                - Competitors or attackers post fake positive or negative reviews to manipulate a product’s rating, impacting credibility and customer perception
            - Bot Upvotes/Downvotes
                - Using bots to artificially skew rankings or influence user-generated content
        - Mitigation
            - Review Verification
                - Requiring verified purchases and implementing CAPTCHA to reduce bot activity
            - Pattern Monitoring

            - Detecting unusual activity, such as rapid or excessive submissions from the same IP
        - Usage Limits and Audits
            - Regular audits and limits on user activities to identify and manage potential abuse
- Antipatterns
    - Definition
        - Poor design choices or practices that appear functional but unintentionally create security vulnerabilities
    - Examples
        - Base64 Encoding of Passwords
            - Using base64 to encode sensitive information like passwords may seem secure but is easily reversible
        - Hardcoding Sensitive Data
            - Storing passwords, API keys, or other sensitive information directly in the code exposes it to anyone who accesses the code
    - Mitigation
        - Environment Variables
            - Use environment variables for sensitive data to keep it separate from code
        - Secure Development Frameworks
            - Tools like Django (Python) and Spring Security (Java) help prevent vulnerabilities through built-in protections
        - Regular Code Reviews and Encryption

            - Secure data handling practices and periodic code reviews reinforce security against common vulnerabilities
- Attack Trees or Graphs
    - Definition
        - Visual tools that outline potential pathways attackers might use to achieve their goals, breaking down attacks into manageable phases and steps
    - Example
        - Sensitive Data Compromise
            - An attack tree could start with entry points (e.g., phishing, weak passwords, unpatched software) and detailed steps like privilege escalation or lateral movement within the network
            - Each node represents an attack phase, enabling teams to identify vulnerable stages and focus defenses accordingly
    - Mitigation
        - Focused Security Resources
            - Allocating resources to strengthen defense at specific attack steps based on attack tree analysis
        - Regular Vulnerability Assessments
            - Identifying and patching vulnerabilities to minimize attack entry points
        - Least-Privilege Access Policies
            - Restricting user permissions to reduce the impact of unauthorized access
- Summary

        - Abuse Cases
            - Exploiting legitimate system features (e.g., fake reviews, bot manipulation)
            - Mitigated by review verification, monitoring, and audits
        - Antipatterns
            - Poor practices that create vulnerabilities (e.g., base64 encoding, hardcoding credentials)
            - Mitigated by secure development frameworks, environment variables, and code reviews
        - Attack Trees or Graphs
            - Visual mapping of attack paths to understand and defend against potential attack phases
            - Mitigated by focused resource allocation, vulnerability assessments, and least-privilege policies

- **Initial Access and Escalation Methods**
    - Initial Access and Escalation Methods
        - Techniques used by attackers to gain unauthorized entry into systems and increase privileges to control broader parts of the network
    - Injections
        - Definition
            - Techniques where malicious code is inserted into a system to gain access or execute unauthorized actions
        - Types
            - SQL Injection
                - Attacker inserts harmful SQL code into user input fields

        - Exploits backend databases to retrieve, modify, or delete unauthorized data
        - Example Tool
            - SQLmap
    - Command Injection
        - Attacker injects harmful commands into a vulnerable application
        - Executes arbitrary code on the server
- Prevention
    - Use parameterized queries to secure input fields
    - Apply input validation to block executable code
    - Deploy Web Application Firewalls (WAFs) to detect and block injection attempts
    - Apply the principle of least privilege for database systems

- Credential Dumping
    - Definition
        - Extraction of stored usernames and passwords from compromised systems
    - Tool Example
        - Mimikatz
            - Extracts plaintext passwords, hashes, and Kerberos tickets from memory, enabling lateral movement and access to high-value accounts
    - Prevention
        - Disable outdated protocols like LM and NTLM; use stronger encryption methods

            - Implement multi-factor authentication (MFA) to reduce the risk of credential misuse
            - Limit access to memory by disabling unnecessary administrator rights
            - Enforce secure, updated password policies
    - Privilege Escalation
        - Definition
            - Exploiting vulnerabilities to gain higher privileges within a system
        - Types
            - Vertical Escalation
                - Gain administrator or root-level access
            - Horizontal Escalation
                - Access accounts or resources at a similar privilege level to expand control
        - Tool Example
            - Metasploit
                - Searches for and exploits privilege escalation flaws, such as outdated software or misconfigured permissions
        - Prevention
            - Regularly update systems and applications with the latest patches
            - Apply the principle of least privilege, restricting access to only what is necessary
            - Monitor systems for unusual behavior, such as attempts to access sensitive files or execute commands with elevated privileges
    - Summary

        - Attackers use initial access and escalation methods to compromise systems and gain control
            - Injections
                - Insert harmful code (e.g., SQL or command injection) to gain unauthorized access or manipulate systems
            - Credential Dumping
                - Steal stored usernames and passwords to impersonate legitimate users and bypass security
            - Privilege Escalation
                - Exploit vulnerabilities to gain elevated privileges, increasing the attacker's control over the network
        - Mitigation Strategies
            - Secure inputs with parameterized queries and validation
            - Implement multi-factor authentication and strong encryption methods
            - Regularly update and patch systems to address vulnerabilities
            - Apply the principle of least privilege to restrict access rights
            - Monitor for unusual system behavior to detect potential attacks early

- **Post-exploitation and Evasion Methods**
    - Post-Exploitation and Evasion Methods
        - Techniques attackers use after initial system access to maintain control, expand their reach, and avoid detection within a network
        - Key techniques include Lateral Movement, Unauthorized Execution, and Defensive Evasion

- Lateral Movement
  - Definition
    - Attackers move within the same network segment to access additional resources without detection
  - Example
    - WannaCry Ransomware: Spread laterally within network segments by exploiting the Windows SMB protocol vulnerability (EternalBlue) to access other vulnerable systems
  - Mitigation
    - Network Segmentation
      - Limits access by isolating critical assets
    - Least-Privilege Access
      - Ensures minimal permissions for users to reduce the impact of compromised accounts
    - Internal Traffic Monitoring
      - Detects unusual account and device activity that may signal lateral movement
    - Restricting Remote Tools
      - Disable tools like PsExec, and use MFA to reduce access risks
- Unauthorized Execution
  - Definition
    - Attackers run unauthorized commands or scripts on compromised systems to achieve their objectives, such as data exfiltration or system disruption
  - Example

        - Emotet Malware
            - Executed unauthorized scripts to deploy ransomware or data-stealing malware across systems
    - Mitigation
        - Regular Security Patches
            - Keeps systems updated to close known vulnerabilities
        - Endpoint Detection and Response (EDR)
            - Identifies unusual execution patterns to stop attacks early
        - Permission Restriction and Monitoring
            - Reduces user privileges and tracks unauthorized command executions to limit attack reach
        - Robust Malware Detection
            - Detects and mitigates unauthorized execution risks
- Defensive Evasion
    - Definition
        - Techniques to avoid detection, such as disabling security tools, hiding processes, clearing logs, and using encryption to bypass monitoring systems
    - Example
        - Malware Obfuscation
            - Attackers obfuscate or encrypt malicious code to bypass signature-based detection, making it difficult for antivirus software to recognize threats
    - Mitigation
        - Behavioral-Based Detection

            - Identifies unusual patterns beyond traditional signature detection
        - Advanced Logging and Tamper Protection
            - Maintains reliable records and prevents attackers from modifying security software or logs
        - Regular Security Audits
            - Ensures configurations align with security policies and detects tampering
- Summary
    - Lateral Movement
        - Attackers explore systems within the same network segment to access more resources
        - Mitigated by network segmentation, access controls, monitoring, and disabling remote tools
    - Unauthorized Execution
        - Attackers execute malicious code or commands to disrupt systems or steal data
        - Prevented with security patching, EDR, permission restrictions, and malware detection
    - Defensive Evasion
        - Techniques to avoid detection, like obfuscation, disabling tools, and log tampering
        - Countered by behavior-based detection, logging, tamper protection, and security audits

Threat Modeling Frameworks

Objective 1.4: Perform threat modeling activities

- **Cyber Kill Chain**
  - Cyber Kill Chain
    - A threat modeling framework developed by Lockheed Martin that outlines the sequential steps attackers typically follow during a cyber attack
    - Breaking any link in the chain disrupts the attack, forcing attackers to restart from the beginning
    - The framework includes seven steps
      - Reconnaissance
      - Weaponization
      - Delivery
      - Exploitation
      - Installation
      - Command and Control (C2)
      - Actions on Objectives
  - Reconnaissance
    - Definition
      - Attackers gather information on their target, typically using Open Source Intelligence (OSINT) or active scanning, to identify potential weaknesses
    - Techniques

            - Passive reconnaissance (OSINT) Active reconnaissance (network scanning)
        - Example
            - Attackers scan for open ports or exposed endpoints to find entry points
        - Mitigation
            - Restrict external information sharing, monitor for suspicious scanning activities, and use network intrusion detection systems
    - Weaponization
        - Definition
            - Attackers create a malicious payload or an exploit customized for the target's vulnerabilities discovered during reconnaissance
        - Example
            - Combining malware with exploit code to maximize effectiveness and test in a controlled environment
        - Mitigation
            - Use sandboxing for testing files and inspect email attachments to detect suspicious content
        - Delivery
            - Definition
                - Attackers deliver the crafted payload to the target through phishing emails, compromised websites, or infected USB devices
            - Example
                - Sending phishing emails with a malicious attachment
            - Mitigation

            - Set up email filtering, implement security awareness training, and use anti-phishing solutions
- Exploitation
    - Definition
        - The payload executes on the target system, exploiting a vulnerability to gain access
    - Example
        - A victim clicks a malicious link or opens a harmful attachment, triggering code execution
    - Mitigation
        - Apply regular security patching and endpoint protection to block known vulnerabilities
- Installation
    - Definition
        - Malware installs itself on the compromised system, often creating a backdoor for continued access
    - Example
        - Attackers establish persistence mechanisms to ensure ongoing access
    - Mitigation
        - Use endpoint detection and response (EDR) solutions to detect unauthorized software installations
- Command and Control (C2)
    - Definition
        - Attackers establish a communication channel to control the compromised system remotely

        - Example
            - Attackers use encrypted or concealed communication to avoid detection
        - Mitigation
            - Monitor network traffic for suspicious outbound connections and use firewall rules to block known malicious IP addresses
    - Actions on Objectives
        - Definition
            - Attackers achieve their ultimate goal, such as data exfiltration, ransomware deployment, or infrastructure sabotage
        - Example
            - Exfiltrating sensitive data or deploying ransomware
        - Mitigation
            - Use data loss prevention (DLP) tools and implement strict access controls to minimize data theft risk
    - Example of Early Detection in the Cyber Kill Chain
        - Scenario
            - Employees report suspicious emails with unknown attachments
        - Response
            - Block the sender, implement additional email filters, and inform employees
            - This prevents exploitation, installation, and further steps in the chain
    - Cyber Kill Chain and the "Six Ds" of Defense
        - Detect
            - Identify suspicious activities early

        - Deny
            - Block malicious communications or access
        - Disrupt
            - Interrupt the attacker’s progress, e.g., filtering emails
        - Degrade
            - Slow down attacks to buy response time
        - Deceive
            - Use honeypots or decoy systems
        - Destroy
            - Neutralize malware or compromised systems
    - Summary
        - The Cyber Kill Chain framework breaks down an attack into seven sequential phases
        - Understanding each phase helps defenders disrupt attacks before they reach their objectives by recognizing and blocking patterns at any point in the chain
        - The framework enables structured defense, proactive detection, and effective response strategies, helping security teams safeguard systems against multi-step cyber threats

- **Common Attack Pattern Enumeration and Classification (CAPEC)**
    - CAPEC (Common Attack Pattern Enumeration and Classification)
        - A threat modeling framework developed by MITRE that categorizes and describes common attack patterns used by attackers to exploit system vulnerabilities

        - CAPEC provides detailed attributes for each attack pattern, including prerequisites, outcomes, and typical steps involved
        - Security professionals and developers use CAPEC to anticipate, understand, and mitigate potential threats during the Software Development Lifecycle (SDLC)
    - Key Aspects of CAPEC
        - Attack Pattern Attributes
            - Prerequisites
                - Conditions or configurations necessary for an attack to succeed (e.g., certain software versions or permissions)
            - Outcomes
                - Expected results if the attack succeeds, such as data theft, denial of service, or system disruption
            - Attack Steps
                - Typical procedures an attacker might follow to carry out the attack, offering insight into the attack's methodology
        - Structure and Organization
            - CAPEC entries are categorized to allow for easy lookup of related attacks or tactics, helping defenders focus on specific vulnerabilities
            - Available as a public resource at https://capec.mitre.org/, making it accessible for security teams worldwide
    - CAPEC in the Software Development Lifecycle (SDLC)
        - Use Case
            - Early Stage Vulnerability Identification

            - CAPEC helps developers identify vulnerabilities in design or early development, allowing for proactive mitigation
        - Guidance for Security Best Practices
            - CAPEC patterns provide defensive recommendations that can inform secure coding practices, such as input validation and strong encryption
    - SQL Injection example
        - Scenario
            - A development team identifies a SQL injection vulnerability
        - CAPEC Entry
            - The team refers to the CAPEC entry for SQL injection, which provides
                - Attack Methodology
                    - How attackers submit malicious SQL queries to manipulate databases
                - Defense Recommendations
                    - Use of parameterized queries and input validation to secure database interactions
        - Outcome
            - Implementation of these practices mitigates SQL injection risks, strengthening application security
- CAPEC for Ongoing Security
    - Benefits for Security Teams
        - Detailed Threat Information

        - CAPEC entries give insight into attack conditions, impact, and attacker techniques
      - Organized Attack Patterns
        - Security teams can quickly locate information by category or subcategory, aiding efficient threat management
      - Industry-Specific Threats
        - CAPEC helps identify threats related to specific industries, enabling targeted defense strategies
    - Ongoing Threat Management example
      - Scenario
        - A security team monitors for common injection attacks
      - CAPEC Reference
        - The team consults injection attack patterns within CAPEC for information on typical techniques and defenses
      - Outcome
        - With CAPEC guidance, the team enforces input validation on multiple applications, helping to secure against various injection risks
- Summary
    - CAPEC, developed by MITRE, categorizes attack patterns and provides in-depth information on each, including attack prerequisites, possible outcomes, and steps taken
    - Available as a free resource, CAPEC serves as a comprehensive tool for identifying, understanding, and mitigating threats in software development and ongoing security

        - By referencing CAPEC, security professionals and developers can build stronger, more resilient defenses into applications, effectively reducing vulnerabilities throughout the software lifecycle

- **MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK)**
    - MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK)
        - The MITRE ATT&CK framework categorizes the tactics, techniques, and procedures (TTPs) used by adversaries to compromise and exploit systems
        - It is organized into matrices that detail the stages of an attack, including specific techniques, permissions required, affected platforms, and mitigation strategies
    - Key Concepts
        - MITRE ATT&CK
            - Matrices Enterprise ATT&CK Matrix
                - Techniques applicable to desktops, servers, and cloud environments
            - ATT&CK for Mobile
                - Tactics and techniques used in attacks targeting mobile devices
            - PRE-ATT&CK Matrix
                - Adversary behavior before an attack begins, such as reconnaissance and planning
            - ATT&CK for ICS (Industrial Control Systems)
                - Techniques targeting industrial environments like energy or manufacturing sectors

        - ATT&CK Navigator
            - Definition
                - A visual tool that maps and analyzes how specific techniques align with threat actors
            - Usage
                - Map techniques used by threat actors like APT29
                - Identify gaps in defenses by comparing known attacker tactics with current security measures
            - Example
                - Highlight techniques commonly used in credential dumping
                - Analyze which defenses are missing and prioritize improvements
    - Defensive Applications
        - Understanding Attack Progression
            - Use the framework to model how attackers move through each stage of an attack
            - Identify vulnerabilities in an organization's defenses at specific stages
        - Enhancing Security Posture
            - Analyze real-world threat actor techniques (e.g., credential dumping)
            - Implement targeted defenses, such as
                - Restricting administrative permissions
                - Monitoring audit logs for credential-related events
                - Setting up alerts for suspicious access attempts

- Customized Defense Strategies
  - Tailor defenses to address high-risk techniques relevant to the organization's environment
  - Utilize insights to fortify areas where attackers are most likely to exploit vulnerabilities.
- Example
  - Credential Dumping
    - Technique
      - Extract stored usernames and passwords
    - Framework Insights
      - Required permissions
      - Platforms affected
      - Detection and mitigation strategies
    - Defensive Actions
      - Enable and monitor audit logs for credential-related events
      - Implement multi-factor authentication
      - Restrict administrative permissions
- Summary
  - The MITRE ATT&CK framework is a powerful tool for understanding and countering cyber threats by categorizing TTPs into structured matrices
    - Key Components
      - Enterprise ATT&CK, ATT&CK for Mobile, PRE-ATT&CK, and ATT&CK for ICS matrices
      - ATT&CK Navigator for visualizing attacker techniques and defense alignment
    - Use Cases

            - Identify vulnerabilities in defenses
            - Strengthen security posture by mapping defenses against known attack methods
            - Develop targeted defense strategies based on real-world threat intelligence
        - By leveraging the MITRE ATT&CK framework, organizations can align their defenses with real-world attack patterns, ensuring comprehensive protection against potential threats

- **26_05: Diamond Model of Intrusion Analysis**
    - Diamond Model of Intrusion Analysis
        - A threat modeling framework that examines cyber incidents by focusing on four main components: adversary, infrastructure, capability, and victim
        - Each component represents a specific aspect of an attack and is visualized as a point on a diamond to show their interconnected nature
        - The Diamond Model provides a structured way to analyze attacks, revealing the relationships and dependencies within an intrusion
    - Core Components
        - Adversary
            - Represents the individual or group responsible for the attack
            - Includes motives, resources, and intent
            - Examples
                - Hacktivist groups, nation-state actors, organized crime groups
        - Infrastructure

            - Refers to the network, tools, or systems used by the adversary to carry out the attack
            - Examples
                - Command-and-Control (C2) servers, phishing sites, malicious domains
        - Capability
            - Encompasses the specific tactics, techniques, and procedures (TTPs) the attacker uses
            - Examples
                - Malware deployment, exploiting software vulnerabilities, social engineering
        - Victim
            - The target of the attack, including the organization, individual, or system impacted
            - Includes specific vulnerabilities exploited by the adversary
            - Examples
                - Corporate servers, customer databases, individuals with privileged access
    - Meta-features of the Diamond Model
        - Timestamp
            - Records when the attack occurred
        - Phase
            - Indicates the attack's stage
        - Result
            - Describes the outcome, such as successful access or data theft
        - Direction

            - Details the network path or connection used
        - Methodology
            - Outlines the attack techniques or tactics
        - Resources
            - Lists assets needed by the adversary, such as tools or exploits
    - Application of the Diamond Model
        - Responsive Scenario
            - Example
                - An organization detects malware on a system
            - Victim Analysis
                - Identifies the compromised system and vulnerabilities exploited
            - Capability Analysis
                - Examines the malware's functions, like a connection to a C2 domain
            - Infrastructure Analysis
                - Traces C2 domains to the attacker's resources
            - Adversary Identification
                - Links infrastructure to a known threat group
        - Proactive Use
            - Phishing Prevention
                - Focus defenses on the victim by implementing email filters
            - Infrastructure Disruption
                - Block malicious IPs at the firewall or redirect suspicious DNS requests
            - Capability Mitigation

            - Implement endpoint security to detect and block malware functions
    - Summary
        - The Diamond Model of Intrusion Analysis offers a structured way to examine cyber incidents by breaking them into four main components
            - Adversary
            - Infrastructure
            - Capability
            - Victim
        - Each component reveals different aspects of the attack, from the attacker’s motives to the specific techniques they use and the systems they target
        - By analyzing these interconnected components, security analysts gain valuable insights into how the attack unfolded and can identify strategic points to strengthen defenses, thereby improving incident response and preventing future threats

- **Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege (STRIDE)**
    - STRIDE Framework
        - A threat modeling tool used to identify and categorize potential security threats within a system across six categories
            - Spoofing
            - Tampering
            - Repudiation
            - Information Disclosure

            - Denial of Service
            - Elevation of Privilege
        - Each category represents a distinct threat type, making STRIDE a comprehensive framework for evaluating security vulnerabilities, especially useful during the design phase of software development
    - STRIDE Categories
        - Spoofing
            - Definition
                - Impersonating another user or system to gain unauthorized access
            - Example
                - Attacker gains access by impersonating a legitimate user
            - Mitigation
                - Use multi-factor authentication (MFA) and strong identity verification techniques
        - Tampering
            - Definition
                - Unauthorized modification of data or processes to alter outcomes or inject malicious content
            - Example
                - Altering a database record or injecting malicious code into an application
            - Mitigation
                - Implement data integrity checks, encryption, and input validation
        - Repudiation

- Definition
  - Denying actions that have been performed, challenging accountability
- Example
  - A user claims they did not perform a transaction to avoid responsibility
- Mitigation
  - Enable secure logging and audit trails to record and verify user actions

■ Information Disclosure

- Definition
  - Unauthorized release or exposure of sensitive information
- Example
  - Data breach exposing customer records or sensitive financial data
- Mitigation
  - Use data encryption in transit and at rest, enforce access controls, and employ monitoring for data leaks

■ Denial of Service (DoS)

- Definition
  - Disrupting system availability, rendering it unusable for legitimate users
- Example
  - Overloading a website with traffic to make it inaccessible
- Mitigation

            - Implement rate limiting, load balancing, and anti-DDoS protection to maintain availability
    - Elevation of Privilege
        - Definition
            - Gaining higher access rights than intended, potentially accessing restricted areas of the system
        - Example
            - A regular user escalating privileges to gain admin access
        - Mitigation
            - Enforce role-based access control (RBAC) and regularly review permissions to prevent unauthorized access
- Application of the STRIDE Framework in Software Development
    - Spoofing
        - Verify user identities using multi-factor authentication (MFA) to prevent impersonation
    - Tampering
        - Apply integrity checks and data validation to ensure data authenticity
    - Repudiation
        - Securely log all actions to create audit trails and ensure accountability
    - Information Disclosure
        - Encrypt sensitive data to prevent unauthorized access
    - Denial of Service
        - Set rate limits to avoid resource overload and maintain availability
    - Elevation of Privilege

            - Implement RBAC to ensure users operate within permitted access levels
    - Summary
        - The STRIDE framework is a comprehensive threat modeling tool that identifies security threats in six categories
            - Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege
        - Each category addresses a specific risk to software security, from impersonation and data modification to unauthorized access and service disruption
        - Using STRIDE during software design helps anticipate potential threats, enabling teams to implement preventive measures early in development

- **Open Web Application Security Project (OWASP)**
    - OWASP
        - A nonprofit organization dedicated to enhancing software security, with a focus on web applications
        - OWASP offers resources, standards, and tools to guide developers and security professionals in building secure applications
        - Key offerings include the OWASP Top 10, Application Security Verification Standard (ASVS), and Zed Attack Proxy (ZAP)
    - Key OWASP Offerings
        - OWASP Top 10
            - Definition
                - A list of the 10 most critical web application security risks
            - Importance

        - Serves as a baseline for identifying and addressing common vulnerabilities in web applications
    - Examples
        - Injection Flaws
            - Vulnerabilities where untrusted data is sent as part of a command or query
        - Broken Authentication
            - Weak or flawed authentication systems that may allow unauthorized access
        - Cross-Site Scripting (XSS)
            - Injection of malicious scripts into trusted websites viewed by other users
    - Usage
        - Provides a standardized framework for assessing application security, often used by enterprises to meet regulatory or compliance standards
- Application Security Verification Standard (ASVS)
    - Definition
        - A framework providing specific security requirements for application design and development
    - Purpose
        - Helps development teams set and verify security objectives to ensure applications meet high-security standards
    - Examples of ASVS Guidance
        - Secure Session Management

                ■ Guidelines to manage user sessions securely
            ○ Data Validation
                ■ Recommendations for validating input data to prevent injection attacks
            ○ Authentication Requirements
                ■ Specifies standards for secure user authentication
        ● Usage
            ○ Used throughout the software development lifecycle (SDLC) to establish detailed security controls
    ■ Zed Attack Proxy (ZAP)
        ● Definition
            ○ A dynamic application security testing (DAST) tool used to identify vulnerabilities in web applications
        ● Functionality
            ○ Automates checks for vulnerabilities, such as SQL injection and XSS, but also allows for manual testing
        ● Usage
            ○ Can be run against applications pre-deployment to catch security issues, similar to conducting a structural inspection before finalizing a product
○ Practical Application of OWASP Resources
    ■ Starting Point
        ● Use the OWASP Top 10 as a checklist to identify general security threats
    ■ Detailed Design

        - Refer to ASVS to build robust security features, ensuring specific controls for critical security aspects
    - Testing and Validation
        - Run ZAP scans to catch potential vulnerabilities before deployment, ensuring there are no hidden security flaws
- Summary
    - The Open Web Application Security Project (OWASP) provides critical resources to help developers and security professionals secure web applications
    - The OWASP Top 10 offers a foundational checklist of common vulnerabilities, while ASVS provides detailed security requirements to guide application design
    - Additionally, ZAP is a dynamic testing tool that identifies security issues in applications
    - Together, these OWASP tools and standards help teams create secure applications by understanding and mitigating common security risks

Attack Surface Determination

Objective 1.4: Perform threat modeling activities

- **Technical Attack Surface**
  - Technical Attack Surface
    - The sum of all potential entry points in an organization’s technology infrastructure that could be exploited by attackers
    - Key components include architecture reviews, data flows, trust boundaries, and code reviews, which collectively help identify and secure vulnerable points
  - Architecture Reviews
    - Definition
      - Evaluations of a system's design and structure to identify potential security weaknesses or misconfigurations
    - Purpose
      - Ensures that the system’s architecture aligns with security goals, spotting potential risks early in the design process
    - Example
      - Identifying centralized storage for sensitive data accessible from multiple networks
      - An architecture review might lead to restructuring this design to segment access
    - Tools
      - Microsoft Threat Modeling Tool, OWASP Threat Dragon

- Data Flows
  - Definition
    - Analysis of how data moves within and across systems to identify where sensitive information could be exposed
  - Purpose
    - Ensures data is securely transmitted and does not cross insecure channels
  - Example
    - Discovering that customer data is transmitted in plain text, prompting the implementation of encryption for secure data transfers
  - Tools
    - Visio, Lucidchart for mapping; specialized security tools for in-depth analysis
- Trust Boundaries
  - Definition
    - Points in a system where data moves between zones with different trust levels, such as between an internal network and the internet
  - Purpose
    - Ensures that data crossing these boundaries is secure, with access controlled to prevent unauthorized entry
  - Example
    - Using firewalls or network segmentation at trust boundaries between internal networks and external entities
  - Tools

                - Azure Trust Boundary Analysis
    - Code Reviews
        - Definition
            - Detailed examination of source code to detect vulnerabilities before deployment
        - Analysis Types
            - Static Analysis
                - Inspects code without execution to identify vulnerabilities
            - Dynamic Analysis
                - Tests code in a controlled environment to observe runtime interactions
            - Side-Channel Analysis
                - Observes indirect outputs, like network traffic
            - Reverse Engineering
                - Analyzes obfuscated code to understand its structure
            - Software Composition Analysis (SCA)
                - Checks third-party components for vulnerabilities
            - Fuzz Testing (Fuzzing)
                - Injects randomized data into code to test input handling
        - Tools
            - Static Analysis
                - SonarQube, Checkmarx
            - Dynamic Analysis
                - Burp Suite, Wireshark
            - Reverse Engineering
                - IDA Pro, OllyDbg

            - Software Composition Analysis
                - Snyk, WhiteSource
            - Fuzzing
                - Peach fuzzer, WSFuzzer
    - Summary
        - The technical attack surface represents all potential entry points in an organization's infrastructure that attackers might exploit
        - Architecture reviews analyze design weaknesses, data flows map secure data transmission, trust boundaries monitor access across security zones, and code reviews inspect vulnerabilities in code
        - By implementing these methods, teams reduce exposure to attacks, ensuring infrastructure is resilient and secure before system deployment

- **Operational Attack Surface**
    - Operational Attack Surface
        - The vulnerabilities that arise from daily operations and human factors within an organization, covering areas such as user risks, unsanctioned assets, and unsanctioned accounts
        - These vulnerabilities, if left unaddressed, can provide attackers with entry points to bypass standard security measures
    - User Factors
        - Definition
            - Risks introduced by users interacting with the organization's network, such as weak passwords, phishing susceptibility, or misuse of privileges
        - Purpose

        - Highlights human-related vulnerabilities, often the weakest link in security
    - Example
        - An employee using a weak password or falling for a phishing email can open a gateway into secure systems
    - Mitigation Strategies
        - Implement Multi-Factor Authentication (MFA) and strong password policies
        - Conduct Security Awareness Training regularly to educate employees on phishing and social engineering
        - Use Phishing Simulators to assess susceptibility to phishing
        - Implement Role-Based Access Control (RBAC) to limit users' permissions
- Enumeration and Discovery of Unsanctioned Assets
    - Definition
        - Identifying unauthorized or unmanaged devices connected to the network
    - Purpose
        - Ensures that only approved, monitored devices with standardized security configurations access the network
    - Example
        - An employee connects an unauthorized personal device to the corporate Wi-Fi, which could spread malware if infected
    - Mitigation Strategies
        - Use Network Discovery and Inventory Tools (e.g., Nmap, Nessus, SolarWinds) to identify all connected devices

        - Implement Network Access Control (NAC) to enforce device compliance with security policies
        - Regularly update asset inventories to ensure prompt identification of new devices
- Enumeration and Discovery of Unsanctioned Accounts
    - Definition
        - Identifying unauthorized, orphaned, or unmanaged user accounts within systems
    - Purpose
        - Prevents unauthorized access through accounts that are not actively monitored or no longer necessary
    - Example
        - An account remains active after an employee leaves, creating an entry point for unauthorized access
    - Mitigation Strategies
        - Conduct Periodic Account Audits to remove or deactivate inactive accounts
        - Use tools like Microsoft Azure Active Directory or Okta for account status tracking and deactivation
        - Employ Identity Governance Solutions (e.g., SailPoint, Oracle Identity Governance) for detecting orphaned accounts
        - Establish standardized Onboarding and Offboarding Processes to prevent unsanctioned accounts
- Summary
    - The operational attack surface involves vulnerabilities arising from everyday user behavior, unauthorized assets, and unmanaged accounts

        - Key areas of focus are user factors, which include human errors like weak passwords or phishing susceptibility; unsanctioned assets, referring to unauthorized devices on the network that lack standard security; and unsanctioned accounts, which are unmanaged accounts that could provide access to malicious actors
        - Identifying and managing these vulnerabilities is essential to reducing exposure to potential attacks and strengthening overall security

- **Organizational Attack Surface**
    - Organizational Attack Surface
        - Encompasses vulnerabilities related to an organization's external partnerships and public digital presence
        - It focuses on risks that arise from third-party access and the visibility of the organization's digital assets to potential attackers
    - Enumeration and Discovery of Third-Party Connections
        - Definition
            - Identifying all external entities (e.g., vendors, partners, service providers) that have access to the organization's systems or data
        - Purpose
            - Recognizes that third-party connections increase an organization's attack surface, as breaches in these external systems can impact the organization directly
        - Example
            - In the 2013 Target breach, attackers infiltrated Target's network through a compromised HVAC vendor, leading to the theft of millions of customers' credit card details

    - Mitigation Strategies
        - Conduct Regular Audits of third-party access to ensure each connection is necessary and secure
        - Use Vendor Security and Risk Management (VSRM) Software to assess third-party vulnerabilities
        - Implement Network Segmentation and Zero-Trust Principles for third-party access to limit the impact of a breach
        - Use automated access revocation tools (e.g., Okta Lifecycle Management, CyberArk) to terminate third-party access when it's no longer needed
- Enumeration and Discovery of an Organization's Public Digital Presence
    - Definition
        - Identifying and mapping all online assets, such as websites, social media profiles, and publicly accessible systems
    - Purpose
        - Ensures the organization is aware of its entire public digital footprint, identifying and securing exposure points against cyber threats
    - Example
        - An organization may have outdated social media accounts or web applications that could be exploited by attackers for unauthorized access or brand damage
    - Mitigation Strategies
        - Conduct Regular Digital Asset Inventories and Security Audits to identify and manage all publicly accessible assets

            - Use tools like Shodan and Censys to scan for publicly exposed devices and services
            - Set up Automated Alerts for unauthorized brand name usage
            - Keep Online Profiles and Public-Facing Systems updated to reduce exposure to vulnerabilities
            - Perform regular Vulnerability Scans and Penetration Tests on public-facing applications to identify and mitigate potential threats proactively
- Summary
    - The organizational attack surface includes risks associated with third-party connections and the organization's public digital footprint
    - First, third-party connections represent the external vendors or partners with system access, requiring regular audits and security assessments to manage risks
        - Tools like VSRM software and access revocation systems help secure these connections
    - Second, the public digital presence includes all internet-facing assets, such as websites and social media profiles, which must be continuously monitored for exposure points
        - By leveraging digital asset inventories, vulnerability scans, and monitoring tools like Shodan, organizations can strengthen defenses against external threats
    - Together, these practices minimize the vulnerabilities tied to the organizational attack surface, ensuring both external partnerships and public-facing assets are secure

- **Cloud Attack Surface**
    - Cloud Attack Surface
        - The vulnerabilities and entry points associated with an organization's internal and external cloud infrastructure, where security risks may arise from misconfigured resources, publicly accessible assets, unmonitored services, and malicious processes within the cloud environment
    - Enumeration and Discovery of Internally Facing Assets
        - Definition
            - Identifying cloud-based resources accessible only within an organization's network, such as databases, virtual machines, application servers, storage systems, and internal APIs
        - Purpose
            - Ensures that internal assets, which should remain private, are not accidentally exposed due to misconfigurations
        - Example
            - In the 2023 U.S. No Fly List leak, a server was misconfigured, unintentionally exposing sensitive data to the public
        - Mitigation Strategies
            - Use Cloud Management Tools (e.g., AWS Config, Azure Policy) to enforce configuration compliance
            - Conduct Regular Security Assessments and Internal Audits to confirm that only authorized users can access these assets
    - Enumeration and Discovery of Externally Facing Assets
        - Definition

        - Identifying cloud resources that are accessible to the public, such as web applications, public APIs, storage buckets, DNS servers, and CDNs
    - Purpose
        - Reduces exposure by securing publicly accessible resources, which are prime targets for attackers
    - Example
        - Public APIs with weak authentication can expose internal data if accessed by unauthorized parties
    - Mitigation Strategies
        - Use tools like Shodan and Censys to scan for public cloud resources
        - Implement Strict Access Controls (e.g., OAuth) and apply regular Updates and Patches
        - Employ cloud-native solutions (e.g., AWS Inspector, Google Cloud Security Scanner) to monitor public assets
- Cloud Services Discovery
    - Definition
        - Cataloging all cloud services in use across multi-cloud environments to identify potential vulnerabilities or misconfigurations
    - Purpose
        - Provides visibility into active services and their security configurations, reducing risk from unmonitored or misconfigured services
    - Example

        - A misconfigured database that unintentionally allows access to unintended users can expose sensitive information
    - Mitigation Strategies
        - Use Cloud Security Posture Management (CSPM) tools (e.g., Prisma Cloud, Lacework, AWS CloudTrail) to automate inventory and monitor for vulnerabilities
        - Set up alerts for suspicious configuration changes and misconfigurations
- Malicious Cloud Services and Daemons
    - Definition
        - Unauthorized or harmful background processes that maintain control over cloud resources, allowing attackers to perform malicious actions undetected
    - Purpose
        - Identifying and removing malicious daemons is critical to prevent unauthorized access, persistent backdoors, or data exfiltration
    - Example
        - Attackers install a malicious daemon within a virtual machine, which periodically sends data to an external server without triggering alerts
    - Mitigation Strategies
        - Use Advanced Threat Detection Tools (e.g., CrowdStrike, AWS GuardDuty, Azure Security Center) to monitor for unusual process behavior

                - Regularly conduct Forensic Investigations and use Endpoint Detection and Response (EDR) solutions to detect and neutralize malicious services
    - Summary
        - The cloud attack surface includes vulnerabilities within an organization's internal and external cloud infrastructure
        - It involves
            - Internally Facing Assets
                - Private resources that must be safeguarded to avoid accidental exposure
            - Externally Facing Assets
                - Publicly accessible resources that require strict access controls and regular updates to mitigate risk
            - Cloud Services Discovery
                - Cataloging and monitoring cloud services to identify misconfigurations or unmonitored assets
            - Malicious Cloud Services and Daemons
                - Detecting harmful background processes to prevent unauthorized control and data exfiltration
        - Each component of the cloud attack surface requires consistent monitoring to secure vulnerabilities
        - Using specialized tools and regular audits, organizations can reduce exposure to threats and maintain a resilient cloud environment

- **Organizational Change Attack Surface**
    - Organizational Change Attack Surface

        - Refers to security vulnerabilities introduced during periods of transition within an organization, such as staffing changes, mergers, acquisitions, and divestitures
        - Each of these changes can impact security by creating gaps in access control, integrating varying security protocols, or leaving sensitive data exposed
    - Staffing Changes
        - Definition
            - The addition or departure of employees, introducing risks around account management
        - Risks
            - Orphaned accounts from departed employees may retain access, or new employees might receive excessive permissions, leaving access control gaps
        - Example
            - An employee leaves, but their account remains active, potentially allowing unauthorized access
        - Mitigation Strategies
            - Use Identity and Access Management (IAM) tools (e.g., Okta, Azure Active Directory) for streamlined onboarding/offboarding
            - Enforce Automated Deactivation of accounts for departing employees
            - Implement Access Control Policies to manage permissions for new hires based on role
    - Mergers
        - Definition

        - The integration of two companies into a single entity, combining networks, systems, and data
    - Risks
        - Different security protocols may create vulnerabilities; unaligned access controls could lead to unauthorized data access
    - Example
        - Employees from one company gain access to confidential data from the other due to inadequate access control integration
    - Mitigation Strategies
        - Conduct a Comprehensive Security Audit to align security protocols
        - Implement Network Segmentation (e.g., Cisco's ISE, firewalls) to control access
        - Use SIEM Tools for real-time monitoring of abnormal activity during integration
- Acquisitions
    - Definition
        - When one organization takes control of another, requiring system and data integration
    - Risks
        - Misconfigurations or differing security standards in the acquired organization could introduce vulnerabilities
    - Example
        - The acquired company's network includes outdated servers, potentially exposing the larger network to attacks
    - Mitigation Strategies

            - Perform Security Assessments (e.g., with Rapid7 or Nessus) to scan for vulnerabilities pre-integration
            - Carefully review and adjust Oncoming Access Controls using IAM tools to ensure only necessary access
            - Align security protocols before finalizing integration to prevent exposure
    - Divestitures
        - Definition
            - The sale or separation of a portion of a business, assets, or subsidiaries
        - Risks
            - Potential for data exposure if resources are not properly secured or decommissioned
        - Example
            - Failure to decommission shared servers during a divestiture may allow continued access to sensitive data by the separated entity
        - Mitigation Strategies
            - Use Data Loss Prevention (DLP) tools (e.g., Symantec DLP, McAfee Total Protection) to monitor and prevent unauthorized data transfers
            - Conduct a Security Audit to confirm shared resources are properly reconfigured or decommissioned
            - Plan data separation thoroughly to prevent inadvertent access or data leakage
    - Summary

        - The organizational change attack surface includes security risks that emerge during staffing changes, mergers, acquisitions, and divestitures
        - Each type of transition can
            - Staffing Changes
                - Create gaps in access control if accounts are mismanaged
            - Mergers
                - Lead to vulnerabilities if differing security protocols are not aligned
            - Acquisitions
                - Introduce misconfigurations or legacy vulnerabilities from acquired systems
            - Divestitures
                - Risk data exposure if resources are not properly decommissioned or segregated
        - By implementing IAM systems, SIEM tools, and DLP solutions, organizations can reduce exposure and secure data integrity during these critical organizational transitions

- **Modeling with an Existing System**
    - Modeling with an Existing System
        - A process to evaluate potential threats and vulnerabilities specific to an organization's current IT environment
        - It identifies exploitable weaknesses and recommends appropriate security controls to mitigate them
    - Steps in Threat Modeling for an Existing IT System
        - Understanding System Architecture and Operations

    - Purpose
        - Identify the structure, interactions, and operational characteristics of the current IT environment
    - Approach
        - Map out the system architecture, including hardware and software components
        - Analyze how data flows between components and identify potential entry points
        - Determine any reliance on specific technologies (e.g., databases, web servers) that may introduce unique vulnerabilities
    - Outcome
        - Provides a clear picture of the system's strengths and weaknesses, focusing on areas most at risk
- ■ Identifying Relevant Threats
    - Purpose
        - Determine specific types of attacks that the current system is vulnerable to, based on architecture and operations
    - Examples
        - SQL Injection
            - ■ Particularly relevant for systems relying on databases
        - Cross Site Scripting (XSS)
            - ■ A major concern for web-based applications interacting with external users
        - Distributed Denial of Service (DDoS)

            - A threat for public-facing applications or services
        - Approach
            - Use threat intelligence and security frameworks (e.g., OWASP) to identify common threats for each component type
        - Outcome
            - A list of potential attack vectors specific to the organization's IT environment
    - Implementing Mitigating Controls
        - Purpose
            - Reduce the likelihood or impact of identified threats through specific security measures
        - Examples of Controls
            - Multi-Factor Authentication (MFA)
                - Protects against unauthorized access, especially important for remote access to sensitive systems
            - Encryption
                - Secures sensitive data at rest and in transit, making it unreadable without decryption keys
            - Intrusion Detection Systems (IDS)
                - Monitors for suspicious activity and alerts security teams
            - Intrusion Prevention Systems (IPS)
                - Automatically blocks malicious traffic, complementing IDS
            - Network Segmentation

            - Isolates critical or vulnerable systems, reducing exposure to unauthorized access
        - Approach
            - Prioritize controls based on identified threats. Implement layers of security for comprehensive protection
        - Outcome
            - A customized set of defenses tailored to the current IT system's vulnerabilities and threats
- Special Considerations for Legacy Systems
    - Challenges
        - Often lack modern security features
        - May have unpatched vulnerabilities or weak authentication
        - Can complicate integration with new security tools
    - Additional Controls
        - MFA
            - Adds a layer of security to outdated access controls
        - Network Segmentation
            - Isolates legacy systems to minimize potential exposure
        - Encryption
            - Protects sensitive data within legacy environments
        - IPS/IDS
            - Actively monitors and blocks suspicious traffic, protecting legacy systems from exploitation
- Summary
    - Modeling threats in an existing IT system requires
        - Architecture Understanding

            - Mapping components, data flow, and interactions within the system to identify entry points
          - Identifying Relevant Threats
            - Assessing which vulnerabilities are most relevant based on the system's architecture
          - Mitigating Controls
            - Implementing security measures like MFA, encryption, IDS, IPS, and network segmentation to address identified threats
          - Legacy Systems
            - Applying additional security layers and controls to secure outdated systems
        - Regular assessments help ensure that controls remain effective, adapting to any changes in the threat landscape

- **Modeling without an Existing System**
    - Modeling without an Existing System
        - A process to identify and mitigate potential threats and vulnerabilities in a system that is still in development or not yet deployed
        - It allows an organization to proactively plan security controls based on how the system is envisioned to function, ensuring a secure design from the start
    - Steps in Threat Modeling for a Non-Existing IT System
        - Anticipating System Architecture
          - Purpose

        - Envision how the system will be structured to identify possible vulnerabilities
    - Considerations
        - Deployment Environment
            - Will the system be cloud-based, on-premise, or hybrid?
        - Third-Party Interactions
            - Will it interface with external systems or services?
    - Examples
        - Cloud-based systems may require additional controls for cloud resources, such as securing APIs to prevent unauthorized access
    - Outcome
        - A foundational understanding of potential entry points for attackers, helping to identify areas needing strong security measures
- Identifying Data Types and Sensitivity
    - Purpose
        - Determine what data the system will handle and the level of protection required
    - Approach
        - Identify if the system will manage sensitive data, such as personal information, financial records, or intellectual property
    - Examples

        - Systems handling customer credit card data are vulnerable to breaches and thus need strong encryption controls
    - Outcome
        - A prioritized list of data protection needs, guiding controls like data encryption to secure sensitive information both at rest and in transit
- Analyzing User Interactions
    - Purpose
        - Define who will interact with the system and how, to anticipate potential access-related risks.
    - Approach
        - Assess the types of users (e.g., employees, customers, third parties) and the level of access each will require
    - Examples
        - Role-Based Access Control (RBAC) can restrict access based on user roles, limiting exposure of sensitive data
    - Outcome
        - Identification of access control requirements, supporting security measures like MFA and RBAC to protect against unauthorized access
- Assessing Integration Points
    - Purpose
        - Identify where the system will connect with other IT systems, as integrations can expand the attack surface
    - Approach

            - Map out integration points, especially connections with third-party APIs or services
        - Examples
            - APIs require strong authentication, rate limiting, and monitoring to prevent unauthorized access and abuse
        - Outcome
            - A clear understanding of integration-related vulnerabilities, informing controls to secure data exchanges and prevent API misuse
- Mitigating Controls for a Non-Existing System
    - Multi-Factor Authentication (MFA)
        - Purpose
            - Protects against unauthorized access by requiring two or more authentication factors
        - Example
            - Employees logging in from remote locations need to enter both a password and a verification code sent to their mobile device
        - Outcome
            - Enhanced security for user access, minimizing the risk of unauthorized entry
    - Encryption
        - Purpose
            - Secures sensitive data, ensuring it remains unreadable even if accessed by unauthorized parties
        - Example

- Encrypting customer data both when stored (data at rest) and when transmitted (data in transit)

- Outcome
  - Protection for data confidentiality, ensuring compliance with data security standards for sensitive information

■ Network Segmentation

- Purpose
  - Isolates different sections of the network to contain potential breaches
- Example
  - Creating separate network segments for various parts of the system, so attackers cannot easily move through the network
- Outcome
  - Reduced spread of attacks, enabling better control over data flows and limiting attacker access within the system

○ Summary

■ Modeling threats for a non-existing system requires envisioning the future system's architecture, data types, user interactions, and integration points

■ Key controls include

- MFA
  - Enhances security by requiring multiple authentication factors
- Encryption
  - Protects sensitive data, ensuring it's secure at all stages

        - Network Segmentation
            - Limits attackers’ ability to navigate across network segments
    - By proactively identifying these risks and implementing appropriate controls, organizations can ensure a secure foundation from the system’s inception, minimizing vulnerabilities once the system is deployed

Monitoring and Response

Objective 4.1: Analyze data to enable monitoring and response activities

- **Aggregate Data Analysis**
    - Aggregate Data Analysis
        - The process of gathering, combining, and examining large volumes of data from various sources to detect patterns, trends, and potential security threats
        - It aids in efficiently managing data to detect vulnerabilities and prioritize responses
    - Audit Log Reduction
        - Purpose
            - Filters out unnecessary or low-priority logs to focus on critical events
        - Process
            - High-Volume Filtering
                - Routine or benign events (e.g., successful logins) are removed to highlight anomalies (e.g., repeated login failures)
            - Challenge
                - False positives can lead to unnecessary investigations
        - Tools
            - Security Information and Event Management (SIEM) systems like Splunk, which aggregate, normalize, and filter logs

        - Outcome
            - Enhanced detection of potential threats by reducing the log volume, making analysis more manageable
    - Data Correlation
        - Purpose
            - Connects related events from multiple data sources for comprehensive incident understanding
        - Example
            - Combining logs from failed logins and traffic spikes can indicate a coordinated brute force attack
        - Tools
            - SIEM tools like IBM QRadar or ArcSight that unify event data from different systems
        - Outcome
            - Improved incident response by providing context and detecting patterns across systems
    - Data Prioritization
        - Purpose
            - Assigns importance to security issues, allowing teams to address high-impact threats first
        - Approach
            - Evaluate the potential impact of incidents to determine which events require immediate action
        - Example
            - Repeated failed logins on a system with sensitive data might take priority over minor login issues on a non-critical system

        - Outcome
            - Focused use of resources on threats that pose the highest risk
    - Data Trends
        - Purpose
            - Identifies recurring patterns over time, highlighting vulnerabilities
        - Example
            - Detecting a pattern of unauthorized access attempts during specific times
        - Tools
            - Tools like Splunk and Elasticsearch for visualizing and tracking long-term patterns
        - Outcome
            - Early detection of recurring attack vectors, enabling proactive defense enhancements and reducing the likelihood of repeated attacks
    - Practical Application of Aggregate Data Analysis
        - Audit Log Reduction
            - Prioritize and filter logs to manage security data efficiently
        - Correlation
            - Link events across systems for complete incident insight.
        - Prioritization
            - Address high-risk incidents immediately, preventing impactful breaches
        - Trends
            - Analyze recurring patterns to anticipate and mitigate future attacks

- Summary
  - Aggregate Data Analysis involves gathering and analyzing data from various sources to manage and detect security threats
  - By implementing audit log reduction to highlight critical events, correlation to connect data, prioritization to focus on high-risk incidents, and trend analysis to identify recurring patterns, organizations can enhance their security posture and streamline threat detection and response

- **Threat Intelligence Sources**
  - Threat Intelligence Sources
    - Information streams that provide insights into emerging threats, vulnerabilities, and attacker tactics
    - These sources are used by organizations to enhance their defenses and respond proactively to cyber threats
  - Key Concepts
    - Threat Intelligence Feeds
      - Definition
        - Real-time streams of data providing insights into emerging threats
      - Examples
        - Public Feeds
          - Open-source platforms like AlienVault's Open Threat Exchange (OTX)
        - Private Feeds

            - Paid services offering targeted intelligence, such as ISACs
        - Use Cases
            - Include Indicators of Compromise (IOCs) like malicious IPs, file hashes, and domains
            - Automate defense integration using STIX and TAXII protocols
        - Example Application
            - Blocking known ransomware command-and-control IP addresses using firewall rules
    - Common Vulnerabilities and Exposures (CVEs)
        - Definition
            - Standardized identifiers for publicly known vulnerabilities
        - Details
            - Includes identifier (e.g., CVE-2023-12345), description, affected systems, and severity scores
            - Maintained by MITRE Corporation and linked to databases like the National Vulnerability Database (NVD)
        - Severity Assessment
            - Uses the Common Vulnerability Scoring System (CVSS) (e.g., 0.0 - 10.0)
        - Example
            - CVE-2021-34527 ("PrintNightmare") allowed remote code execution, requiring immediate patching
    - Bug Bounty Programs
        - Definition

        - Programs incentivizing security researchers to report vulnerabilities responsibly
    - How It Works
        - Researchers submit vulnerabilities directly to vendors for monetary rewards
        - Facilitated by platforms like HackerOne and Bugcrowd
    - Rewards
        - Vary by severity and organization (e.g., $500 for minor bugs to $1,000,000 for critical issues)
    - Example
        - Apple offers up to $1,000,000 for vulnerabilities leading to zero-click remote code execution
- Third-Party Reports and Logs
    - Reports
        - Generated by cybersecurity firms, government agencies, or research organizations
        - Provide analysis on specific threats or trends (e.g., Verizon Data Breach Investigations Report)
    - Logs
        - Provided by MSSPs or cloud services (e.g., AWS access logs)
        - Integrated into SIEM tools for enhanced monitoring and detection
    - Use Cases
        - Combining internal and external data for a holistic view of security posture

      - Tracking access to cloud resources for anomaly detection
- Examples of Practical Applications
  - Threat Intelligence Feeds
    - Automate defense updates using threat feeds and protocols like STIX/TAXII
    - Example
      - Blocking ransomware-related IPs identified in private threat feeds
  - CVE Use
    - Prioritize patching based on CVSS scores
    - Example
      - Addressing critical vulnerabilities like CVE-2021-34527
  - Bug Bounty Programs
    - Encourage ethical hacking to uncover vulnerabilities
    - Example
      - Detecting zero-click exploits in mobile operating systems
  - Third-Party Reports and Logs
    - Analyze trends and integrate logs for comprehensive security insights
    - Example
      - Using AWS CloudTrail logs to monitor unauthorized access
- Summary
  - Threat intelligence sources provide critical insights to identify and respond to cybersecurity threats.
  - Key Components
    - Threat Intelligence Feeds

            - Real-time data for proactive defense
        - CVEs
            - Standardized vulnerability tracking
        - Bug Bounty Programs
            - Incentivized discovery of vulnerabilities
        - Third-Party Reports and Logs
            - Comprehensive threat and incident analysis
    - Use Cases
        - Enhance defenses using automated feeds
        - Prioritize patching based on CVSS scores
        - Leverage external insights for improved threat response
    - By utilizing threat intelligence sources effectively, organizations can maintain a proactive stance against evolving cyber threats

- **System Log Sources**
    - System Log Sources
        - Logs generated by network infrastructure and enterprise devices, providing detailed records of activities and events
        - These logs support monitoring and threat detection across the organization’s network and cloud environments
    - Infrastructure Device Logs
        - Purpose
            - Records activities from network hardware like routers, firewalls, and switches
        - Role in Security

        - Detects network health and identifies unusual traffic or unauthorized access attempts
        - Highlights network anomalies such as repeated access attempts from suspicious IPs.
    - Example
        - Firewall logs that show repeated access attempts to restricted network areas
    - Implementation
        - Ensure a log policy covers both on-premises and cloud-based infrastructure to avoid missed data
- Endpoint Logs
    - Purpose
        - Track activities on individual devices (e.g., computers, mobile devices)
    - Log Types
        - Windows
            - Security, Application, System, Setup, and Forwarded Events logs
        - Linux
            - Logs stored in the /var/log directory, covering authentication, system events, specific applications, and cron jobs
    - Role in Security
        - Enables detection of unauthorized access, application issues, and system failures
    - Example

            - Detecting unauthorized login attempts from endpoint Security logs
        - Outcome
            - Comprehensive visibility into user and device activities, supporting compliance and security monitoring
    - Application Logs
        - Purpose
            - Captures events within software applications, aiding in troubleshooting and security checks
        - Log Organization
            - Windows
                - Stored in the Event Viewer, organized by severity (Information, Warning, Error, Critical)
            - Linux
                - Stored in /var/log, often in specific subdirectories like /var/log/httpd/ for web servers
        - Role in Security
            - Identifies software issues, failed login attempts, and suspicious user activity
        - Example
            - Using web server logs to track HTTP requests and identify SQL injection attempts
        - Outcome
            - Insight into application performance and security, allowing for prompt response to critical errors or threats
    - Cloud Security Posture Management (CSPM) Tools

        - Purpose
            - Monitor cloud environments to identify misconfigurations, vulnerabilities, and compliance violations
        - Examples
            - AWS Security Hub, Azure Security Center.
        - Role in Security
            - Ensures cloud resources adhere to best practices, protecting against risks like open S3 buckets or permissive IAM roles
        - Example
            - AWS Security Hub detecting overly permissive S3 bucket permissions and notifying administrators
        - Outcome
            - Supports secure cloud operations by maintaining visibility and compliance within the cloud infrastructure
    - Practical Application of System Log Sources
        - Infrastructure Device Logs
            - Monitor and address network traffic anomalies to preempt unauthorized access
        - Endpoint Logs
            - Track user and device behavior for signs of unauthorized access and other threats
        - Application Logs
            - Review for software performance issues and security incidents, prioritizing responses based on severity
        - CSPM Tools

            - Continuously evaluate cloud configurations to avoid misconfigurations that could expose sensitive data
    - Summary
        - System log sources play a critical role in tracking, monitoring, and securing network and cloud operations within an organization
        - Infrastructure device logs reveal network anomalies; endpoint logs provide insight into device activities; application logs monitor software performance and security; and CSPM tools ensure cloud configurations remain secure
        - Together, these components offer comprehensive visibility and support proactive security and compliance management

- **Vulnerabilities and Data Security**
    - Vulnerabilities and Data Security
        - The practice of identifying system weaknesses and protecting sensitive data from unauthorized access or loss
        - Core tools include vulnerability scans to detect security flaws and Data Loss Prevention (DLP) tools to prevent unauthorized data transmission
    - Vulnerability Scans
        - Purpose
            - To identify potential security flaws, such as unpatched software or misconfigurations, across a network
        - Importance
            - Regular scans ensure systems are secure and compliant with industry standards

        - Automated, scheduled scans (e.g., weekly) catch new vulnerabilities as they arise
    - Common Tools
        - OpenVAS
            - An open-source tool widely used for network scans
        - Nessus
            - A commercial tool with a comprehensive vulnerability database and plugin customization for targeted scans (e.g., PCI DSS compliance)
        - Qualys
            - A cloud-based solution that continuously scans infrastructure and integrates with asset management systems for real-time monitoring
    - Example
        - Running Nessus weekly to detect vulnerabilities and prioritize them for patching
- Data Loss Prevention (DLP) Tools
    - Purpose
        - To prevent sensitive data from being accidentally or maliciously shared with unauthorized parties
    - Key Features
        - Data Monitoring
            - Tracks data in transit, at rest, and in use across enterprise systems
        - Policy Enforcement
            - Defines how sensitive data should be handled

            - Common actions include
                - Alerting
                    - Notifies security teams of potential data leaks
                - Blocking
                    - Stops the transmission of sensitive data
                - Quarantining
                    - Holds data for further review before release
                - Tombstoning
                    - Replaces sensitive content with placeholders for safer sharing
    - Common Tools
        - Symantec DLP
            - Monitors data across multiple channels (e.g., endpoints, email, cloud)
        - Microsoft Purview
            - Integrates DLP into Microsoft apps (SharePoint, Teams, Outlook), making it easy to secure data within Microsoft's ecosystem
    - Example
        - Microsoft Purview stopping an email containing credit card numbers and alerting the security team for investigation
- Practical Application
    - Vulnerability Scans
        - Schedule regular scans (e.g., weekly or post-update) to detect new vulnerabilities

            - Prioritize patching based on scan reports to address the most severe issues first
        - Data Loss Prevention (DLP)
            - Use DLP policies to enforce rules around sensitive data. Set up alerts, blocks, or tombstoning to prevent unauthorized data sharing
            - Monitor compliance with regulatory requirements (e.g., GDPR, HIPAA)
    - Summary
        - Vulnerabilities and data security practices focus on protecting IT systems and sensitive data
        - Vulnerability scans identify weaknesses in systems, and DLP tools monitor data flows to prevent unauthorized access or sharing
        - Together, these tools support proactive security management, helping organizations protect against threats and ensure data privacy

- **Behavior Baselines and Analytics**
    - Behavior Baselines and Analytics
        - The practice of establishing normal activity patterns for networks, systems, users, and applications to detect deviations indicating security threats
        - It includes defining baselines for network, systems, user behavior, and application/service usage
    - Network Baselines
        - Definition

            - Typical patterns of data traffic, such as common data flows, connection types, and traffic volumes over time
        - Importance
            - Helps identify unusual patterns that may signal threats like data exfiltration or DDoS attacks
            - For example, sudden surges in data to unknown IPs could indicate malware communication with a command and control server
        - Tools
            - Snort
                - An Intrusion Detection System (IDS) for monitoring traffic
            - Suricata
                - Can function as both an IDS and Intrusion Prevention System (IPS)
        - Example
            - Detecting a spike in network traffic to unfamiliar IPs and flagging it as a potential exfiltration attempt
    - System Baselines
        - Definition
            - Regular usage of system resources, including CPU, memory, and disk I/O
        - Importance
            - Detects abnormal resource usage which may indicate malware or system issues (e.g., memory leaks)
        - Tools
            - PerfMon
                - Monitors performance on Windows systems

        - Datadog, Prometheus
            - For continuous system monitoring and alerting on deviations
    - Example
        - A sudden CPU usage spike to 100% indicating malware or an unauthorized process
- User Baselines
    - Definition
        - Regular user activities like login times, accessed files, and device usage
    - Importance
        - Identifies unusual user behavior which could suggest compromised credentials or insider threats
    - Tools
        - Splunk User Behavior Analytics (UBA): Tracks and analyzes user activity over time
    - Example
        - A user logging in from an unusual location at a late hour, indicating a potential credential compromise
- Application and Service Baselines
    - Definition
        - Expected usage patterns and performance metrics for applications/services, such as response times and error rates
    - Importance
        - Detects deviations that may indicate performance issues or attacks like DoS

    - Tools
        - AppDynamics
            - Monitors and tracks application performance over time
    - Example
        - A sudden increase in error rates suggesting application performance issues or exploitation
- Practical Application
    - Network Baselines
        - Monitor real-time traffic against established baselines for any deviations
        - Set alerts for unusual traffic surges to detect DDoS or exfiltration attempts
    - System Baselines
        - Establish baseline resource usage metrics and detect anomalies in CPU/memory consumption
    - User Baselines
        - Identify abnormal login locations or access patterns indicating possible account compromise
    - Application and Service Baselines
        - Monitor applications for deviations in response times or transaction volumes
- Summary
    - Behavior baselines and analytics are crucial for identifying normal activity across networks, systems, users, and applications, enabling detection of deviations that may indicate security threats

- Network, system, user, and application baselines each offer a unique focus area for proactive security monitoring, helping organizations quickly spot and address potential issues before they escalate

- **SIEM Event Management**
    - SIEM Event Management
        - The process of collecting, analyzing, and responding to security event data from various sources in real-time, helping to detect and mitigate potential threats in enterprise environments
    - Event Parsing
        - Definition
            - Breaking down raw event data into a structured format to enable consistent analysis by the SIEM
        - Purpose
            - Normalizes data from different sources (e.g., firewalls, servers) for easier interpretation and correlation
        - Example
            - Parsing log data from a firewall into standardized fields like IP address, port number, and timestamp
        - Benefit
            - Creates a consistent dataset, allowing security teams to reduce noise and quickly focus on critical security issues
    - Event Duplication
        - Definition
            - Filtering out repeated alerts or events to prevent unnecessary noise in the SIEM system

- Purpose
  - Minimizes alert fatigue by consolidating identical or repetitive alerts, focusing security efforts on unique incidents
- Example
  - Grouping multiple login failure alerts from the same user within a specific timeframe into one alert
- Benefit
  - Improves the efficiency of security teams by reducing redundant data and helping analysts prioritize real threats

○ Identification of False Positives and False Negatives
- Definition
  - Ensuring accurate threat detection by distinguishing between benign activity and actual threats
- False Positive
  - An event incorrectly flagged as a threat (e.g., a legitimate login flagged as suspicious)
- False Negative
  - A real threat that goes undetected (e.g., malware download missed by restrictive rules)
- Purpose
  - Reduces wasted time on benign incidents while ensuring true threats are detected and addressed
- Example
  - Adjusting correlation rules to prevent legitimate activities from triggering false alarms
- Benefit

            - Helps security analysts focus on real threats, enhancing the organization's overall security posture by minimizing both undetected threats and benign misidentifications
    - Summary
        - SIEM Event Management involves collecting, analyzing, and responding to security event data in real time
        - This process includes
            - Event Parsing
                - Organizing data into a structured format for consistency across sources
            - Event Duplication
                - Filtering redundant alerts to focus on unique incidents
            - Identification of False Positives and False Negatives
                - Ensuring accurate threat detection and minimizing unnecessary responses
        - These concepts help security teams efficiently manage event data, focus on actual threats, and maintain strong security oversight

- **SIEM Data Management**
    - SIEM Data Management
        - The process of organizing and maintaining security event data for effective analysis, detection, and long-term storage within a Security Information and Event Management (SIEM) system, ensuring ongoing visibility and compliance with regulations
    - Data from Non-Reporting Devices
        - Definition

        - Systems or endpoints that fail to send their logs or security event data to the SIEM
    - Purpose
        - Identifies any devices that stop reporting data, as this may create security blind spots in the network
    - Example
        - A firewall stops sending logs due to connectivity issues, potentially allowing threats to go undetected
    - SIEM Functionality
        - Configures alerts for non-reporting devices, often with heartbeat checks to confirm device connectivity
    - Benefit
        - Ensures continuous visibility across all systems and endpoints, allowing security teams to quickly identify and resolve issues
- Data Retention
    - Definition
        - Policies and practices that determine how long SIEM stores collected event data
    - Purpose
        - Balances storage efficiency, regulatory compliance, and support for forensic investigations.
    - Example
        - Retaining security logs for at least seven years in the healthcare industry to comply with HIPAA
    - SIEM Functionality

            - Configures retention policies based on regulatory needs; archives or deletes logs after a specified period
        - Benefit
            - Maintains necessary historical records for incident investigations and compliance while optimizing storage capacity
    - Summary
        - SIEM Data Management encompasses
            - Data from Non-Reporting Devices
                - Ensures that the SIEM receives logs from all devices, alerting the security team if any device stops reporting, reducing security blind spots
            - Data Retention
                - Follows specific policies to store event data for a defined period, ensuring compliance, support for audits, and optimizing storage
        - Together, these components help organizations maintain continuous security visibility, meet regulatory standards, and manage SIEM storage efficiently

- **Alerting**
    - Alerting
        - The process of notifying security teams about potential threats or suspicious activities using predefined rules and event triggers within a security system, such as a Security Information and Event Management (SIEM) platform
    - Vulnerability Alerts

- Definition
  - Notifications triggered when potential security weaknesses in a system are detected
- Purpose
  - Provides timely information to security teams to address vulnerabilities before they are exploited
- Example
  - An alert for an unpatched web server vulnerability that could be targeted by attackers
- Associated Tools
  - Vulnerability scanners, SIEM systems, and file integrity monitoring (FIM) tools

- False Positives and False Negatives
  - False Positives
    - Definition
      - Alerts triggered by normal, non-malicious activities that are incorrectly flagged as threats
    - Impact
      - Can overwhelm security teams, leading to alert fatigue
  - False Negatives
    - Definition
      - Real threats that go undetected by the system
    - Impact
      - Leaves systems vulnerable as legitimate security incidents may go unnoticed
  - Example

        - An intrusion detection system fails to detect a sophisticated attack, resulting in a false negative
    - Prevention
        - Regular fine-tuning of security systems and continuous rule updates
- Malware Alerts
    - Definition
        - Alerts generated upon the detection of malicious software, such as viruses, worms, or ransomware
    - Purpose
        - Enables prompt response to contain and mitigate malware infections
    - Detection Methods
        - Signature-based detection
            - Identifies known malware based on established signatures
        - Heuristic analysis
            - Recognizes suspicious patterns in code
        - Behavioral monitoring
            - Detects unusual system activities, like unauthorized file modifications
    - Example
        - A malware alert that quarantines ransomware upon detection to prevent its spread across the network
- Alert Failures
    - Definition

            - Occurs when an alert is delayed, missed, or not generated, potentially allowing threats to progress unchecked
        - Causes
            - Misconfiguration
                - Incorrectly set up rules or missing indicators of compromise
            - Poor prioritization
                - Failing to rank alerts based on severity, leading to overlooked critical issues
            - Overloaded systems
                - High data volumes overwhelming security tools, causing delays in alert processing
        - Example
            - A SIEM unable to process high volumes of data in real-time, resulting in delayed alerts
        - Mitigation Strategies
            - Regular maintenance, correct configuration, prioritization of alerts, and use of scalable SIEM solutions
    - Summary
        - Vulnerability Alerts
            - Notify security teams of system weaknesses for timely remediation
        - False Positives
            - Represent non-malicious activity flagged as a threat
        - False Negatives

            - Represent undetected threats, with both requiring careful management
        - Malware Alerts
            - Inform teams of infections to allow prompt containment and mitigation
        - Alert Failures
            - Happen due to misconfigurations, poor prioritization, or system overload, necessitating continuous tuning and reliable monitoring solutions to reduce risks
        - Ensuring accurate and effective alerting helps maintain a strong security posture, providing visibility into threats and minimizing risks associated with missed alerts

- **Alert Prioritization Factors**
    - Alert Prioritization Factors
        - Criteria used to determine the urgency of security alerts, ensuring the most critical threats are addressed first in an organization's security incident response process
    - Alert Criticality
        - Definition
            - The severity level of an alert based on how impactful and urgent the threat is to the organization
        - Purpose
            - Allows categorization of alerts into levels (e.g., warning, low, medium, high, critical) to ensure the most severe issues are prioritized

        - Example
            - A critical alert for ransomware on a database versus a low-level alert for failed login attempts
    - Alert Impact
        - Definition
            - Assesses the potential damage or disruption a threat could cause if left unaddressed
        - Purpose
            - Ensures high-impact threats, such as those affecting critical data or services, are resolved promptly
        - Example
            - An alert for data exfiltration from a customer database is prioritized over a minor configuration error in a non-critical system
    - Asset Type
        - Definition
            - Refers to the nature and importance of the asset (e.g., database, production server) involved in the alert
        - Purpose
            - Higher priority is given to alerts affecting high-value assets, such as financial databases or essential infrastructure
        - Example
            - Prioritizing an alert on a production server over an alert on a test environment server
    - Residual Risk
        - Definition

        - The remaining risk after all security measures and controls have been applied
    - Purpose
        - Focuses on alerts related to areas with high residual risk, where vulnerabilities remain despite existing defenses
    - Example
        - High residual risk in an area where multi-factor authentication has been bypassed
- Data Classification
    - Definition
        - The sensitivity and confidentiality level of data involved in an alert, with classified or confidential data given higher priority
    - Purpose
        - Ensures alerts involving sensitive data (e.g., PHI, financial records) are prioritized due to potential legal and financial repercussions
    - Example
        - Prioritizing an alert related to a system handling financial data over one with general information
- Summary
    - Alert prioritization factors enable security teams to efficiently address threats by focusing on
        - Alert Criticality
            - The severity level of the alert
        - Alert Impact
            - The potential damage if the alert is unaddressed
        - Asset Type

            - The importance of the asset affected
        - Residual Risk
            - The remaining risk after controls are applied
        - Data Classification
            - Sensitivity of data involved, ensuring classified information receives urgent attention
    - These factors collectively guide security teams in managing threats, protecting critical assets, and safeguarding sensitive information efficiently and effectively

- **Reporting and Metrics**
    - Reporting and Metrics
        - The process of collecting and presenting security data to measure performance, identify trends, and guide decision-making
    - Visualization Products
        - Definition
            - Tools that use charts, graphs, and visual aids to transform complex security data into easily interpretable formats
        - Purpose
            - Simplifies large volumes of security logs, alerts, and metrics, making it easier for both security teams and executives to understand and make informed decisions
        - Example
            - A line graph showing network intrusions over time to track trends and determine if incidents are increasing or decreasing
        - Common Tools

            - Tableau, Grafana
    - Dashboards
        - Definition
            - Centralized, real-time displays of key security metrics, providing a unified view (single pane of glass) for tracking incidents and responses
        - Purpose
            - Aggregates data from multiple sources to enable quick identification of threats, monitor ongoing incidents, and assess overall security health
        - Example
            - A dashboard showing active threats, endpoint protection status, and unusual network traffic to enable fast response to critical events
        - Common Tools
            - Splunk, Kibana, Microsoft Sentinel
    - Summary
        - Reporting and metrics help organizations measure security performance, spot trends, and make data-driven decisions
        - Key tools include
            - Visualization Products
                - Simplify complex data through visual formats like charts and graphs, enhancing understanding of security metrics
            - Dashboards
                - Offer real-time, centralized views of security data, helping teams quickly assess and respond to potential threats

■ Together, these tools enable security teams to act on critical data, maintain a proactive stance, and ensure defenses remain effective

Threat-hunting

Objectives:

- 2.3 - Integrate appropriate controls in the design of a secure architecture
- 4.3 - Apply threat-hunting and threat intelligence concepts

- **29_02: Indicators of Attack**
    - Indicators of Attack (IoA)
        - Observable behaviors and patterns that suggest an ongoing or imminent attack, based on the Tactics, Techniques, and Procedures (TTPs) adversaries use
    - Tactics, Techniques, and Procedures (TTPs)
        - Definition
            - Methods and strategies attackers use in carrying out attacks
        - Purpose
            - Enable threat hunters to detect and respond to IoAs by recognizing patterns of attacker behavior
    - Specific TTPs and Threat Hunting Process
        - Initial Access
            - Definition
                - First step attackers take to penetrate a network
            - Common Techniques
                - Phishing, exploiting vulnerabilities, using stolen credentials
            - Detection Tools

        - Email filtering, multi-factor authentication (MFA), SIEM systems
    - Example IoAs
        - Surge in login attempts from unfamiliar IPs, increased phishing reports
- Privilege Escalation
    - Definition
        - Attackers attempt to elevate access to higher privilege levels
    - Common Techniques
        - Exploiting software, using malicious scripts, modifying user privileges
    - Detection Tools
        - Endpoint detection tools like CrowdStrike
    - Example IoAs
        - Suspicious PowerShell commands, unusual privilege changes
- Lateral Movement
    - Definition
        - Attackers moving between systems to reach high-value targets
    - Common Techniques
        - Exploiting Remote Desktop Protocol (RDP), using administrative tools like PsExec
    - Detection Tools
        - SIEM systems, network traffic analyzers

        - Example IoAs
            - Unusual RDP connections, administrative tool usage on atypical systems
    - Exfiltration
        - Definition
            - Attackers attempting to steal data and transmit it outside the network
        - Common Techniques
            - Using encrypted channels, disguising data, leveraging legitimate cloud services
        - Detection Tools
            - Data Loss Prevention (DLP) tools
        - Example IoAs
            - Unusual outbound traffic, large file transfers to external services
    - Command and Control (C2)
        - Definition
            - Attackers maintaining communication with compromised systems for remote control
        - Common Techniques
            - DNS tunneling, connecting to malicious servers
        - Detection Tools
            - Intrusion Detection Systems (IDS), network monitoring tools
        - Example IoAs

- Unexpected outbound connections to suspicious domains, unusual DNS queries

- Persistence
    - Definition
        - Techniques allowing attackers to maintain access to a system even after detection
    - Common Techniques
        - Creating new administrative accounts, installing backdoors, modifying system settings
    - Detection Tools
        - Endpoint Detection and Response (EDR) tools like Carbon Black, CrowdStrike
    - Example IoAs
        - Unauthorized account creation, unexpected system configuration changes

- Summary
    - Indicators of Attack (IoAs) are patterns indicating ongoing or imminent attacks, based on TTPs that outline the adversary's goals and methods
    - Key TTPs to watch for include
        - Initial Access
            - Detecting early penetration attempts
        - Privilege Escalation
            - Monitoring for elevated access
        - Lateral Movement
            - Observing attackers moving within the network
        - Exfiltration

      - Tracking data leaving the network
    - Command and Control (C2)
      - Detecting remote attacker communication
    - Persistence
      - Preventing attackers from regaining access after detection
  - By focusing on IoAs and using tools like EDR and SIEM, organizations can proactively detect and mitigate potential threats, maintaining a secure network environment

- **Behavior and Data Analysis**
  - Behavior and Data Analysis
    - Examining patterns and anomalies in system and user activities to detect potential threats or malicious behavior
  - Internal Reconnaissance
    - Definition
      - Activities an attacker conducts after gaining initial access to gather information about the network, such as system vulnerabilities and critical assets
    - Purpose
      - Allows attackers to map out potential targets for further exploitation
    - Threat Hunting
      - Identifies internal reconnaissance as a sign of breach and attempts to detect it through network log analysis for unusual scanning or unauthorized access
    - Examples

            - Unusual scanning activity from unknown devices
            - User accounts accessing areas of the network that they normally don't
    - Hypothesis-Based Searches
        - Definition
            - Searches driven by specific theories or assumptions about potential attacker behavior based on intelligence or knowledge of common attack methods
        - Purpose
            - Provides a focused, strategic approach to identifying Indicators of Compromise (IoCs) or malicious activity
        - Benefits
            - Efficient, targeted searches that help uncover stealthy or advanced attacks
        - Example
            - Hypothesis
                - Attackers are exploiting a known vulnerability in a web application
            - Investigation might focus on logs for unusual queries, unauthorized access attempts, or signs of privilege escalation
    - User Behavior Analytics (UBA)
        - Definition
            - Analysis of user activities to detect deviations from established behavior patterns, signaling potential insider threats or compromised accounts
        - Purpose

            - Identifies anomalies in user behavior, flagging suspicious activities for further investigation
        - Benefits
            - Can detect subtle indicators of compromise that may be missed by traditional signature-based systems
        - Example
            - An employee's account downloads large amounts of data from a sensitive database after hours
            - UBA flags this as abnormal for that user, prompting further investigation
    - Summary
        - Behavior and Data Analysis focuses on identifying signs of malicious activities by monitoring system and user behaviors
        - Key components include
            - Internal Reconnaissance
                - Detecting signs of attackers gathering internal network information
            - Hypothesis-Based Searches
                - Targeted searches guided by theories on how attackers may exploit vulnerabilities
            - User Behavior Analytics (UBA)
                - Monitoring for deviations from normal user behavior to identify potential insider threats or compromised accounts
        - By leveraging these concepts, organizations enhance their ability to detect threats proactively and prevent attacks from progressing

- **Internal Intelligence Sources**
    - Internal Intelligence Sources
        - Data and insights gathered from within an organization’s own network to identify and understand potential threats
    - Adversary Emulation Engagements
        - Definition
            - Simulations designed to mimic real-world attacker behavior within a controlled environment
        - Purpose
            - Test and improve the organization’s defenses by observing how specific attackers might exploit vulnerabilities and move within the network
        - Implementation
            - Security teams design scenarios based on known threat actors or attack vectors
            - Tools simulate common attack behaviors, such as lateral movement and privilege escalation
            - Security team monitors network response, identifies gaps, and enhances defenses
        - Benefit
            - Mimics real attacker behavior, allowing teams to adjust defenses before a genuine attack, improving the organization’s security posture
    - Honeypots
        - Definition

- Decoy systems designed to attract attackers and study their behavior by mimicking valuable assets

■ Purpose
  - Detect unauthorized access, gather information on attack methods, and divert attackers from real systems

■ Implementation
  - Placed within a visible area of the network, like a screened subnet (DMZ)
  - Configured to log interactions and monitor suspicious behavior

■ Example
  - A honeypot mimics a file server containing sensitive data
  - Attacker attempts to access it, allowing security teams to observe attack methods

■ Benefit
  - Collects real-time intelligence on attack strategies, helping security teams refine defenses for production systems

○ Honeynets

■ Definition
  - A controlled and monitored network of multiple honeypots that simulate a full network environment

■ Purpose
  - Lure attackers into a simulated network to reveal complex, multi-stage attack tactics

■ Implementation
  - Designed to appear as a legitimate network with different components like web servers and databases

            - Usually placed near public-facing servers or within an internal network
        - Example
            - A honeynet mimics an enterprise's internal network, with user workstations and databases
            - Attackers move laterally within the honeynet, allowing the security team to observe escalation and pivoting tactics
        - Benefit
            - Provides in-depth intelligence on complex attack methods, such as privilege escalation and lateral movement, which helps strengthen real network defenses
    - Summary
        - Internal Intelligence Sources utilize data and insights from within the organization's network to enhance threat detection and response
        - Key components include
            - Adversary Emulation Engagements
                - Simulating real-world attacker behaviors to test and improve defenses
            - Honeypots
                - Decoy systems that attract attackers to study their tactics without risking real assets
            - Honeynets
                - Networks of honeypots that simulate an entire network, providing a comprehensive view of attacker methods and allowing for better defense against complex, multi-stage attacks

        - Using these sources, organizations gain valuable insights into potential attack patterns, enabling proactive adjustments to their security posture

- **Detection and Threat-hunting Enablers**
    - Detection and Threat-Hunting Enablers
        - Tools and infrastructure that support the discovery and investigation of threats within a network
    - Sensor Placement
        - Definition
            - Strategic deployment of monitoring devices, like intrusion detection systems (IDS) or intrusion prevention systems (IPS), throughout a network
        - Purpose
            - Capture data at key network points (e.g., network perimeter, internal segments) for threat detection
        - Example
            - Tools like Zeek placed between corporate networks and the internet monitor inbound and outbound traffic
            - Sensors in server zones monitor internal communication (east-west traffic)
        - Benefit
            - Increases visibility into various network layers, ensuring suspicious activities are detected across critical areas
    - Continuous Monitoring
        - Definition

            - Constant surveillance of the network, systems, and endpoints to detect unauthorized or unusual activities in real-time
        - Purpose
            - Detect threats as they evolve, allowing for immediate investigation and response
        - Example
            - SIEM tools, like Splunk, monitor data sources (e.g., firewall logs, antivirus software) to detect spikes in unusual activity, such as high outbound traffic, indicating data exfiltration
        - Benefit
            - Allows security teams to act quickly when threats are detected, minimizing potential damage
    - Alerting
        - Definition
            - Setting up predefined rules to trigger notifications when specific events or conditions are detected
        - Purpose
            - Enable security teams to respond swiftly to suspicious activities, like multiple failed logins or unusual data transfers
        - Example
            - SIEMs like Palo Alto Networks alert on conditions, such as a high rate of failed login attempts, suggesting a brute-force attack
        - Benefit
            - Provides early warning to security teams, helping prevent threats from escalating undetected
    - Centralized Logging

        - Definition
            - Aggregating log data from various network devices, servers, applications, and security tools into a unified platform
        - Purpose
            - Simplify analysis and enable correlation of events across different systems
        - Example
            - The ELK Stack (Elasticsearch, Logstash, Kibana) centralizes logs from multiple sources, such as IDS/IPS, web servers, and endpoint protection systems
        - Benefit
            - Speeds up detection by providing a comprehensive view of the network's activities, facilitating effective threat-hunting
    - Summary
        - Detection and Threat-Hunting Enablers provide the essential tools and infrastructure for identifying and investigating potential threats
        - Key components include
            - Sensor Placement
                - Deploys monitoring devices at critical points to capture data for threat analysis
            - Continuous Monitoring
                - Keeps a watch on the network and systems to detect abnormal behaviors in real time
            - Alerting
                - Uses predefined criteria to notify security teams of suspicious activity, allowing timely responses

- Centralized Logging
  - Consolidates data from multiple sources into a single platform, streamlining the analysis process
- By using these enablers, organizations maintain comprehensive visibility across their network, enhancing their ability to detect, investigate, and mitigate potential threats

- **External Intelligence Sources**
  - External Intelligence Sources
    - Data and insights gathered from outside an organization to help identify potential threats or vulnerabilities
  - Open-Source Intelligence (OSINT)
    - Definition
      - The collection of publicly available information from sources like websites, social media, and forums
    - Purpose
      - Helps organizations understand their external exposure and how attackers might view their network
    - Example
      - Tools like Maltego and Shodan show public-facing systems and vulnerabilities that might be visible to attackers
    - Benefit
      - Allows organizations to proactively address vulnerabilities by viewing their digital footprint through the perspective of potential attackers
  - Information Sharing and Analysis Centers (ISACs)

- Definition
  - Organizations that facilitate cybersecurity information-sharing among companies within specific industries
- Purpose
  - Enable collaboration to understand and respond to cyber threats by pooling resources and knowledge
- Example
  - The Financial Services ISAC (FS-ISAC) helps banks share information about emerging threats
  - Health-ISAC supports hospitals in sharing intelligence on healthcare-specific threats
- Benefit
  - Reduces response time to threats by enabling industry-wide information sharing, improving resilience against attacks

- Reliability Factors
  - Definition
    - Evaluation of the credibility and accuracy of external intelligence sources
  - Purpose
    - Ensures that intelligence is trustworthy, timely, and likely to be accurate
  - Example
    - The Admiralty Scale assigns two ratings: one for the source’s reliability (graded A to F) and one for the information’s accuracy (graded 1 to 6)

        - "A1" indicates a highly reliable source with credible information, while "F6" indicates low reliability and unverified data
    - Benefit
        - Reduces the risk of acting on inaccurate intelligence, avoiding wasted resources and unnecessary disruptions
- Dark Web Monitoring
    - Definition
        - Tracking and analyzing activity on hidden parts of the internet often used for illegal activities, like selling stolen data
    - Purpose
        - Provides early indicators of breaches or potential attacks by monitoring criminal marketplaces
    - Example
        - Tools like DarkOwl track data sales or hacker discussions targeting specific organizations
    - Benefit
        - Offers early detection of breaches or planned attacks, allowing organizations time to respond before damage escalates
- Summary
    - External Intelligence Sources aid organizations in identifying and mitigating threats by providing insights from outside their network.
    - Key components include
        - OSINT
            - Collects public data to reveal an organization's external vulnerabilities
        - ISACs

            - Enable collaboration within industries to share threat intelligence and improve defense
        - Reliability Factors
            - Assess the credibility of intelligence to avoid acting on unverified or inaccurate data
        - Dark Web Monitoring
            - Tracks criminal activities, alerting organizations to data leaks or targeted attacks
    - Together, these sources offer a well-rounded approach to external threat intelligence, allowing organizations to proactively strengthen their security posture and stay ahead of potential threats

- **Threat Intelligence Platforms (TIPs)**
    - Threat Intelligence Platforms (TIPs)
        - Tools that collect, analyze, and distribute threat data to help organizations detect and respond to security risks more effectively
    - Key Components of Threat Intelligence Platforms (TIPs)
        - Third-Party Vendors
            - Definition
                - TIPs aggregate data from third-party vendors such as Recorded Future, FireEye, and CrowdStrike
            - Purpose
                - Provide intelligence on new threats, vulnerabilities, and attacker tactics
            - Types
                - Open-Source

            - Community-driven feeds, customizable but may require more manual configuration
        - Proprietary
            - Curated, premium threat feeds with advanced integration features.
    - Example
        - TIPs may integrate intelligence from FireEye about a new malware strain targeting specific industries, which can then be cross-referenced with internal logs to detect similar activity in the organization
- Integration with Internal Data
    - Definition
        - TIPs enhance threat detection by combining external intelligence with internal security data (e.g., logs, alerts)
    - Purpose
        - Increases precision in detecting and responding to threats by providing external context for internal alerts
    - Example
        - A TIP can cross-check internal email logs with phishing intelligence from FireEye to detect known malicious email addresses, blocking phishing emails before they reach users
- Automatic Updates and Defense Integration
    - Definition

            - TIPs can automate updates to security defenses, such as firewalls and endpoint detection systems, with new threat intelligence
        - Purpose
            - Ensures defenses are always current and provides proactive threat protection
        - Example
            - Indicators of compromise (IOCs) from third-party intelligence are automatically added to the SIEM or firewall, keeping defenses up-to-date against new threats without manual intervention
    - Threat Prioritization
        - Definition
            - TIPs allow security teams to prioritize threats based on relevance and impact to the organization
        - Purpose
            - Focuses resources on the most critical threats, ensuring efficient use of security efforts
        - Example
            - A TIP can prioritize threats targeting specific sectors, like healthcare, allowing a healthcare organization to address those threats first
- Benefits of Threat Intelligence Platforms
    - Improved Detection
        - Combining internal data with third-party intelligence enhances accuracy in identifying threats

        - Proactive Defense
            - Automation of IOCs and real-time threat intelligence updates ensures defenses are continuously updated
        - Efficient Resource Allocation
            - Threat prioritization enables security teams to focus on the most relevant and high-risk threats
        - Comprehensive Threat Visibility
            - By using both internal and external data, TIPs provide a fuller understanding of potential threats
    - Summary
        - Threat Intelligence Platforms (TIPs) empower organizations to detect and respond to security threats by aggregating and analyzing threat data from various sources
        - By integrating insights from third-party vendors, such as FireEye and CrowdStrike, TIPs enrich internal security data, improving detection accuracy and response speed
        - With features like automated updates to security defenses and threat prioritization, TIPs enable organizations to maintain proactive, efficient threat management and allocate resources to the most critical risks

- **IoC Sharing**
    - Indicator of Compromise (IoC) Sharing
        - The exchange of threat data between organizations to improve detection and response efforts
        - IoC data includes malicious IP addresses, file hashes, domain names, etc
    - Structured Threat Information eXpression (STIX)

        - Definition
            - A standardized language used to represent and communicate cyber threat information in a structured format
        - Purpose
            - Ensures consistency in describing threats, making it easier to share and understand IoC data across organizations
        - Usage
            - Integrates with SIEM systems or Threat Intelligence Platforms (TIPs) to identify suspicious behavior
        - Example
            - A financial institution using STIX to format threat data about phishing campaigns for easy sharing among banks
    - Trusted Automated eXchange of Intelligence Information (TAXII)
        - Definition
            - A protocol for securely exchanging threat data over the internet
        - Purpose
            - Automates the sharing of IoCs, allowing real-time updates on emerging threats
        - Usage
            - Integrated with TIPs and SIEM systems to pull in real-time threat intelligence
        - Example
            - A healthcare organization using TAXII to share ransomware threat information across the sector for timely defense
    - Automated Indicator Sharing (AIS)
        - Definition

            - A program by the U.S. Department of Homeland Security (DHS) for real-time sharing of cyber threat indicators between public and private sectors
        - Purpose
            - Enhances national cybersecurity by enabling rapid sharing of IoCs, like IP addresses and file hashes, at machine speed
        - Usage
            - Security tools configured to receive AIS feeds for proactive threat detection
        - Example
            - An energy provider using AIS to receive real-time updates on cyber threats targeting critical infrastructure
    - Benefits of IoC Sharing
        - Improved Threat Detection
            - Organizations gain insights into emerging threats from shared IoCs
        - Enhanced Collaboration
            - Consistent language (STIX) and secure protocols (TAXII) streamline threat communication between sectors
        - Proactive Defense
            - Real-time data from AIS enables organizations to act on the latest threats quickly
        - Standardization
            - STIX and TAXII ensure threat data is consistently formatted and securely exchanged
    - Summary

- IoC sharing improves cybersecurity by allowing organizations to exchange threat data in real time
- Key concepts in IoC sharing include
  - STIX
    - Standardizes threat information for clear communication
  - TAXII
    - Secures IoC data exchange over the internet
  - AIS
    - Facilitates real-time, automated threat sharing between public and private sectors
- These tools and protocols enable efficient collaboration, helping organizations proactively defend against emerging threats by leveraging shared IoCs

- **Rule-Based Languages**
  - Rule-based Languages
    - Used to create patterns and detection rules that identify specific malicious activities or behaviors within systems and networks
  - Sigma
    - Definition
      - A generic rule-based language for defining security event patterns across multiple platforms
    - Purpose
      - Allows security teams to write cross-platform rules to detect malicious activities within SIEMs and other monitoring tools
    - Format

            - YAML, making rules easy to read and share
        - Usage
            - Deploy within SIEMs to monitor for suspicious command-line executions, alerting security teams to potential reconnaissance
    - Yet Another Recursive Acronym (YARA)
        - Definition
            - A rule-based language used to identify and classify malware by defining specific patterns in files
        - Purpose
            - Automates malware detection by matching defined strings, byte sequences, or characteristics associated with malware
        - Usage
            - YARA rules are implemented in endpoint detection systems, scanning files for signs of malware or reconnaissance commands
    - Rita
        - Definition
            - An open-source framework that detects network anomalies, particularly identifying unusual behaviors like beaconing
        - Purpose
            - Focuses on analyzing network flow data to detect anomalies associated with hidden threats
        - Usage
            - Integrated with network monitoring tools (e.g., Zeek) to process logs and uncover stealthy attacks by analyzing long-running connections or regular, suspicious communication patterns (e.g., DNS tunneling)

        - Value
            - Helps detect advanced persistent threats (APTs) by identifying unusual traffic patterns undetectable by traditional security tools
    - Snort
        - Definition
            - An open-source Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) that analyzes network traffic in real-time using predefined rules
        - Purpose
            - Detects and blocks network-based attacks, such as malware payloads or exploitation attempts
        - Usage
            - Monitors network traffic, applying custom or community-generated rules to alert or prevent intrusion attempts
    - Summary
        - Rule-based languages enable organizations to define detection patterns for identifying malicious activities
        - Sigma
            - Defines security event patterns across SIEMs, focusing on system-level indicators
        - YARA
            - Identifies and classifies malware through file pattern recognition, assisting malware researchers
        - Rita
            - Detects network anomalies, focusing on hidden or stealthy attacks
        - Snort

            - Analyzes network traffic in real-time, applying IDS/IPS rules to prevent exploitation attempts
        - Together, these tools cover various layers of security, from files and processes to network traffic, enhancing an organization's ability to detect and respond to threats effectively

- **Counterintelligence and Operational Security**
    - Counterintelligence and Operational Security
        - Identifying and mitigating efforts by adversaries to gather intelligence or exploit weaknesses within an organization
    - Cyber Deception
        - Definition
            - A proactive defense strategy that uses decoy systems, false data, or misleading pathways to confuse and divert attackers away from actual assets
        - Purpose
            - Misleads adversaries by making it difficult to distinguish real targets from fake ones, protecting critical assets and gathering intelligence on attackers' methods
        - Key Terms
            - Honeypot
                - A decoy system that appears as a legitimate target, designed to attract and observe attackers' behavior
            - Honeynet

        - A network of honeypots simulating a real network environment, allowing for observation of various attack strategies
      - Honeyfile
        - A fake document placed on a system to be accessed by attackers; triggers alerts if opened or moved
      - Honeytoken
        - A fake credential or data item that, when accessed, signals unauthorized activity
    - Value
      - By monitoring attacks on decoys, organizations can adjust defenses on actual systems, prepare for potential threats, and share intelligence with the broader security community
  - Monitoring and Response
    - Definition
      - Continuously observing network traffic and system activity to detect and respond to potential security incidents in real-time
    - Purpose
      - Detects and responds to suspicious behavior as soon as it occurs, allowing organizations to neutralize threats quickly
    - Tools and Techniques
      - Intrusion Detection Systems (IDS)
        - Detects unauthorized access attempts
      - Security Information and Event Management (SIEM)
        - Aggregates and analyzes data from various sources to detect anomalies and generate alerts

        - Behavioral Analytics
            - Identifies unusual patterns, such as unexpected login times or unauthorized data access
    - Response Actions
        - Isolating compromised systems Blocking malicious network traffic Alerting security teams for further investigation
    - Example
        - A SIEM system detects unusual access patterns on a user account and flags it for investigation, allowing security analysts to secure the account before damage occurs
    - Importance
        - Early detection and rapid response reduce potential harm, such as data loss or system downtime, especially in environments handling sensitive data
- Summary
    - Counterintelligence and Operational Security involve strategies to prevent adversaries from exploiting an organization's weaknesses
    - Cyber Deception
        - Uses fake assets, like honeypots and honeyfiles, to mislead attackers and gather insights into their tactics
        - This proactive approach confuses adversaries and protects real assets
    - Monitoring and Response
        - Ensures continuous observation of network activities to detect and respond to threats in real-time, using tools like SIEM systems and IDS to identify anomalies and enable prompt actions

- Together, these strategies enhance an organization’s defenses by disrupting attackers’ methods and enabling timely detection and response to threats

Indication Analysis

Objective 4.4: Analyze data and artifacts in support of incident response activities

- **Infrastructure Analysis**
  - Infrastructure Analysis
    - The process of examining hardware, software, and network components to detect vulnerabilities or signs of compromise, ensuring the security and integrity of an organization's systems
  - Joint Test Action Group (JTAG) Interfaces
    - Purpose
      - A hardware interface standard used for testing and debugging embedded systems
    - Functionality
      - Provides direct access to embedded system internals
      - Identifies hardware-level defects, vulnerabilities, or unauthorized modifications
    - Use Cases
      - Example
        - Analyzing a network switch's firmware for signs of tampering using JTAG access
      - Allows recovery of detailed hardware-level logs for diagnostics
    - Importance
      - Ensures the hardware layer of an infrastructure is secure, reducing the attack surface

- Host Analysis
  - Purpose
    - Examines endpoints like servers and workstations to detect signs of compromise or vulnerabilities
  - Focus Areas
    - Indicators of Compromise (IoCs)
      - Malware, misconfigurations, and unusual behaviors
    - Logs, file changes, and system processes
  - Tools
    - Sysmon
      - Monitors system activities
    - EnCase and Autopsy
      - Perform forensic investigations
  - Use Cases
    - Example
      - Investigating slow endpoint performance and unauthorized file modifications to uncover malware
  - Importance
    - Protects the network by ensuring all endpoints are secure and free of malicious activity
- Network Analysis
  - Purpose
    - Monitors data traffic, communication patterns, and connections to identify malicious activity or vulnerabilities
  - Focus Areas

                - Anomalies like unusual data transfers or unauthorized connections
                - Detecting ongoing attacks such as data breaches or denial-of-service (DoS) attacks
            - Tools
                - Wireshark
                    - Captures detailed packet data (header and payload)
                    - Provides in-depth investigation but requires significant storage and time
                - NetFlow
                    - Captures metadata about traffic (source/destination IPs, ports, protocols)
                    - More efficient for long-term monitoring but offers less detail
            - Use Cases
                - Example
                    - Detecting unusual traffic spikes during off-peak hours and uncovering malware exfiltrating sensitive data
            - Importance
                - Ensures real-time detection and response to protect the organization’s data
        - Summary
            - Infrastructure Analysis
                - A critical security practice that examines hardware, software, and network components for vulnerabilities or compromises.
            - Key Concepts

                - JTAG Interfaces
                    - Debugging and testing embedded systems
                    - Helps identify hardware-level anomalies and vulnerabilities
                - Host Analysis
                    - Scrutinizes endpoints to detect malware or misconfigurations
                    - Protects the network by isolating compromised devices
                - Network Analysis
                    - Monitors traffic and connections to uncover malicious activities
                    - Uses tools like Wireshark and NetFlow for real-time threat detection
            - Benefits
                - Enhances security by ensuring all system layers—hardware, software, and networks—are protected from vulnerabilities and attacks

- **Metadata Analysis**
    - Metadata Analysis
        - The examination of hidden data within files, media, or communications to uncover details about origin, manipulation, or potential malicious intent
        - Metadata can reveal information such as file creation dates, editing history, user permissions, or the transmission path of communications
    - Files and Filesystems
        - Purpose

        - Examines metadata to track file creation, modification, and access history
    - Key Details
        - File creation and modification dates
        - User permissions and ownership
        - Access control details
    - Tools
        - Linux Command
            - stat /path/to/file.txt (Displays timestamps, ownership, permissions, and other metadata)
        - Forensic Tools
            - Autopsy, FTK Imager (Extract metadata from disk images for forensic investigations)
    - Use Case
        - Example
            - Detecting unauthorized file modifications by analyzing timestamps for changes made outside normal working hours
    - Importance
        - Helps trace security incidents, such as malware infections or unauthorized file alterations
- Images, Audio, and Video
    - Purpose
        - Extracts metadata from media files to uncover details like device origin, GPS coordinates, and editing history
    - Key Details

        - Camera model and recording device
        - GPS location data (if available)
        - Editing history and timestamps
    - Tools
        - ExifTool
            - Command
                - exiftool /path/to/image.jpg (Displays metadata like creation date, editing history, and GPS coordinates)
    - Use Case
        - Example
            - Verifying an image's authenticity by detecting hidden metadata that indicates it was edited with specific software
    - Importance
        - Critical for digital forensics and authenticity verification, especially in legal or investigative contexts
- Email Headers
    - Purpose
        - Analyzes email headers to trace the origin and transmission path of messages
    - Key Details
        - Fields
            - Received, Message-ID, Return-Path, IP addresses
        - Authentication records
            - DKIM, SPF, and DMARC
    - Tools

                - MXToolbox
                    - Decodes and interprets email headers
                - Email Clients
                    - Built-in header analysis tools (e.g., Gmail, Outlook)
            - Use Case
                - Example
                    - Identifying phishing emails by detecting discrepancies in sender IP addresses or missing authentication records
            - Importance
                - Detects spoofing, phishing, and rerouted emails, preventing further compromises
        - Summary
            - Metadata Analysis
                - A method to extract and interpret hidden data within files, media, or communications to uncover details about origin, manipulation, or malicious activity
            - Key Concepts
                - Files and Filesystems
                    - Reveals file creation and modification details
                    - Detects unauthorized changes or tampering
                - Images, Audio, and Video
                    - Extracts GPS coordinates, device details, and editing history
                    - Verifies the authenticity of media files
                - Email Headers
                    - Tracks email origins and paths through servers

      - Detects spoofing, phishing, or rerouted communications
    - Benefits
      - Enhances investigations by uncovering anomalies
      - Provides critical insights for tracing suspicious activities
      - Strengthens defenses against tampering, phishing, and other malicious actions

- **Volatile and Non-volatile Storage Analysis**
  - Volatile and Non-Volatile Storage Analysis
    - The process of examining temporary and permanent storage to uncover evidence of malicious activity or compromise
    - Volatile storage contains short-lived data, lost when the system is powered off, while non-volatile storage persists data even after power loss
  - Order of Volatility
    - Definition
      - The sequence in which digital evidence should be collected during forensic investigations, prioritizing the most ephemeral data
    - RFC 3227 Guidelines
      - Registers and CPU cache
        - Short-lived data, lost immediately when powered off
      - Routing tables, ARP cache, process tables, kernel statistics, and RAM
        - Contains active system and network processes
      - Temporary file systems or swap space
        - Holds intermediate data used by operating systems

                - Disk drives
                    - Permanent storage for files and logs
                - Remote logging and monitoring data
                    - Archived remotely for additional context
                - Physical configurations and network topologies
                    - Documented hardware and network layouts
                - Archival media
                    - Long-term storage like backup tapes
            - Importance
                - Ensures highly volatile data is preserved before it is lost, followed by stable data like disks and logs
        - Forensic Imaging
            - Definition
                - Creating an exact bit-by-bit replica of a storage device to preserve evidence without altering the original
            - Key Details
                - Includes visible files, slack space, unallocated space, and deleted files
                - Ensures integrity and admissibility in court
            - Tools
                - dd
                    - Standard Unix/Linux command for creating bit-by-bit copies
                    - Command
                        - dd if=/dev/sdX of=/path/to/diskimage.img bs=64K conv=noerror,sync

        - dcfldd
            - Enhanced version of dd with added features for forensic work
            - Command
                - dcfldd if=/dev/sdX of=/path/to/diskimage.img bs=64K hash=md5,sha256 hashlog=hashlog.txt
        - FTK Imager
            - Graphical tool for imaging, hashing, and chain of custody management
    - Best Practices
        - Use a write blocker to prevent modifications to the original device
        - Create two images
            - one for secure storage and another for analysis
        - Verify the image with hashing to ensure it matches the original
- Summary
    - Volatile and Non-Volatile Storage Analysis
        - The investigation of temporary and permanent data to detect signs of compromise
        - Volatile Storage
            - Includes CPU cache and RAM; data is lost when powered off
            - Order of Volatility
                - Ensures the most transient data is captured first
        - Non-Volatile Storage
            - Includes hard drives and archival media; data persists after shutdown

            - Forensic Imaging
                - Creates an exact copy of storage for analysis, preserving the integrity of the original data
        - Key Tools and Techniques
            - Follow RFC 3227 for evidence collection
            - Use tools like dd, dcfldd, and FTK Imager for forensic imaging
            - Utilize write blockers to protect original data
        - Benefits
            - Preserves critical ephemeral data
            - Ensures evidence is admissible in legal proceedings
            - Protects data integrity throughout investigations

- **Reverse Engineering**
    - Reverse Engineering
        - The process of breaking down software or hardware components to understand their structure, functionality, and potential vulnerabilities
    - Byte Code
        - Definition
            - A low-level representation of code executed by virtual machines, acting as an intermediary between high-level programming languages and machine code
        - Key Details
            - Platform-independent, executed by virtual machines
            - Example in Java Byte Code
                - 0: iconst_1
                - 1: istore_1

        - 2: iload_1
        - 3: ireturn
      - Comparison with Assembly Language
        - MOV AX, 1
        - MOV [VAR1], AX
        - MOV AX, [VAR1]
        - RET
      - Use in Forensics
        - Analyzing software/malware behavior without original source code
        - Extracted from compiled files like .class (Java), .pyc (Python), or .dll (.NET)
- Binary Code
  - Definition
    - Machine-level instructions directly executed by the computer, represented as ones and zeros
  - Key Details
    - Essential for evaluating executables, firmware, or operating systems
  - Tools
    - binwalk
      - Inspects firmware images to extract components
    - hexdump
      - Displays binary files in hexadecimal format for manual inspection
    - strace

            - Tracks system calls made by a binary
        - ldd
            - Identifies shared libraries relied upon by a binary
    - Example Use
        - binwalk to extract firmware components
        - hexdump to inspect anomalies in binary structure
- Disassembly
    - Definition
        - Converts binary code into assembly language, providing a low-level, human-readable representation of machine instructions
    - Key Details
        - Reveals how software operates at a fundamental level
        - Useful for identifying malicious behavior or vulnerabilities
    - Tools
        - IDA Pro
            - Professional disassembly tool
        - Ghidra
            - Open-source disassembler and reverse engineering tool
        - OllyDbg
            - Debugger for Windows applications
        - GDB
            - Debugger for Unix/Linux systems
    - Key Features
        - Breakpoints
            - Pause execution to examine memory, variables, and register values at critical points

- Decompilation
    - Definition
        - Translates executable binary or bytecode back into a high-level programming language for easier understanding
    - Key Details
        - Produces human-readable code, often resembling the original source
        - More effective for languages like Java, Python, and JavaScript due to structural information retention
    - Tools
        - CFR
            - For Java bytecode
        - uncompyle6
            - For Python .pyc files
        - JEB
            - Advanced Java decompiler
        - Ghidra
            - Supports decompilation alongside disassembly
    - Example Use
        - Reverse engineering malware to uncover hidden functions or malicious logic
        - Simplifies forensic analysis by reconstructing program logic
- Summary
    - Reverse Engineering
        - The study of software or hardware components to uncover their structure, functionality, and vulnerabilities

        - Byte Code
            - Low-level, platform-independent code executed by virtual machines
        - Binary Code
            - Machine-level instructions analyzed using tools like binwalk or hexdump
        - Disassembly
            - Converts binary code into assembly language, aiding low-level analysis
        - Decompilation
            - Translates executable code into a high-level language for easier understanding
    - Use in Forensics
        - Detecting malicious behavior, hidden functions, or vulnerabilities in software
        - Enables detailed analysis of malware, firmware, and proprietary applications
    - Tools
        - binwalk, hexdump, IDA Pro, Ghidra, CFR, uncompyle6, and more
    - Benefits
        - Simplifies the understanding of software behavior
        - Identifies potential threats or security gaps
        - Assists in uncovering hidden or malicious program logic

- **Malware Analysis**
    - Malware Analysis

        - The process of examining malicious software to understand its behavior, potential damage, and identifying Indicators of Compromise (IoCs)
    - Sandboxing
        - Definition
            - Isolating potentially harmful code or applications in a secure environment to observe behavior safely
        - Application
            - Allows analysts to: Prevent broader system compromise
            - Monitor file modifications, registry changes, and network activity
        - Tools
            - Joe Sandbox
            - Cuckoo Sandbox
            - FireEye
        - Benefit
            - Enables controlled execution and insight into malware's capabilities
        - Example
            - An analyst receives a suspicious file and runs it in a Cuckoo Sandbox to observe attempts at external communication or file encryption
    - Malware Detonation
        - Definition
            - Intentionally executing malware in a sandbox to trigger its full functionality
        - Application

            - Identifies hidden or dormant behaviors. Reveals malware’s tactics, techniques, and payloads
        - Tools
            - Cuckoo Sandbox
            - VMware-based virtual environments
        - Benefit
            - Uncovers full operational details of malware, aiding in countermeasure development
        - Example
            - Detonating ransomware to observe encryption processes, targeted files, and communication with command-and-control servers
    - Indicator of Compromise (IoC) Extraction
        - Definition
            - Identifying artifacts left by malware, such as file hashes, IP addresses, and domain names
        - Application
            - Enables detection and mitigation of future attacks
            - Builds threat profiles and facilitates intelligence sharing
        - Tools
            - YARA
            - MISP
            - Splunk
        - Benefit
            - Strengthens defenses against similar malware through detection and blocking

        - Example
            - Extracting an IoC like an IP address from a sandbox analysis and adding it to a firewall blocklist
    - Summary
        - Malware Analysis
            - Examines malware to understand its behavior and damage potential
        - Key Techniques
            - Sandboxing
                - Isolates and observes malware in a controlled environment
            - Malware Detonation
                - Triggers full functionality to reveal behaviors
            - IoC Extraction
                - Identifies artifacts for proactive threat defense
        - Use in Forensics
            - Protects systems from further compromise
            - Builds defenses by studying malware's techniques and sharing IoCs
        - Tools
            - Joe Sandbox, Cuckoo Sandbox, YARA, and MISP
        - Benefits
            - Provides detailed insights into malware behavior
            - Enables proactive defense measures
            - Facilitates intelligence sharing across organizations

- **Code Stylometry**
  - Code Stylometry
    - The process of analyzing a developer's coding style to identify unique patterns for malware attribution or tracing the origin of specific software
  - Variant Matching
    - Definition
      - Identifies similarities between different versions or variants of the same malware family
    - Application
      - Detects modified malware strains that share coding structures with their predecessors
      - Helps security teams quickly determine if malware is a known threat or entirely new
    - Benefit
      - Enables rapid detection and adaptation to evolving malware, saving time and resources
    - Example
      - A ransomware strain introduces a new payload but retains the same propagation code
      - Variant matching links it to its original malware family, helping to create detection rules
  - Code Similarity
    - Definition
      - Compares segments of code across multiple samples to detect shared structures, techniques, or functions
    - Application

        - Discovers connections between different malware families or legitimate software and malicious programs
        - Identifies reused algorithms or plagiarism in software development
    - Benefit
        - Provides broader insights into attacker coding practices and potential vulnerabilities
    - Example
        - A malware sample uses the same encryption algorithm as a known banking trojan
        - Code similarity links the malware to a group targeting financial institutions, aiding in tailored defense strategies
- Malware Attribution
    - Definition
        - Links malware to specific developers, groups, or threat actors based on unique coding patterns and habits
    - Application
        - Helps understand the nature and source of threats, aiding law enforcement and strategic defense
        - Uses specific markers like reused libraries or algorithms for attribution
    - Benefit
        - Enhances predictive security strategies by identifying known attacker tactics, techniques, and procedures (TTPs)
    - Example

            - A piece of ransomware shares unique error-handling techniques with past attacks attributed to a specific group
            - The security team attributes the attack and prepares for future risks from the same threat actor
    - Summary
        - Code Stylometry
            - Analyzes coding styles to identify unique patterns for malware attribution or tracing software origins
        - Key Techniques
            - Variant Matching
                - Detects new malware strains through similarities with older versions
            - Code Similarity
                - Identifies shared structures across different malware samples
            - Malware Attribution
                - Links malicious code to specific developers or groups
        - Use in Cybersecurity
            - Proactively detects malware variants
            - Identifies connections between attacks
            - Aids in forensic investigations and threat actor attribution
        - Benefits
            - Improves incident response times
            - Strengthens defenses against known threat actors
            - Enhances collaboration with law enforcement and industry partners

- **Cloud Workload Protection Platform**
  - Cloud Workload Protection Platform (CWPP)
    - A security solution designed to detect, protect, and respond to threats targeting cloud-based workloads, ensuring consistent security across various cloud environments
  - Detection and Response
    - Definition
      - Continuous monitoring of cloud workloads to identify suspicious behavior and respond to security threats in real-time
    - Application
      - Monitors workloads for unusual activities, such as unauthorized data access or irregular user actions
      - Triggers automated responses, like isolating compromised workloads or revoking user access
    - Benefit
      - Stops threats early, minimizing damage and ensuring swift resolution
    - Example
      - A CWPP detects unusual database access by a compromised workload and automatically isolates it to prevent data exfiltration
  - Integration
    - Definition
      - The ability of a CWPP to seamlessly work with existing cloud infrastructures and security tools
    - Application

        - Connects with platforms like AWS, Azure, and Google Cloud, as well as third-party tools (e.g., SIEMs, CSPMs)
        - Shares threat data across tools to coordinate defenses
    - Benefit
        - Provides a centralized view of risks and ensures consistent enforcement of security policies, eliminating blind spots
    - Example
        - A CWPP detects a threat in AWS and shares the information with Azure Security Center, enabling coordinated responses across all cloud platforms
- Multi-Cloud Environments
    - Definition
        - Configurations where an organization uses multiple cloud service providers to manage workloads, requiring consistent security across all platforms
    - Application
        - Uniformly applies security measures across providers like AWS, Azure, and Google Cloud
        - Protects workloads regardless of the hosting provider
    - Benefit
        - Ensures comprehensive security, avoids vendor lock-in, and reduces complexity in managing multi-cloud environments
    - Example
        - An organization using AWS and Azure relies on a CWPP to ensure consistent threat detection and response across both platforms
- Summary

- Cloud Workload Protection Platform (CWPP)
    - A security solution for safeguarding cloud-based workloads
    - Monitors, detects, and responds to potential threats in real-time
- Key Features
    - Detection and Response
        - Identifies and mitigates suspicious activities quickly to prevent damage
    - Integration
        - Seamlessly connects with cloud infrastructures and tools for coordinated defenses
    - Multi-Cloud Environments
        - Provides uniform security across multiple cloud providers like AWS, Azure, and Google Cloud
- Benefits
    - Enhances threat visibility and response capabilities
    - Prevents vendor lock-in while maintaining robust security
    - Reduces complexity in managing dynamic cloud environments

Incident Response

Objectives:

- 1.2 - Perform risk management activities
- 4.4 - Analyze data and artifacts in support of incident response activities

- **Preparedness Exercises**
    - Preparedness Exercises
        - Activities designed to evaluate and improve an organization’s readiness to handle security incidents through scenario-based practices and tests
    - Tabletop Exercises
        - Definition
            - Discussion-based sessions where team members review their roles and responses to a hypothetical incident
        - Application
            - No interaction with live systems, enabling a low-stakes environment for planning and coordination
            - Facilitates understanding of the incident response plan and identifies gaps or misalignments
        - Analogy
            - Like a fire drill assembly where everyone discusses their safety routes without leaving the building
        - Example
            - A data breach scenario where team members outline steps to contain and remediate the breach

- Walkthrough Exercises
  - Definition
    - Step-by-step reviews of specific response procedures with team members actively practicing their roles
  - Application
    - Focuses on rehearsing specific actions, such as verifying communication channels and testing access points
    - Conducted in a controlled environment, not interacting with live systems
  - Analogy
    - Like a practice run for an evacuation drill where participants check their routes and exits without leaving
  - Example
    - Rehearsing system shutdown protocols or testing backup systems to validate readiness
- Parallel Exercises
  - Definition
    - Simulates incidents in a staging environment that mirrors the production network, without affecting live operations
  - Application
    - Allows hands-on practice in a realistic setup while maintaining normal business activities
    - Tests detection, containment, and remediation processes in a safe, parallel space
  - Analogy

        - Like a backup rehearsal space identical to the main stage, allowing realistic practice without disrupting performances
      - Example
        - Responding to a simulated malware outbreak in a parallel environment while production systems remain unaffected
    - Simulation Exercises
      - Definition
        - High-intensity, realistic scenarios conducted in the live production environment under controlled conditions
      - Application
        - Tests the team’s response skills in real-time, mimicking actual incidents on live systems
        - Often involves "Red Team" attackers and "Blue Team" defenders in a dynamic setup
      - Analogy
        - Like a full-scale emergency drill where alarms are sounded, and actions are taken as if the threat were real
      - Example
        - A Red Team simulates ransomware attacks, and the Blue Team works to identify and neutralize the threat
    - Summary
      - Preparedness Exercises: Activities designed to strengthen incident response and organizational readiness.
      - Types
        - Tabletop Exercises
          - Discussion-based reviews for planning and coordination

                - Walkthrough Exercises
                    - Hands-on practice of specific steps in a controlled setting
                - Parallel Exercises
                    - Simulations in a staging environment to test realistic responses without impacting live systems
                - Simulation Exercises
                    - Realistic drills in the live production environment for comprehensive testing
            - Recommendations
                - Conduct walkthroughs quarterly, tabletop exercises twice a year, and simulations annually or rotate simulations with parallel tests

- **Immediate Response**
    - Immediate Response
        - The rapid actions taken to contain and mitigate a security incident as soon as it is detected, focusing on crisis management and threat response
    - Crisis Management
        - Definition
            - The process of managing significant security events by coordinating communication with stakeholders to maintain trust and transparency
        - Focus
            - Handling events that impact operations or reputation
            - Ensuring smooth recovery and preventing loss of credibility
        - Application
            - Transparent communication with internal and external parties

        - Use of out-of-band communication methods (e.g., encrypted apps or dedicated lines) to secure sensitive discussions
    - Example
        - During a DDoS attack, the crisis management team
            - Notifies key leaders like the CIO or CISO to assess the impact
            - Prepares an external statement for customers, explaining the disruption and outlining steps being taken
            - Uses out-of-band channels to coordinate internally and safeguard information
- Threat Response
    - Definition
        - A structured approach to quickly identifying, containing, and mitigating specific security threats to protect assets and restore normal operations
    - Framework (NIST SP 800-61)
        - Preparation
            - Implement monitoring tools and train staff to recognize incidents
        - Detection and Analysis
            - Identify suspicious activities and assess severity
            - Example
                - Detecting a phishing attempt and analyzing its scope (e.g., compromised accounts)
        - Containment, Eradication, and Recovery
            - Containment

            - Prevent further damage (e.g., isolating affected systems)
         - Eradication
            - Remove the threat (e.g., deleting malicious files)
         - Recovery
            - Restore systems with security patches or backups
      - Post-Incident Activity
         - Conduct a root-cause analysis to determine vulnerabilities
         - Document lessons learned and update response protocols
   - Application
      - Tools like threat intelligence platforms and monitoring systems support real-time detection and mitigation
      - Cybersecurity Incident Response Teams (CSIRTs) lead these efforts
- Summary
   - Immediate Response: Rapid actions to address security incidents, focusing on
      - Crisis Management
         - Ensures trust and transparency with stakeholders
         - Secures sensitive communication channels and facilitates smooth recovery
      - Threat Response
         - Uses a structured process (e.g., NIST SP 800-61) to identify, contain, and resolve security threats
         - Involves preparation, detection, containment, eradication, and post-incident activities
   - Benefits

                - Mitigates risk and minimizes incident impact
                - Strengthens organizational resilience and preparedness

- **Event Response**
    - Event Response
        - Coordinated actions taken to handle, investigate, and mitigate a security incident after detection and containment, focusing on timeline reconstruction, data recovery and extraction, and breach response
    - Timeline Reconstruction
        - Definition
            - Gathering data from logs, system alerts, and monitoring tools to create a chronological account of events, identifying the attack path and exploited vulnerabilities
        - Tools
            - SIEM Systems (e.g., Splunk, LogRhythm)
                - Centralize log data and automate comparisons to detect inconsistencies
            - Forensic Analysis Tools (e.g., Autopsy, EnCase)
                - Provide deeper device-level insights
        - Processes
            - Analyze logs for unusual patterns (e.g., login anomalies, administrative privilege misuse)
            - Detect manipulation attempts, such as log gaps or false timestamps, by cross-referencing multiple data sources
        - Example

            - An investigator identifies that a critical server access attempt appears in firewall logs but is missing from application logs, suggesting log tampering
            - Using Splunk, they correlate network and endpoint logs to uncover gaps and reconstruct the attack timeline
    - Data Recovery and Extraction
        - Definition
            - Restoring compromised or lost data and extracting critical information for analysis to minimize operational impact and improve forensic accuracy
        - Tools
            - Data Recovery Tools (e.g., Disk Drill, Stellar Data Recovery)
                - Recover data from damaged or encrypted drives
            - Mobile Device Tools (e.g., Cellebrite)
                - Extract data from mobile devices
        - Processes
            - Secure backups and use recovery tools to restore critical data
            - Extract compromised credentials or exfiltrated files for further analysis
        - Example
            - After a breach of a financial server, the team uses backup recovery to restore lost files
            - Data extraction reveals attackers accessed sensitive financial records two days prior, highlighting monitoring deficiencies
    - Breach Response
        - Definition

        - Steps taken to mitigate damage, secure systems, and meet legal obligations following a security breach
    - Frameworks
        - NIST SP 800-61
            - Guides on incident handling and breach mitigation
        - ISO/IEC 27035
            - Standards for managing information security incidents
    - Tools
        - Endpoint Detection and Response (EDR) (e.g., CrowdStrike, Carbon Black)
            - Identifies and isolates compromised endpoints
        - Vulnerability Scanners (e.g., Nessus)
            - Assess and mitigate system weaknesses
    - Processes
        - Contain and remove traces of attackers from the network
        - Notify stakeholders and regulators per legal requirements (e.g., GDPR, CCPA)
        - Implement stricter access controls and update incident response plans
    - Example
        - After a healthcare data breach, the security team isolates affected systems, resets compromised accounts, and reports the incident to regulatory bodies
        - They notify impacted patients and enhance access control measures to prevent future breaches
- Summary

        - Event Response
            - Focuses on managing security incidents after detection and containment, addressing
        - Timeline Reconstruction
            - Tracks the sequence of events to identify vulnerabilities and attacker movements
        - Data Recovery and Extraction
            - Restores lost data and extracts key information for analysis to minimize disruption
        - Breach Response
            - Mitigates damage, secures systems, and ensures compliance with regulatory obligations
        - Benefits
            - Ensures a comprehensive understanding of the incident
            - Minimizes operational and reputational impact
            - Enhances security resilience and compliance

- **Attribution**
    - Attribution
        - The process of identifying the source or actor responsible for a security incident, focusing on gathering evidence to confirm the origin and intent behind the breach or compromise
    - Insider Threat
        - Definition

            - Occurs when an individual within the organization, such as an employee, contractor, or trusted partner, compromises security either intentionally or unintentionally
        - Types
            - Intentional
                - Deliberate actions like data theft or sabotage
            - Unintentional
                - Negligence or accidental exposure of sensitive information
        - Purpose of Attribution
            - Determines whether the threat is intentional or accidental
            - Identifies gaps in security policies or controls
    - Evidence Gathering
        - Key Evidence Sources
            - Access Logs
                - Reveal patterns of suspicious activity, such as
                    - Accessing unauthorized files or systems
                    - Unusual login times or locations
            - Behavioral Analytics
                - Tools like Splunk UBA or Exabeam flag anomalies by comparing actions against user baselines
                - Examples of flagged activities
                    - Accessing sensitive files outside normal hours
                    - Copying unusually large amounts of data
        - Physical Security
            - Cameras
                - Monitor movement in restricted areas

        - Badge Systems
            - Track entries to secure locations, correlating with digital access logs
- Offboarding Process
    - Importance
        - Ensures former employees cannot access sensitive systems or data post-employment
    - Key Steps
        - Deactivating Credentials
            - Immediate removal of access to systems like VPNs, cloud services, and emails
        - Revoking Permissions
            - Disable multifactor authentication tokens
            - Remove remote desktop access
            - Revoke physical access (e.g., building badges)
        - Example
            - An employee leaves but retains VPN access, posing a risk if permissions aren’t revoked
- Response to Insider Threats
    - Actions
        - Restrict access immediately
        - Conduct internal interviews and investigations
        - Take legal action if malicious intent is confirmed
    - Long-Term Measures
        - Strengthen monitoring systems
        - Update security policies and offboarding processes

                - Train employees on security awareness to reduce negligence
    - Summary
        - Attribution
            - Identifies the source of a security incident, particularly insider threats
        - Insider Threat
            - Involves internal actors, either intentionally or accidentally compromising systems
            - Requires analysis of access logs, behavioral patterns, and physical security measures
        - Offboarding
            - Key to preventing post-employment insider threats by revoking all access points promptly
        - Response
            - Involves swift containment, investigation, and mitigation to reduce further risks

- **Root Cause Analysis**
    - Root Cause Analysis (RCA)
        - A systematic process for identifying the underlying reason for a security incident to prevent recurrence
        - RCA traces issues back to their origins, analyzing technical failures, vulnerabilities, or human errors
    - Purpose of RCA
        - Objective

        - Goes beyond addressing symptoms to uncover and address the fundamental causes of incidents
    - Goal
        - Implement long-term solutions to prevent similar incidents in the future
    - Focus
        - Not about assigning blame but improving organizational security and resilience
- Iterative “Why” Technique
    - Methodology
        - Continuously ask “why” for each identified issue
        - Trace the sequence of events until no further questions can be asked
    - Example
        - Incident
            - Unauthorized access occurred
        - Why
            - Weak password was exploited
        - Why
            - No multi-factor authentication (MFA) was required
        - Why
            - Security policies didn’t mandate MFA for certain systems
        - Root Cause
            - Outdated security policy and lack of enforcement
- Feedback Loop
    - Integration into Planning

            - RCA findings are used to enhance security policies, procedures, and training programs
            - Updates ensure that lessons learned are applied proactively to prevent similar incidents
        - Example
            - RCA reveals insufficient phishing training
            - Organization implements recurring phishing awareness programs and updates email filtering systems
    - Practical Example
        - Scenario
            - Successful phishing attack leads to data compromise
        - Initial Perception
            - User error—employee clicked a malicious link
        - RCA Process
            - Why
                - User didn’t recognize phishing email
            - Why
                - Lack of training on phishing recognition
            - Why
                - Training frequency and scope were inadequate
            - Root Cause
                - Inadequate training programs and insufficient email filtering
        - Solutions
            - Enhance training on phishing awareness
            - Strengthen email filtering to block suspicious emails

            - Regularly review and communicate security policies
    - Summary
        - Root Cause Analysis (RCA)
            - Identifies the underlying cause of security incidents to prevent recurrence
        - Process
            - Ask “why” iteratively to trace issues back to their origins
            - Focuses on technical, procedural, or human errors
        - Outcome
            - Findings inform security policies, training programs, and procedural updates
            - Prevents recurrence through proactive, organization-wide improvements
        - Mindset
            - Emphasizes improvement over assigning blame

Conclusion

- Course Overview
    - Covered all four domains of the CompTIA SecurityX Certification exam
    - The course was structured to make learning easier, not necessarily in the order listed in the exam objectives
    - Each video in the course is labeled with the relevant objective number for easy reference and review
- Domain 1: Governance, Risk, and Compliance (20% of the exam)
    - Managing organizational risk, ensuring compliance with regulations, and establishing governance practices
    - Major Topics:
        - Governance, Risk, and Compliance (GRC) strategies
        - Threat modeling
        - Artificial intelligence challenges in enterprise environments
- Domain 2: Security Architecture (27% of the exam)
    - Designing resilient systems and secure infrastructures
    - Major Topics:
        - Design resiliency
        - Software and hardware assurance
        - Secure architecture design
        - Enterprise cloud implementation
        - Zero trust architecture
- Domain 3: Security Engineering (31% of the exam)
    - Designing, troubleshooting, and securing complex systems

        - Major Topics:
            - Advanced identity and access management
            - Endpoint protection
            - Network infrastructure security
            - Specialized and legacy systems
            - Automation and cryptographic techniques
    - Domain 4: Security Operations (22% of the exam)
        - Continuous monitoring, analysis, and response to security events
        - Major Topics:
            - Security Information and Event Management (SIEM)
            - Behavior analytics
            - Vulnerability analysis
            - Threat intelligence
            - Incident response, malware analysis, reverse engineering, and threat hunting