Certified Information Systems Security Professional (CISSP) Certification Training Course

CISSP® is a registered trademark of (ISC)² ®

Domain 08: Software Development Security

Learning Objectives

By the end of this lesson, you will be able to:

- Incorporate various software development methods to evaluate their impact on project workflows
- Evaluate source code analysis tools to assess their effectiveness in identifying vulnerabilities
- Implement software security and assurance strategies to protect systems and ensure operational security
- Assess software security measures to validate threat defenses
- Integrate web security concepts to enhance defenses and ensure secure interactions

Software Development Life Cycle (SDLC)

Importance of Software Development Security

It is important to have secure software development practices to protect systems from potential risks and ensure smooth operations.

It often prioritizes functionality over security, with security controls treated as an afterthought.

It should be integrated into the product's core, protecting all layers rather than just the front-end or wrapper around the functionality to achieve optimal security.

Software Development Life Cycle (SDLC)

It is a structured process that outlines the steps involved in creating, deploying, and maintaining software.

- It is a framework used by development teams within software organizations to plan, develop, maintain, and replace specific software.
- It often includes a **gate process** approach, which requires a formal review at the end of each phase before the next phase can begin.
- If the current phase is completed successfully and all requirements are met, the project can proceed to the next phase.

Software Development Life Cycle (SDLC)

Phase 1
Feasibility study

Phase 2
Requirements definitions

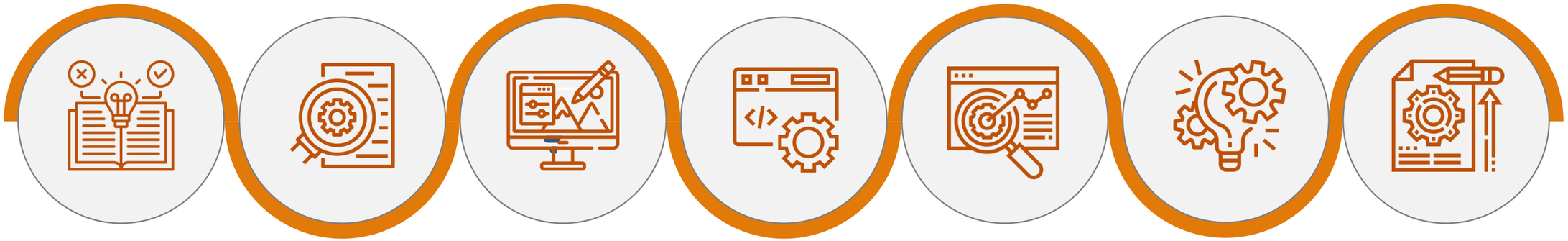
Phase 3
Design

Phase 4
Development

Phase 5
Debugging and testing

Phase 6
Implementation

Phase 7
Post implementation



Pre-SDLC Events

An instantiation of the SDLC is created when management determines that a new software application is necessary or significant changes are required for an existing application.

This decision is made in response to an event, which may include any of the following:

- Changes in market conditions
- Changes in costs or expenses
- Changes in regulation
- Changes in risk
- Changes in customer requirements

Feasibility Study

It is a crucial initial step in software development that evaluates the practicality and viability of a proposed project across technical, economic, legal, operational, and scheduling aspects.

The following are its goals:

- Determines if the project is achievable within the given constraints
- Identifies potential risks and challenges
- Estimates the project's costs and benefits
- Provides a foundation for decision-making

Factors to Consider in a Feasibility Study

- Estimated time needed to develop or acquire the software
- Difference of costs between custom development and buying
- Ability of the existing system to meet current business needs
- Extent to which the application will support the organization’s strategic business objectives
- Feasibility of developing a solution that works with other IT systems
- Expenses involved in creating interfaces between the new and existing systems
- Potential effects of proposed changes on regulatory compliance
- Capability of the system to accommodate future business requirements
- The likelihood that an innovative change will enhance market share

Requirement Definition

It is a critical phase in the SDLC that involves understanding, documenting, and managing the needs of stakeholders. It's the foundation for the entire software development process.

The types of requirements in software projects are:

- Business functional requirements
- Technical requirements and standards
- Security and regulatory requirements
- Disaster recovery and business continuity requirements
- Privacy requirements

Design Phase

It is a crucial stage in the SDLC where the software's blueprint is created.

A high-level design should already be in place to estimate costs and assess financial viability; if not, it should be developed first.

The design process follows a top-down approach, starting with major components and then breaking them down into details.

Design Phase: Design Types

Architectural design

Defines the overall structure of the software system, including components, modules, and their interactions

Interface design

Defines how users will interact with the software, including user interface (UI) and user experience (UX) elements

Data design

Defines the structure and organization of data, including databases and data storage

Component design

Designs individual software components and their functionalities

Algorithm and data structure design

Selects appropriate algorithms and data structures for efficient processing

Design Freeze

It is a critical milestone in the SDLC that involves making only insignificant changes to the product's design to maintain stability and minimize the risk of defects or inconsistencies.

Further changes after the design freeze signal the completion of the design phase, requiring proper change management to enforce restrictions.

This prevents scope creep, where project requirements extend beyond the original scope, leading to delays, increased costs, and reduced project quality.

Development Phase

In this phase of SDLC, developers transform detailed design specifications into functioning software, using the design document as their guide.

The following are the activities involved:

- Selecting the programming language
- Coding the application
- Deciding between an interpreted or compiled language
- Choosing the CASE tool
- Selecting the IDE

Programming Languages Generation

1GL

- It uses machine code.
- Machine code is a binary series of ones and zeros that the machine would read and run directly.

2GL

- It is a low-level computer programming language.
- Mnemonic labels instead of binary machine instructions make them more accessible for the programmer.

3GL

- It is the general-purpose programming language that developers use.
- Interpreters and compilers are used to translate 3GLs into machine code.

4GL

- It is a subset of domain-specific languages.
- These computer languages are specialized to a particular domain.

5 GL

- It is designed to solve problems independent of the programmer providing the logic.

Computer-Aided Software Engineering (CASE) Tools

These are a set of software applications designed to automate activities throughout the software development life cycle (SDLC).

- They are utilized by software project managers, analysts, and engineers to develop software systems more efficiently.
- They accelerate project development to achieve the desired results and help uncover flaws before progressing to the next stage of software development.

Computer-Aided Software Engineering (CASE) Tools

The following types of CASE tools help simplify the software development life cycle:

01 Analysis tools

02 Design tools

03 Documentation tools

04 Project management tools

05 Database management tools

CASE Tools: Categories

The following are its different categories, each supporting specific stages of the software development life cycle (SDLC):

Upper CASE

This includes activities ranging from requirements gathering to the development of data models, data flow diagrams (DFDs), and interfaces.

Middle CASE

This involves the development of detailed designs, including screen layouts, report definitions, data design, and data flows.

Lower CASE

This involves the creation of program source code and data schemas.

Compiler vs. Interpreter

Basis of difference	Compiler	Interpreter
Compiling a program	A program is compiled once using a compiler.	An interpreted code is compiled each time the program is run.
High-level instructions	It translates high-level instructions directly into machine language.	It translates high-level instructions into an intermediate form.
Speed	Compiled programs run faster.	Interpreted programs run slower than compiled programs.
Search for errors	It searches for all the errors of a program and lists them.	It checks a program statement by statement for errors.
Error list generation	It generates the error message only after scanning the whole program.	It continues translating the program until the first error is met, in which case it stops.
Debugging	It is comparatively hard to debug.	It is easy to debug.
Use	It is difficult to use.	It is easier to use.
Examples	C, C++	Python, Ruby

Software Library

It is a set of precompiled helper functions, objects, or modules that are intended to be reused during software development.

Components

- Standard libraries provided by most languages such as Python, C, C++, C#, Java, and Ruby
- Open-source libraries, which are free to reuse, modify, and publish without any permission
- Third-party libraries
- Custom libraries

Best implementation practices

- Use libraries only from trusted sources that are actively maintained and widely used by several applications.
- Create and maintain an inventory catalog of all the third-party libraries.
- Proactively keep libraries and components up to date.

Tools and Toolsets

A software development tool is a program that software developers use to create, debug, maintain, or otherwise support other programs and applications.

Toolsets refer to a collection of tools often related and used together to achieve broader development tasks.

Tools and Toolsets

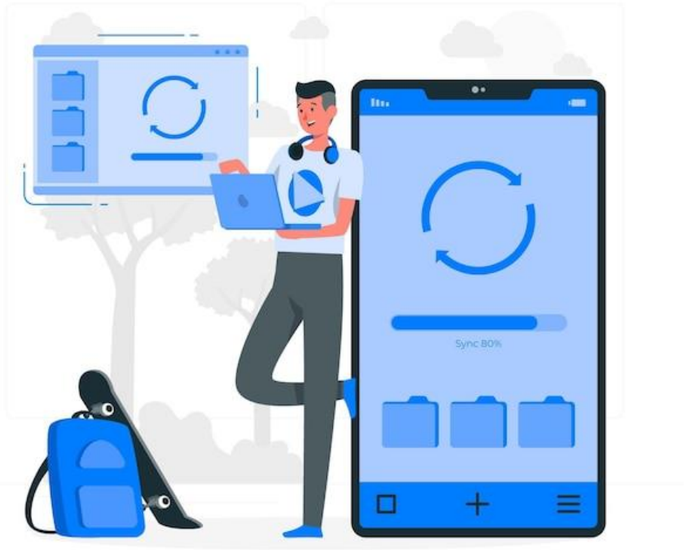
They help improve the developer's productivity.

List of tools:

- Source code editors
- Code review tools
- Compilation and linking tools
- Memory debuggers
- Performance analysis tools
- Unit testing tools
- GUI interface generators

Runtime System

It refers to the collection of hardware and software resources needed for program execution on a computer system.



- The low-level services provided by a runtime system include processor interfacing, memory loading, and digital-to-binary conversion.
- The higher-level services include type checking, code generation, debugging, or code optimization.

Example

Java runtime environment (JRE) provides the complete framework for executing and managing Java programs.

Integrated Development Environment (IDE)

It is a software platform that provides programmers and developers with a comprehensive set of tools for software development in a single product.

Benefits:

- Provides programming capabilities through a text editor or a GUI (Graphical User Interface)
- Offers pre-installed libraries for the specific programming languages
- Supports various features, including compiling, debugging, version control, platform-specific code suggestions, and code deployment

Code Repository

It is a file archive and web hosting facility where many source codes are stored privately or publicly.

Securing a code repository includes:

- Physical security
- System security
- Operational security
- Software communication security
- File systems and backups
- Access control

Example

Open-source projects and other multi-developer projects use a source code repository to handle various versions.

Source Code Management (Version Control)

It is a vital part of the SDLC that tracks and manages changes to source code, enabling efficient collaboration and maintaining code integrity.

Collaboration: Enables multiple developers to work on the same codebase simultaneously without conflicts

Version control: Tracks changes to code, allowing for rollback to previous versions if needed

Code review: Facilitates code inspection and improvement through peer review

Branching and merging: Supports parallel development and integration of code changes

Backup and recovery: Provides a reliable backup of code, preventing data loss

Purpose of Version Control

Control: It employs **check-out** and **check-in** functions to restrict developer access to specific application parts, ensuring code integrity.

Version control: It monitors code versions, allowing developers to check in changes, identify differences, and enable reversion to older versions if issues arise.

Protection: It restricts access to application source code, safeguarding the organization's intellectual property, and preventing unauthorized changes, fraud, or misuse.

Record keeping: It tracks changes to the source code, including check-outs, check-ins, and modifications, allowing management to monitor the alterations and their authors.

Debugging and Testing

In this phase, the errors are identified and resolved, and the software's functionality is verified.

Debugging

- Identifies and remove errors or defects from software code
- Follows iterative steps of analysis, correction, and retesting if necessary

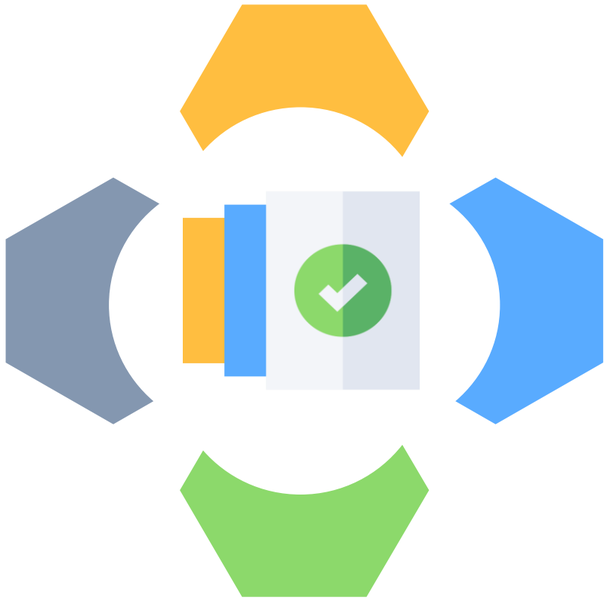
Testing

- Executes the software to evaluate its compliance with specified requirements and identify defects
- Includes levels of testing such as unit testing, integration testing, system testing, and acceptance testing

Debugging: Objectives

Monitor operation
Ensures that software modules correctly manipulate data and perform calculations

Control
Checks input fields and records to prevent errors and tampering which can lead to application abuse and security incidents



Ensure proper resource usage
Tests modules for proper resource utilization to avoid issues like memory leaks

Validate output
Performs output validation to verify that output data is within acceptable bounds, detecting malfunctions in application modules

Implementation

It involves putting the completed application software into the production environment before starting User Acceptance Testing (UAT) and Quality Assurance Testing (QAT).

The following are the activities involved:

- Prepare physical space for on-premises production systems
- Build production systems
- Prepare virtual machines for cloud-based production systems
- Install application software
- Migrate data

UAT and QAT should be performed on the production environment anticipated for use once approvals are obtained.

Types of Implementation

Parallel cutover

The organization runs both old and new applications simultaneously to compare performance and ensure proper functionality.

All-at-once cutover

The organization migrates the entire environment to the new application at once.

Module-by-module cutover

The organization migrates different parts of the application at various times.

Geographic cutover

In large retail networks, individual locations migrate to the new application sequentially instead of all at once.

Training

It is essential for enhancing software development processes, project management, and delivering high-quality products.

- The success of the software development project depends on the knowledge and skills of various individuals within the organization.
- Separate training sessions are held for end users, customers, support staff, and trainers.

Rollback Planning

It is a strategy that offers organizations a safety net during application migrations to new environments, enabling a return to the previous state if significant issues occur.

It is especially recommended for critical applications, even if a rollback is ultimately not needed.

Data Migration

It is a complex process that involves transferring data from one system or database to another.

It requires careful planning and includes the need for audit trails and logs to verify the accuracy and completeness of the data conversion.

The verification may utilize manual processes, system utilities, vendor tools, and specialized applications.

Planning the Migrations

The data migration project should be meticulously planned, and appropriate methodologies and tools must be employed to minimize the risk of:

- Disrupting routine operations
- Violating the security and confidentiality of data
- Creating conflicts and contention between legacy and migrated operations
- Causing data inconsistencies and loss of data integrity during the migration process

Verification of Data Migrations

Record counts

Utilizes programs to compare record counts in old and new tables, confirming the completeness of data migration

Batch totals

Sums numeric data records in both databases to verify the data integrity of key elements during migration

Checksums

Runs checksum programs on old and new databases to ensure the accuracy of the migrated data

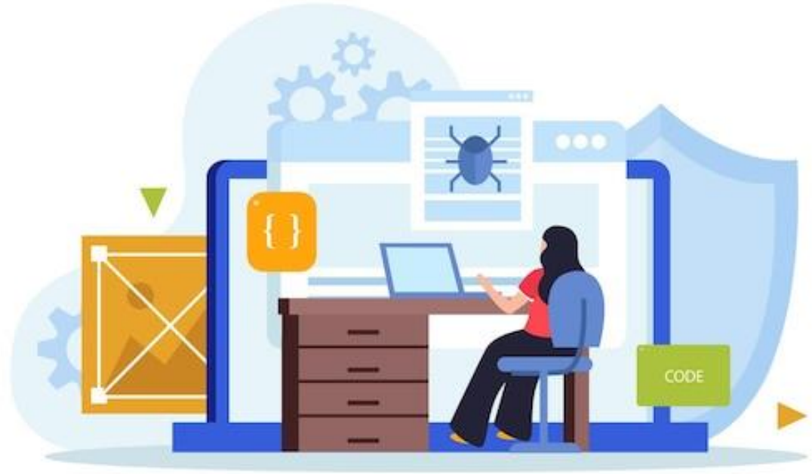
Post Implementation

Once the implementation is done, an information system auditor performs the following:

- Determines if the system’s objectives and requirements were achieved
- Assesses if cost benefits are being measured, analyzed, and accurately reported to the management
- Reviews program change requests to evaluate the type of changes required for the system
- Reviews controls to ensure they operate according to design
- Checks operators’ error logs for any resource or operating problems
- Reviews input and output control balances and reports to verify accurate data processing

Software Development Risks

It refers to any potential event or condition that could harm the success of a software project.



This includes factors that may cause delays, exceed the budget, or result in a substandard end product.

By proactively mitigating these risks, development teams can significantly boost the chances of project success and consistently deliver top-notch, high-quality software.

Software Development Risks

Application inadequacy

Refers to the application not meeting business requirements, leading to its underuse or abandonment

Security and privacy defects

Highlights vulnerabilities within the application that could be exploited for misuse

Project risk

Indicates poor management of the application development or acquisition process, resulting in budget overruns, delays, or project abandonment

Software Development Risks

Business inefficiency

Indicates failure of the application to meet business efficiency expectations, resulting in delays and the need for additional resources to complete critical tasks

Market changes

Highlights that unexpected market changes between project approval and completion can increase costs and reduce profit margins, affecting overall success

Scope creep

Refers to uncontrolled changes or expansions in software requirements that can complicate the project and lead to potential delays or budget overruns

Quick Check

During software development, a programmer writes low-level code in assembly language for a hardware-specific task. Which of the following tools will convert the assembly language into executable machine code for the system to run?

A. Interpreter

B. Compiler

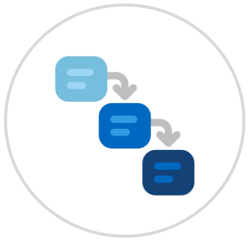
C. Assembler

D. Verifier

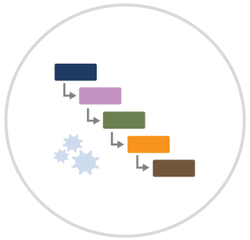
Software Development Models

Software Development Models

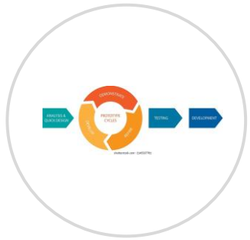
It is a framework that is used to structure, plan, and control the software development process.



Waterfall model



Modern waterfall model



Prototype model

Spiral model

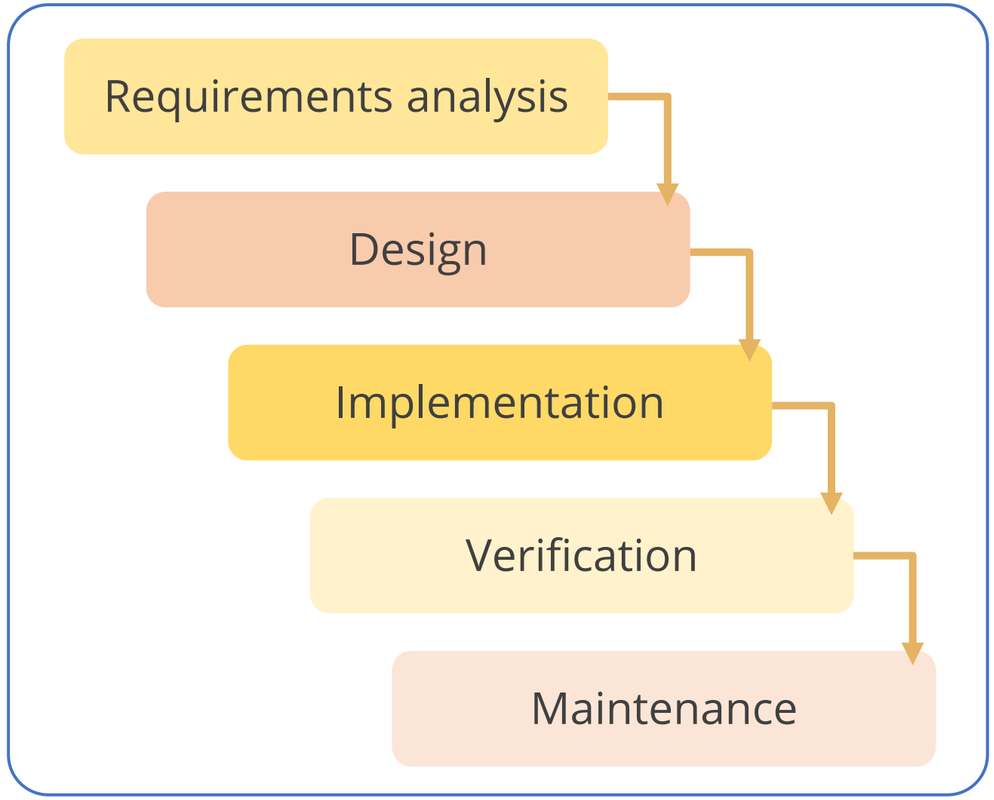
Agile model

To manage a project efficiently, the manager or development team must choose the best-fitting software development model.

All methodologies have different strengths and weaknesses and exist for different reasons.

Waterfall Model

It is a linear application development model that uses the following rigid phases:

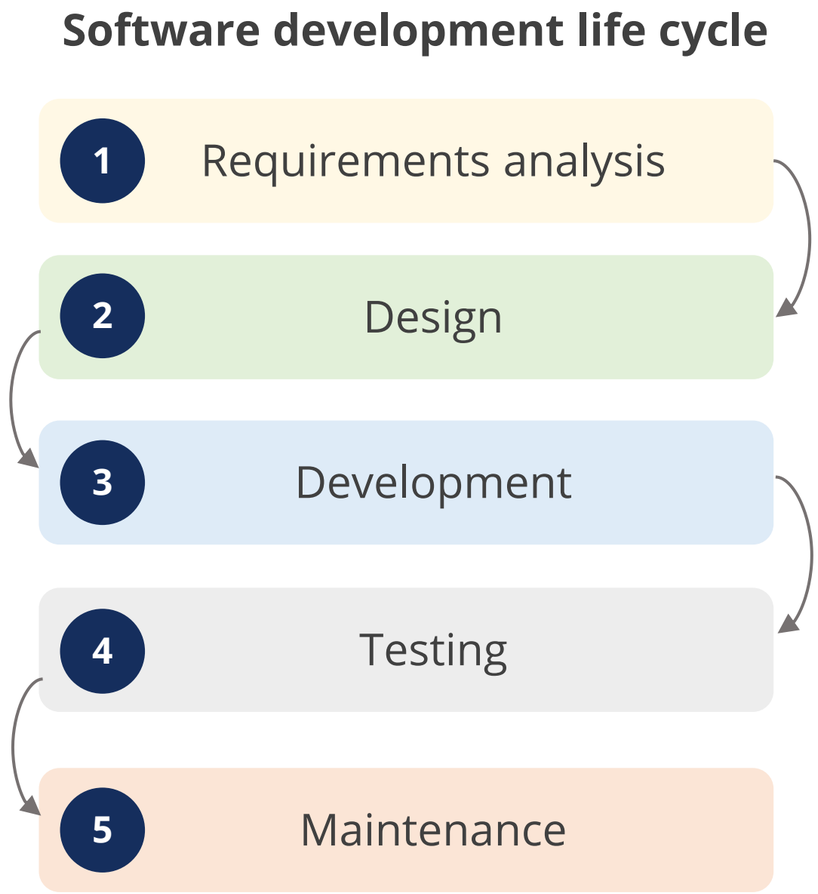


In 1976, Barry Boehm reinterpreted the waterfall model.

Modern Waterfall Model

It allows development to return to the previous phase to correct defects discovered during the subsequent phase.

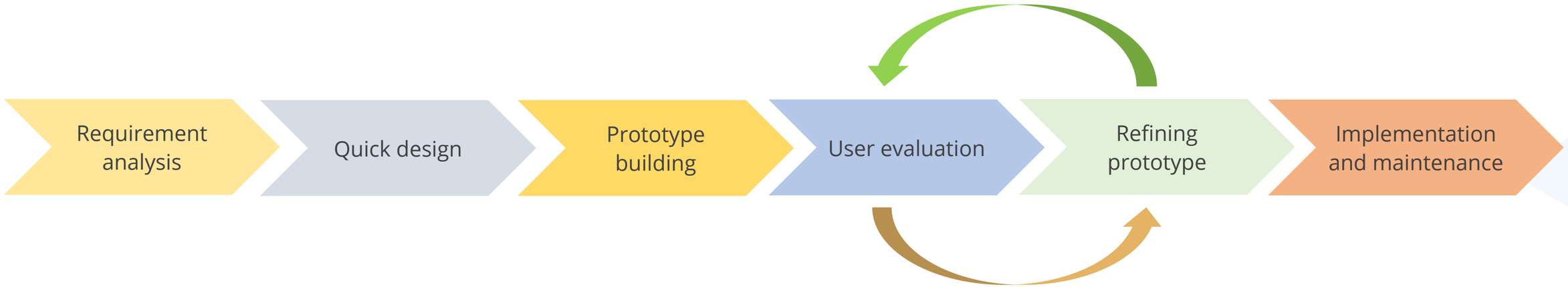
This feature of returning to the previous phase is often known as the feedback loop characteristic of the waterfall model.



Prototype Model

In this model, a prototype of the end product is first developed, tested, and refined repeatedly based on customer feedback until a final acceptable version is achieved.

- It is used when the customers do not know the exact project requirements beforehand.
- In this approach, a sample product is developed to explore a specific approach to a problem before investing expensive time and resources.



Prototype Model: Types

Rapid prototype

- It is quickly created to test the validity of the current understanding of the project requirement.
- It is a quick and raw method of creating the prototype.
- It is not intended to be built upon but rather discarded after use.

Evolutionary prototype

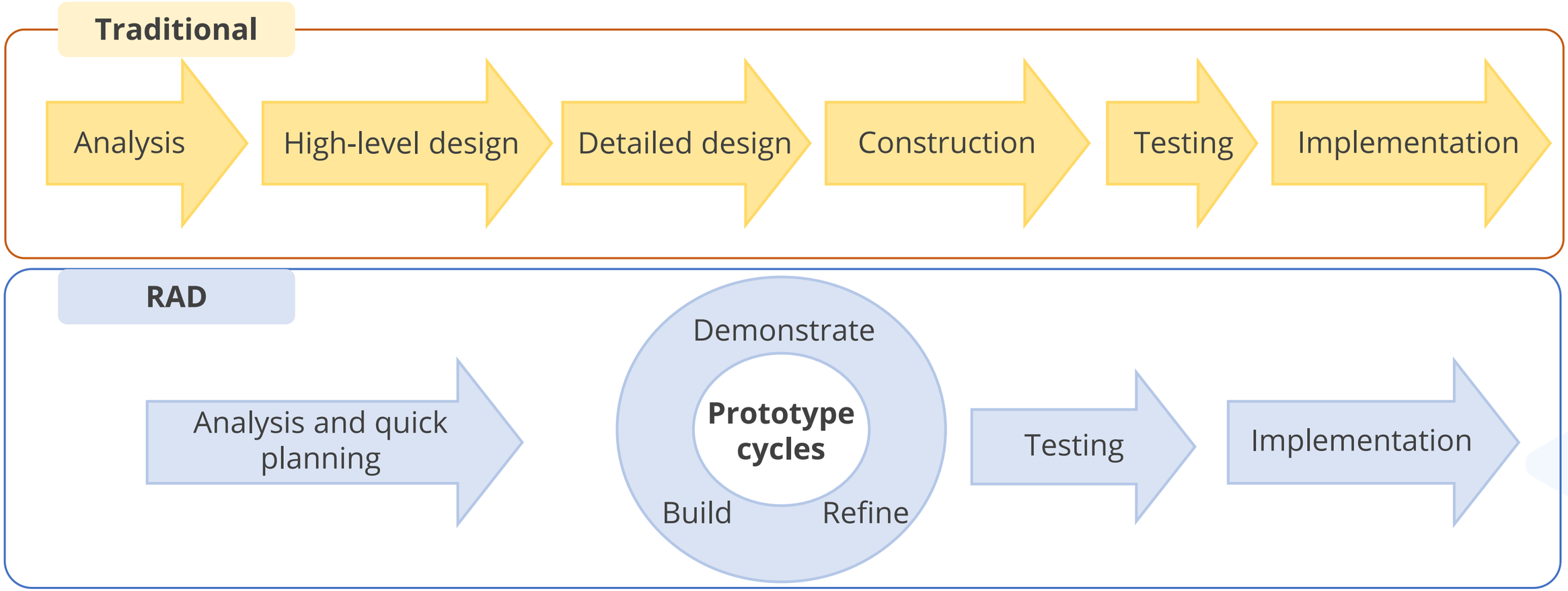
- It is built with the goal of incremental updates.
- It is continuously improved upon until it reaches the final product stage.
- It uses the feedback obtained during each phase to enhance the prototype.

Operational prototype

- It is an extension of the evolutionary prototype.
- It is designed to be implemented in a production environment while still being refined.
- It gathers feedback during this phase and makes changes directly within the working site.

Rapid Application Development Model (RAD)

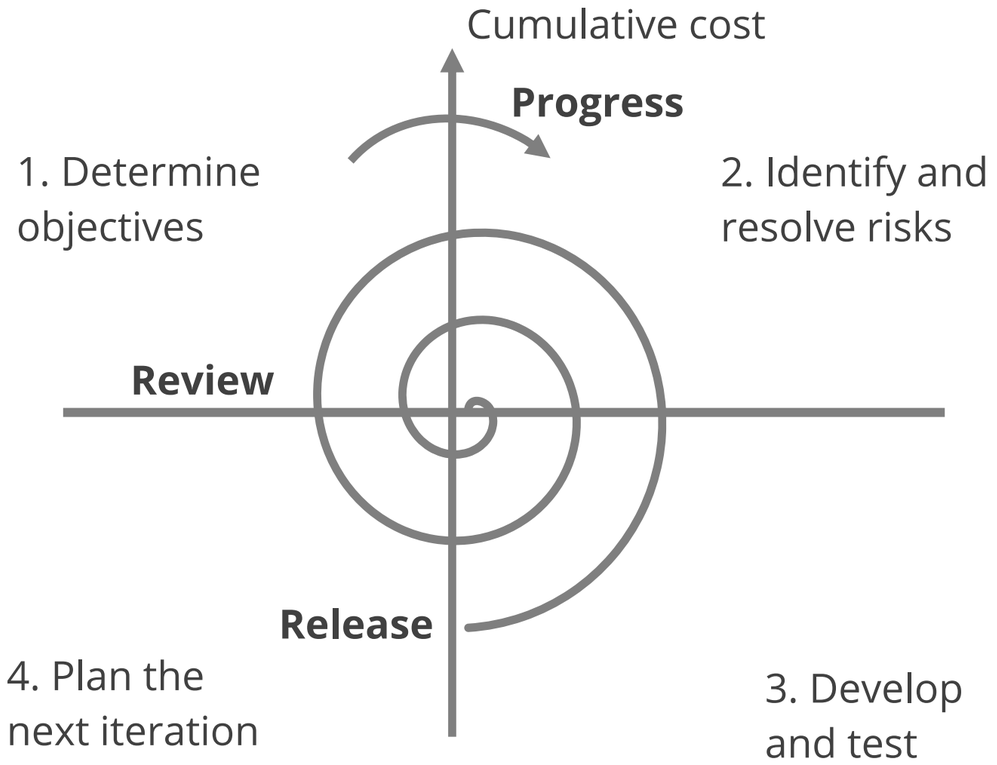
- It is a form of rapid prototyping.
- Software is developed using prototypes, dummy GUIs, and back-end databases.
- The goal is to meet the system's business needs.



Spiral Model

It combines the idea of iterative development with the systematic and controlled aspects of the waterfall model.

- It was developed in 1988 by Barry Boehm.
- It is a meta-model that incorporates several software development models.
- It includes risk management within software development.



Agile Model

It focuses not on rigid, linear, stepwise processes but on incremental and iterative development methods that promote cross-functional teamwork and continuous feedback mechanisms.

It prioritizes:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

This model is considered **lightweight** compared to the traditional methods, which are often considered heavyweight.

Agile Principles

1. Customer satisfaction
2. Changing requirements
3. Frequent delivery
4. Measure of progress
5. Sustainable development
6. Close cooperation
7. Motivated individuals
8. Face-to-face conversations
9. Technical excellence
10. Simplicity
11. Self-organizing team
12. Regular adaptations

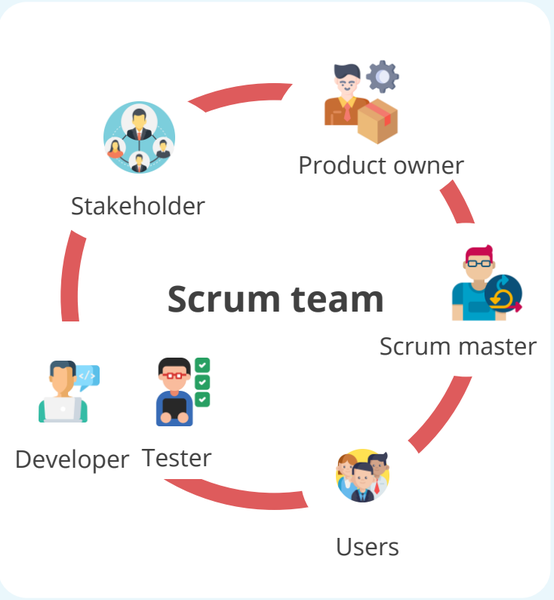
Agile Methodology: Scrum

It is an iterative and incremental process commonly used to manage Agile software development efforts.

- A typical scrum team consists of five to nine members.
- Larger projects are organized into a scrum of scrums that scales upward to include hundreds of programmers.
- A scrum project is divided into sprints. These are focused efforts to deliver specific portions of the overall project.
- Each sprint usually lasts between two to four weeks.

Scrum Roles

- **Scrum master:** Project manager or team leader
- **Product owner:** Customer or the customer's representative who speaks for the customer
- **Team:** Project team members who perform the actual project work
- **Users:** People who will be using the software once it has been developed or updated
- **Stakeholder:** Other parties who contribute to the project in some way, such as customers, vendors, and suppliers



Source: https://www.energy.gov/sites/default/files/cioprod/documents/scmguide.pdf

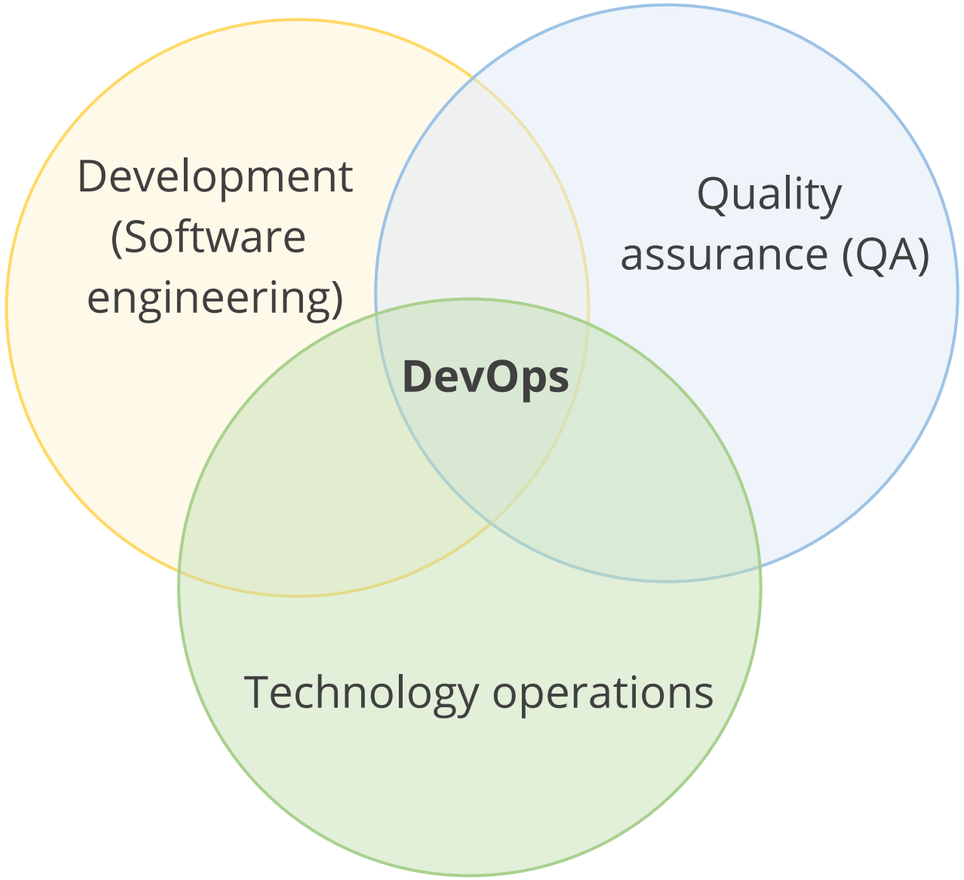
DevOps

It is a software development method that emphasizes communication, collaboration, and integration between the organization’s software developers and IT staff.

Features:

- It helps organizations quickly produce software products and services.
- It ensures the adoption of quality assurance practices to improve the performance of technology operations.

It is derived from the terms, **development** and **operations.**



DevOps: Benefits

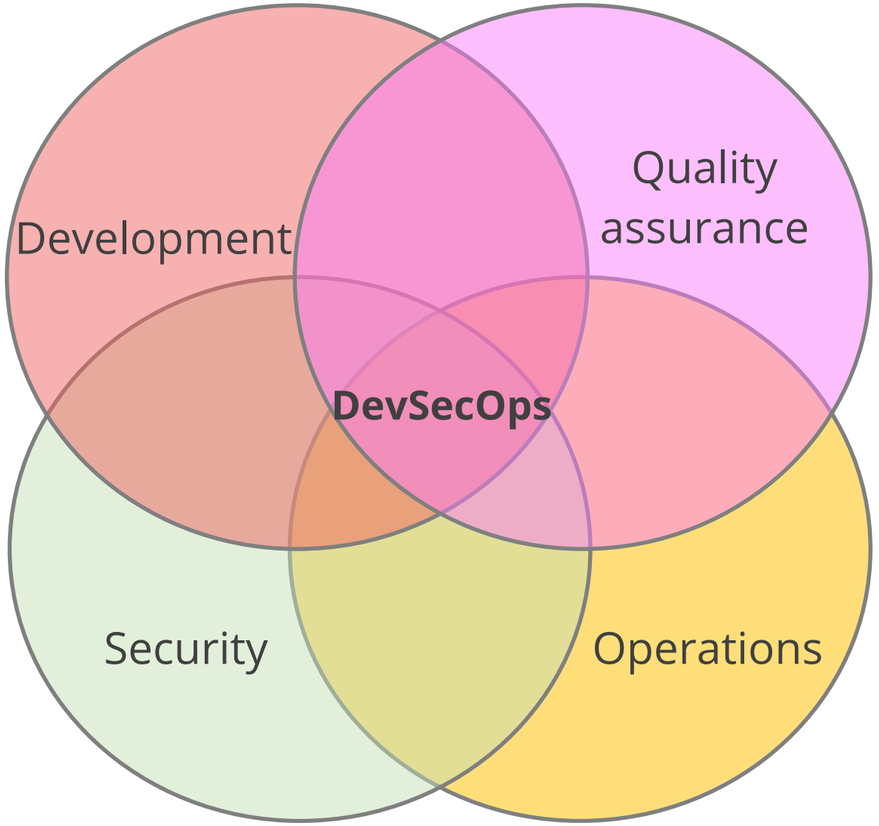
- Improved communication between developers and operations teams
- Continuous integration and continuous deployment
- Enhanced software quality and security
- Rapid improvements through regular feedback
- Faster time to market for software

DevSecOps

It extends the DevOps workflow to include automated security processes and tools.

It follows the secure-by-design principle by:

- Utilizing automated code reviews and automated security testing
- Educating and empowering developers to implement secure design patterns



DevSecOps: Benefits

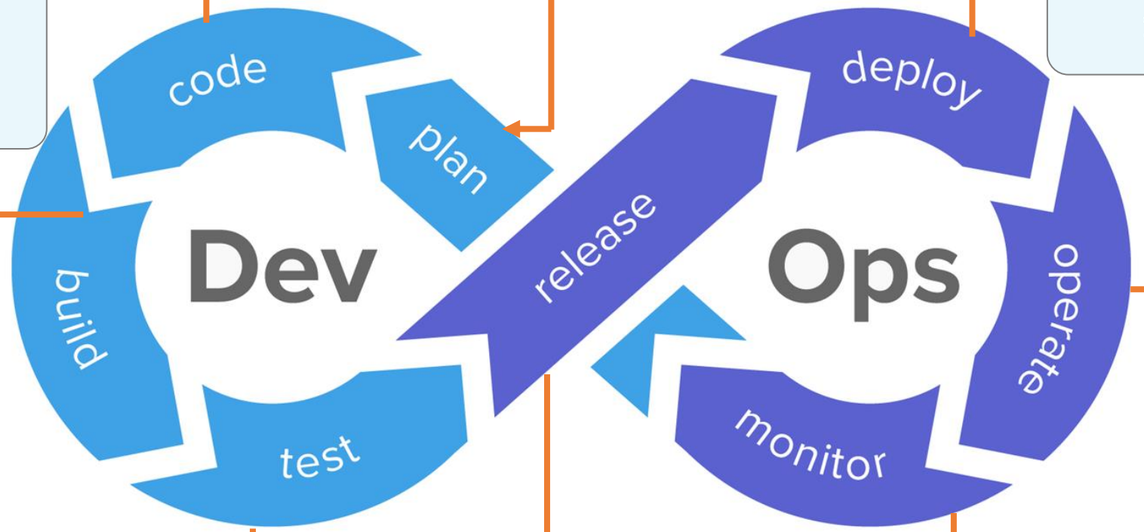
- Identifies vulnerabilities early in the SDLC, reducing costs and increasing deployment speed
- Enhances security by minimizing vulnerabilities and insecure defaults, and increasing automation with immutable infrastructure
- Reduces the mean time to resolve (MTTR) for security incidents through monitoring and efficient remediation practices

DevSecOps: Functioning

- Secure code review
- Static analysis (SAST)
- Third-party component analysis

- Threat model
- Security architecture
- Training

- Configuration management
- Hardening
- Infra-vulnerability scanning



- Access control
- Patch management
- Audit and compliance
- Change control

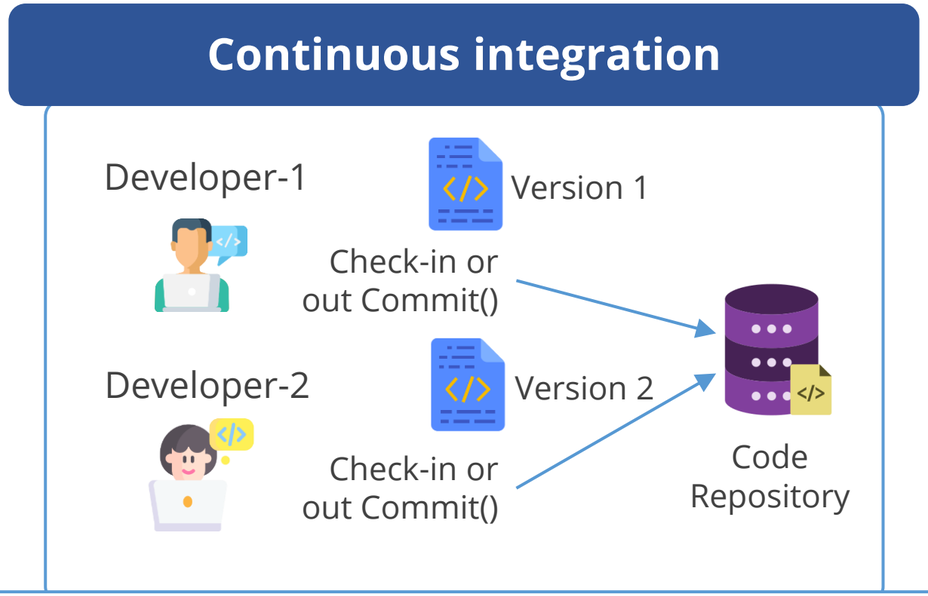
- Penetration testing
- Dynamic analysis (DAST)

- Final security review
- Compliance validation

- Log collection
- Intrusion detection
- Analytics
- SIEM

Continuous Integration and Continuous Delivery (CI/CD)

It is a modern software development practice that allows for frequent code change in an incremental, repeatable, and secure manner.



Application code changes are regularly built,
tested, and merged to a shared repository several times a day.

Continuous Integration and Continuous Delivery (CI/CD)

Continuous delivery

Application code changes are automatically
bug-tested and uploaded to a repository (like GitHub), ready to be
deployed to production by the operations team (manually).

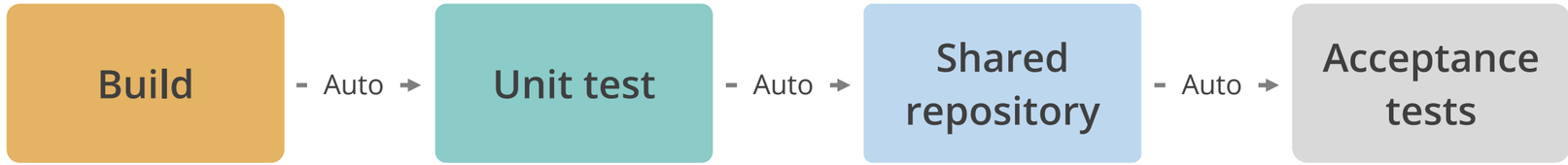
Continuous Integration and Continuous Delivery (CI/CD)

Continuous deployment

It is the step after continuous delivery where application code changes are automatically deployed into production whenever they pass the automated tests.

Continuous Integration, Continuous Delivery, and Continuous Deployment

Continuous integration



Continuous delivery

Continuous deployment

Quick Check

You are managing a project using scrum. A customer requests a major feature change. Which role is responsible for discussing the feature and managing the customer's expectations?

A. Scrum master

B. Product owner

C. Team members

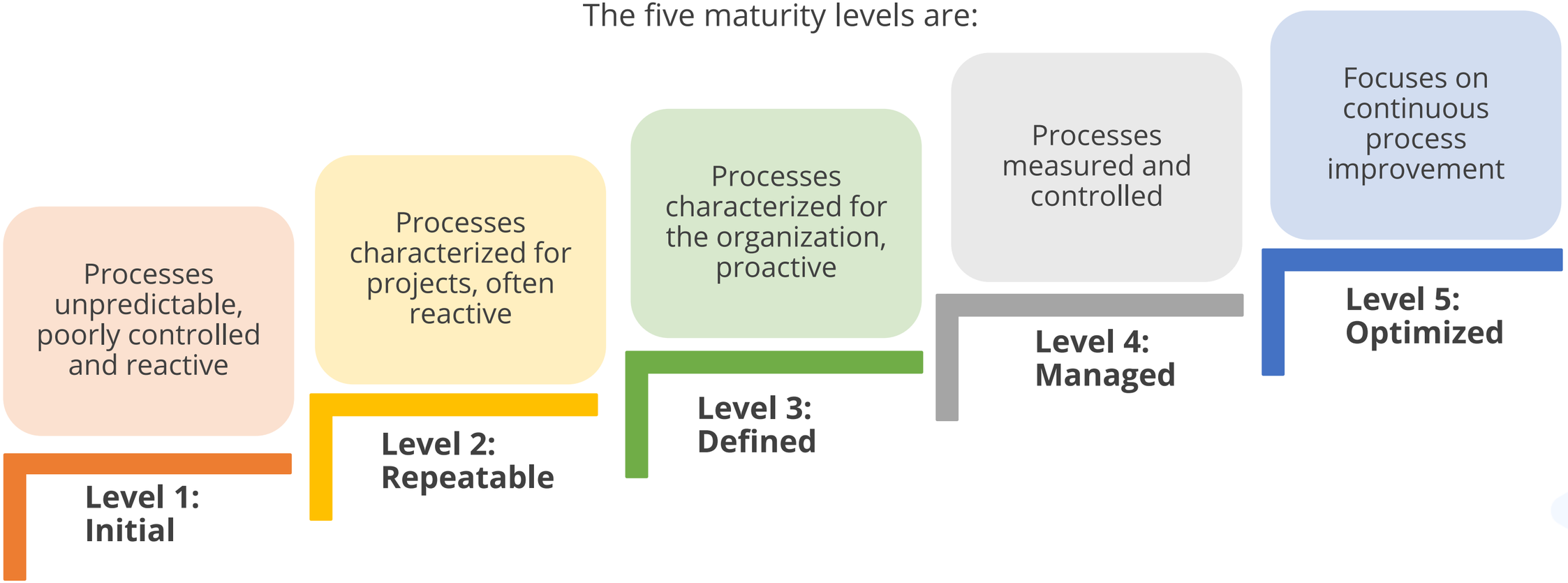
D. Stakeholder

Software Maturity Model

Software Capability Maturity Model (CMM) Levels

They are based on the premise that the quality of a software product is a direct function of the quality of its associated software development and maintenance processes.

The five maturity levels are:



Source: https://en.wikipedia.org/wiki/Capability_Maturity_Model and https://imgur.com/WERTTou

Software Assurance Maturity Model (SAMM)

It is an open framework that provides an effective, measurable way for all types of organizations to analyze and improve their software security posture.

Features

- **Measurable:** Defined maturity levels across business practices
- **Actionable:** Clear pathways for improving maturity levels
- **Versatile:** Technology, process, and organization agnostic

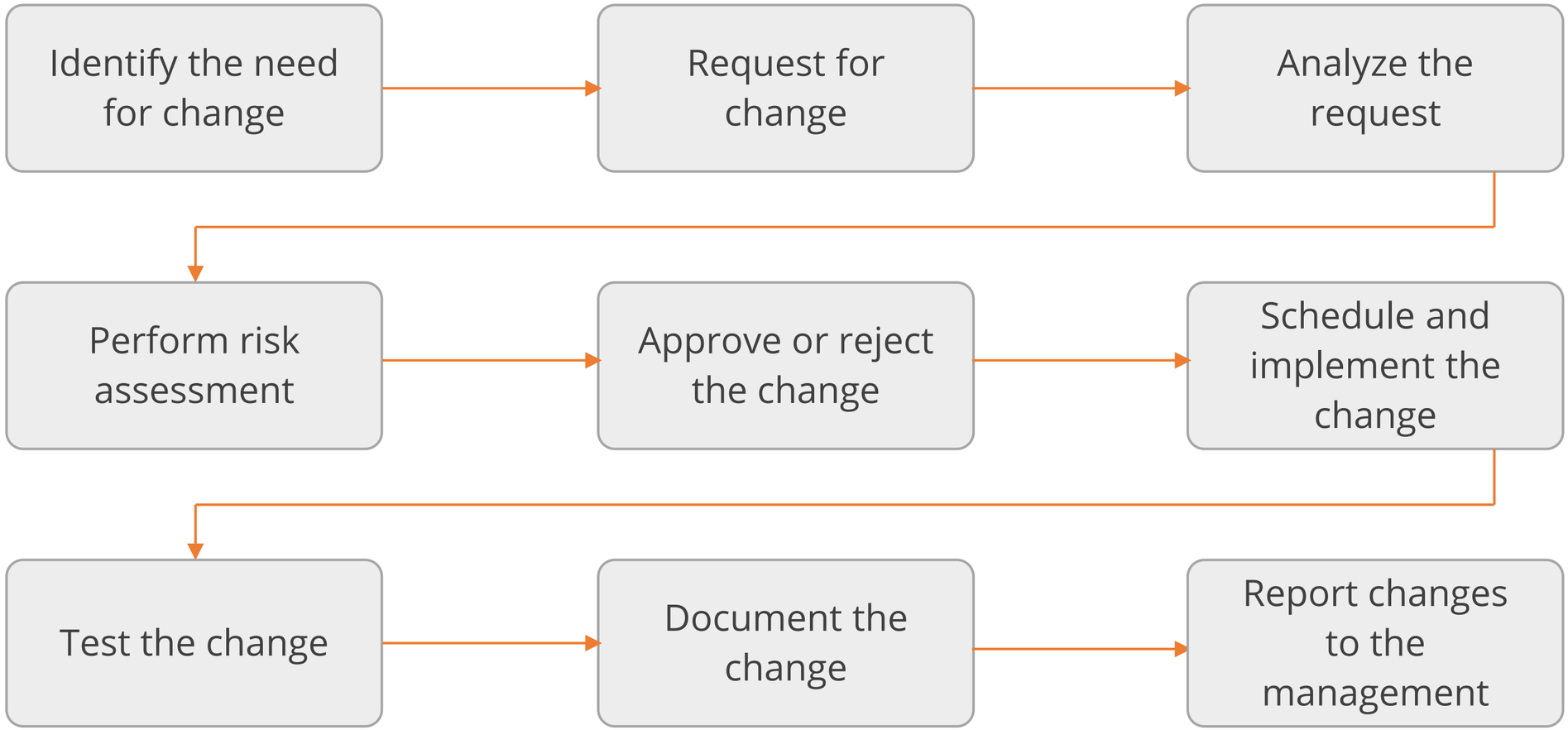
Change Management

It allows companies to manage, monitor, and optimize software changes to ensure that:

- Software change management follows a recognized procedure
- Due diligence is performed to assess the business impact of any software change before deciding whether to proceed with the rollout
- The scope is verified for completion and accuracy both before and after the deployment or change takes place

Change Management

The following flow diagram explains the change management process.



Software Configuration Management (SCM)

It is the process of systematically managing, organizing, and controlling the changes in the documents, codes, and other entities during the SDLC.

Source: https://www.energy.gov/sites/default/files/cioprod/documents/scmguide.pdf

Software Configuration Management (SCM) Processes

- **Identification** determines all the components of a project.
- **Version control** combines procedures and tools to handle different versions of configuration objects that are generated during the software process.
- **Change control** reviews proposed changes to the project and incorporates them into the software configuration if approved, through change control.
- **Configuration audit** confirms that approved changes have been implemented through configuration audits.
- **Reporting** provides accurate data on the current status and configuration.

Source: https://www.energy.gov/sites/default/files/cioprod/documents/scmguide.pdf

Cohesion and Coupling

Cohesion

- It refers to how many different types of tasks a module can carry out.
- High cohesion means a module performs one task or tasks that are similar in nature.
- High cohesion is better.
- Any change in the task can be done without impacting other tasks.

Coupling

- It measures how much interaction one module requires to carry out its tasks.
- Low (loose) coupling is better.
- High (tight) coupling means a module depends upon many other modules.

High cohesion and low coupling are very instrumental in software maintenance and development.

Integrated Product Team (IPT)

It is a multi-disciplinary team that works throughout the SDLC and facilitates decision-making by:

- Working together to build successful programs
- Identifying and resolving issues
- Making comprehensive and timely recommendations

Integrated Product Team (IPT)

This team comprises members from the organization's relevant functional disciplines and serves as:

- **A review and decision-making body**: It is used to review and make decisions on complex programs and projects.
- **A collaborative forum**: It provides a platform for collaboration, engaging all the stakeholders.

Quick Check

A software development team is starting a new project and is discussing when to include security considerations. At what phase should the team first address security to ensure that potential vulnerabilities are identified early on?

A. During requirements gathering

B. During design

C. During integration testing

D. During implementation

Identify and Apply Security Controls in Software Development Ecosystems

Application Controls

Software applications process and transmit information but can only determine valid data with specific programming.

Controls are needed to maintain data integrity at each stage. The following are its types:

Programmed (automated) controls

They are executed automatically using software or systems.

Examples

- Validation and edit checks
- Programmed logic functions and controls

Manual controls

They require human intervention and verification.

Examples

- Review of reconciliation reports
- Review of exception reports

Input Controls: Authorization

It is vital for data security, allowing only authorized individuals to enter data. It complements input validation and requires management authorization for all data entries.

The following are key data security controls:

Workstation identification: Terminals and workstations are identified using an electronic serial number, network address, or certificate and then allowed to input transactions.

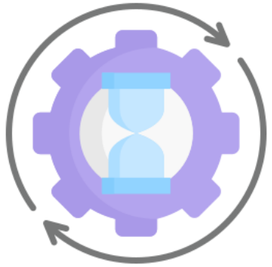
Approved transactions and batches: Transactions and batches are checked and verified by management and authorized personnel before processing.

User access control: Every personnel is verified with a unique ID and password to log in and use applications.

Source documents: In some cases, data can only be input from existing source documents, such as mailed invoices, checks, receipts, or customer-filled forms.

Input Controls: Validation

It ensures accurate, complete, and secure data entry, acting as the first defense against attacks and integrity risks. Its key components include:



Consistency

Compares related data across fields, like verifying ZIP codes against the city's allowed range

Length

Validates input data length, especially for fields like names, to prevent buffer overflow attacks

Input Controls: Validation

Its key components include:

Type check

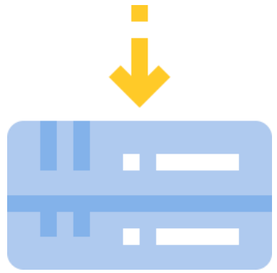
Ensures that input fields accept only the correct data type

Example: Numeric fields accept digits and name fields accept letters.

Range and value check

Validates that input values fall within an acceptable range

Example: The day field should only accept numbers 1-31 and the month field 1-12.



Existence check

Confirms that all required input fields contain data before submission

Processing Controls

These are implemented to ensure the accuracy, completeness, and timeliness of data during processing.

These controls should verify that only authorized data processing occurs and detect any issues that may arise.

They log any overrides and establish a separate responsibility for reviewing these overrides.

Edit controls are applied after data entry but before the processing stage to ensure data integrity.

Processing Control: Checks

Calculation accuracy

Verifies the accuracy of mathematical and logical operations and utilizes pre-defined formulas and algorithms

Data conversion

Ensures accurate conversion between data formats and validates data integrity during conversions

Limit checks

Compares data values against predefined upper and lower limits and prevents data entry error and system failures

Sequence checks

Verifies the order of data processing and identifying missing or duplicate records

Batch totals

Calculates control totals for a group of records and compares control totals before and after processing

Hash totals

Creates a numerical representation of data for comparison purposes and detects data modifications

Processing Controls: File Processing

Software processing controls on files ensure data integrity, security, and accuracy during file operations.

- They are essential to prevent data loss, corruption, unauthorized access, and other potential issues.
- They enable organizations to protect valuable data, maintain integrity, and ensure regulatory compliance.

Processing Controls: File Processing

These controls perform the following:

Ensuring data file security

Configures access controls to ensure that only authorized users or processes are permitted to access data files

Error handling

Verifies erroneous transactions requiring correction or re-input by personnel other than those who originally entered them

Internal and external labeling

Applies proper labeling for both internal and external identification of data files

Processing Controls: File Processing

These controls perform the following:

Verifying the data file version

Verifies the version of a data file independently to ensure that the correct file is being processed

Retaining source files

Retains data input at the beginning of a processing run for a minimum period to facilitate re-running a batch many days or weeks later if needed

Maintaining transaction logs

Keeps log files containing transactions for a minimum period to support troubleshooting or investigations of data errors weeks or months later

Error Handling

It is crucial for detecting, reporting, and recovering from errors during program execution, ensuring system reliability, preventing data loss, and providing informative feedback to users.

It results in more robust and user-friendly software applications.

Error Handling

The following are the error handling procedures in software applications:

Batch rejection
Rejects the entire input batch if the totals do not match the expected values, and assigns data control analysts to review it for issue identification

Transaction rejection
Denies individual input transactions, whether automated or user-entered, using the software application

Hold in suspense
Suspends the entire batch to allow for the correction of errors before it can be rerun

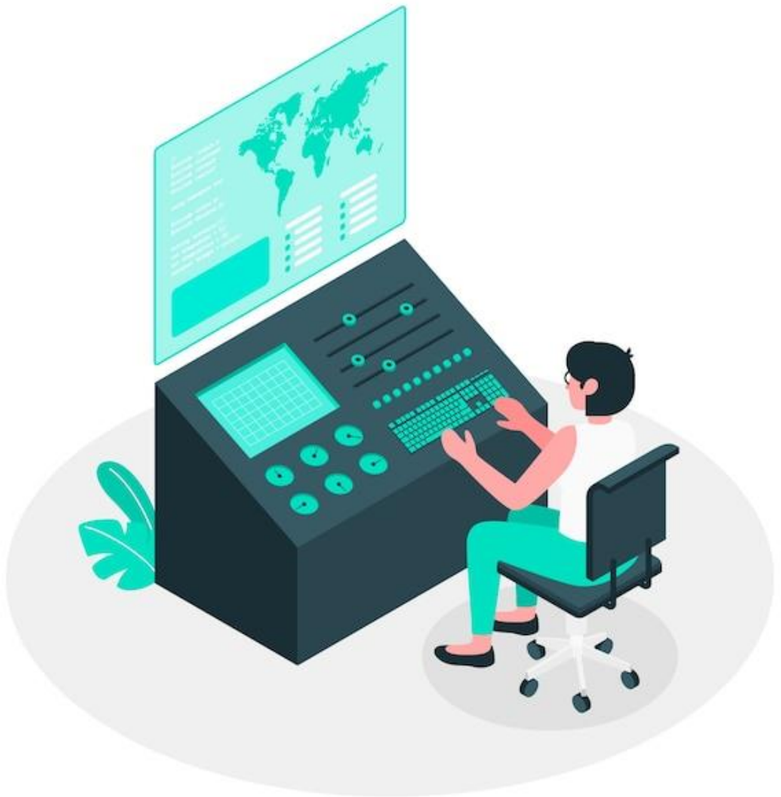
Request re-input
Prompts that the user re-input either the entire form or just the individual field that appears to be incorrect through an interactive user program

Output Controls

They ensure that the data produced by an application is accurate, complete, authorized, and securely delivered.

They act as the final defense in the data processing cycle.

They are crucial for verifying the results for accuracy and validity.



Output Controls: Application

Data file versioning

- Uses specialized physical forms for certain calculation outputs, such as checks, warrants, invoices, and certificates, ensuring they are printed correctly
- Serializes the forms and stores them in a secure, locked cabinet to protect against unauthorized access
- Implements dual custody for high-value situations, necessitating the presence of two individuals for access to sensitive forms

Data security

Prevents unauthorized access, modification, or disclosure by using encryption, digital signatures, and access controls

Quick Check

During a critical software update, your team experiences an unexpected failure, leading to data loss and system crashes. What steps should be taken to handle this error effectively?

A. Ignore the error and continue with the next tasks

B. Report the error immediately, recover from the failure, and provide feedback to users

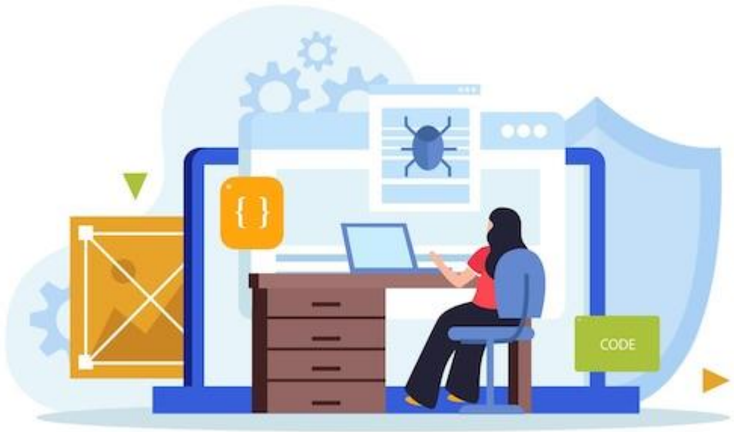
C. Retry the update without investigating the issue

D. Document the error but postpone addressing it

Assess the Effectiveness of Software Security

Secure Software Development: Best Practices

The best practices for secure software development are provided by:



- Open web application security project (OWASP)
- ISO/IEC
- Web application security consortium (WASC)

Software Security and Assurance

The security kernel is responsible for enforcing a security policy.

- It is a strict implementation of a reference monitor mechanism.
- It is a small part of the operating system that all information references and authorization changes must pass through.

Software Security and Assurance

The three basic conditions of the kernel are:

Completeness

Isolation

Verifiability

Software Security and Assurance

The following are its features:

Processor privilege states	• It protects the processor and the activities that it performs. • It records the processor state in a register that can only be altered when the processor is operating in a privileged state. • It controls entry into the privilege mode using the hardware. • It prevents memory access through privilege-level mechanism.
Bounds checking	• It detects whether a variable is within certain bounds. • It prevents buffer overflows on input.

Software Security and Assurance: Parameter Checking

It is implemented by the programmer.

It involves checking the input data for disallowed characters, length, data type, and format.

It also uses other technologies, such as canaries, to protect against buffer overflows.

Software Security and Assurance: Memory Protection

It is necessary to protect the memory used by one process from unauthorized access by another.

It uses memory partitioning:

- To ensure processes cannot interfere with each other's local memory
- To protect common memory areas against unauthorized access

Software Security and Assurance: Granularity of Controls

It ensures that the security controls are granular enough to address both the program and the user.

Inadequate granularity of controls can be addressed by:

- Proper implementation of the concept of least privilege
- Setting reasonable limits on the user
- Separation of duties and functions
- Ensuring programmers are not the system administrators or users of the application
- Granting users only those permissions necessary to do their job

Software Security and Assurance: Separation of Environments

It is essential to control how each environment can access the application and the data. Control measures to protect the various environments include:

- Physical isolation of the environment
- Physical or temporal separation of data for each environment
- Access control lists
- Content-dependent access controls
- Role-based constraints
- Role definition stability
- Accountability
- Separation of duties

Software Security and Assurance: Prevention of TOC or TOU

The common time of check (TOC) or time of use (TOC or TOU) hazards are file-based race conditions.

To avoid them:

- The programmer should avoid any file system call that takes a filename for an input
- Files that are to be used should be kept in their own directory
- Directory must only be accessible by the universal ID (UID) of the program performing the file operation

Software Security and Assurance: Prevention of Social Engineering

Social engineering is a way where attackers try to use social influence over users to extract confidential information.

To protect against social engineering attacks:

- Provide users and help desk staff a proper framework to work
- Make users aware of the threat
- Give users the proper procedures for handling unusual requests for information

Software Security and Assurance: Backup

Backing up operating system and application software is a method of ensuring productivity in the event of a system crash.

Backup makes information available in the event of an emergency through data, programs, documentation, computing, and communications equipment redundancy.

Software Security and Assurance: Backup

The control functions include:

Redundant array of independent disks (RAID)

Disk mirroring

Maintaining the source code

Contingency planning documents

Software Security and Assurance: Software Forensics

It is the study of malicious software and protection against malicious code.

It can be used to:

- Determine whether a problem is a result of carelessness or was deliberately introduced as a payload
- Obtain information about authorship and the culture behind a given programmer
- Acquire the sequence in which related programs were written
- Provide evidence about a suspected author of a program
- Regulate intellectual property issues
- Recover source code that has been lost

Software Security and Assurance: Cryptography

They are used to protect the confidentiality and integrity of information.

Cryptographic techniques protect information by transforming the data through encryption schemes.

Encryption algorithms can be used to encrypt specific files located within the operating system.

Software Security and Assurance: Password Protection

Operating systems and application software use passwords as a convenient mechanism to authenticate users.

- Password protections include controls on:
  - How the password is selected
  - How complex the password is
  - Password time limits
  - Password length
- The most common solution is to encrypt password files with one-way encryption algorithms or hashing.
- It also involves an overstrike or password masking feature.

Software Security and Assurance: Mobile Code Controls

These controls manage and mitigate risks associated with mobile code that transmit across networks and execute on a local system without user intervention.

Implement the following controls:

- Disable mobile code in environments where it's not needed.
- Enforce secure configurations in browsers and applications to restrict its execution.
- Use digital code signing to verify the code’s integrity and publisher identity.
- Run mobile code with minimal privileges.
- Implement content security policies (CSPs) in web applications to control the allowed sources of scripts, styles, and other resources.

Software Security and Assurance: Sandbox

It places limits on the amount of memory and processor resources the program can consume.

It is one of the control mechanisms for mobile code.

It provides a protective area for program execution.

It can be created on the client side to protect the resource usage from applets.

Software Security and Assurance: Strong Language Support

It ensures that arrays stay in bounds, pointers are always valid, and code cannot violate variable typing.

It is a method of providing safe execution of programs such as Java.

Java does an internal check called static type checking.

Software Security: XML and SAML

These languages help provide software security.

XML

- It stands for Extensible Markup Language.
- It is a world wide web consortium standard for structuring data in a text file.
- It is extensible because the symbols are unlimited and can be defined by the user or author.

SAML

- It stands for Security Assertion Markup Language.
- It is a format that uses XML to describe security information.
- It requires the web browser single sign-on (SSO).

Methods to Assess the Effectiveness of Software Security

- System authorization
- Auditing and logging
- Risk analysis and mitigation
- Testing and verification

- It involves certification and accreditation or authorization of systems that process, store, or transmit information.
- It ensures a control framework is selected and uniformly implemented across the organization with the help of standards.

Methods to Assess the Effectiveness of Software Security

- System authorization
- **Auditing and logging**
- Risk analysis and mitigation
- Testing and verification

- It helps identify security issues, as most software is released with vulnerabilities.
- It ensures that organizations address these issues by applying procedures to maintain and check information integrity, accuracy.

Methods to Assess the Effectiveness of Software Security

- System authorization
- Auditing and logging
- **Risk analysis and mitigation**
- Testing and verification

It must be integrated into the SDLC as an ongoing activity and in change management. It involves:

- Using standardized methods outlined in frameworks, such as ISO and NIST, to assess risk and report to stakeholders
- Tracking and managing vulnerabilities
- Taking corrective actions for mitigation by reviewing and prioritizing the findings

Methods to Assess the Effectiveness of Software Security

- System authorization
- Auditing and logging
- Risk analysis and mitigation
- **Testing and verification**

All mitigations applied should be thoroughly tested and verified by independent security assessors to ensure that the security flaw has been mitigated.

Audit and Assurance Mechanisms

The following are the mechanisms used:

Certification and Accreditation (C and A)

It is a two-step process used to evaluate and approve a system for use.

Certification

- It is a technical review that assesses the security mechanism and evaluates its effectiveness.
- It may use safeguard evaluation, risk analysis, verification, testing, and auditing techniques.
- It ensures the system is right for the customer’s purpose.
- It is often an internal verification and is only trusted within the organization.

Certification and Accreditation (C and A)

Accreditation

- It is the formal declaration by the designated approving authority (DAA) that an IT system is approved to operate in a particular security mode using a prescribed set of safeguards at an acceptable level of risk.
- Once the accreditation is performed, management can formally accept the adequacy of an evaluated system's overall security performance.

Quick Check

A company is developing new software. What is the most critical step to minimize security vulnerabilities during development?

A. Train software developers on application security

B. Identify compliance requirements

C. Identify regulatory requirements

D. Develop and enforce a security policy

Security Impact of Acquired Software

Assessing the Security Impact of Acquired Software

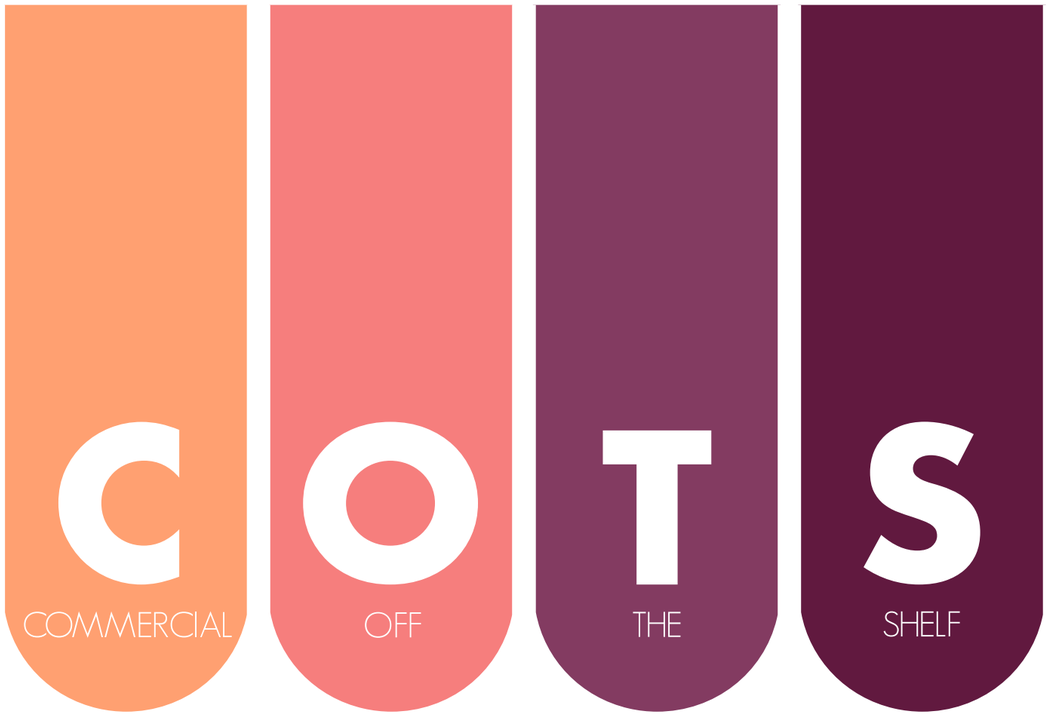
Acquired software can introduce new vulnerabilities into the system and may impact the organization’s risk posture.

The security can be assessed by:

- Using security tools to test the software for vulnerabilities
- Verifying whether the software development firm followed secure processes
- Checking developer conformance to international standards such as ISO 27034

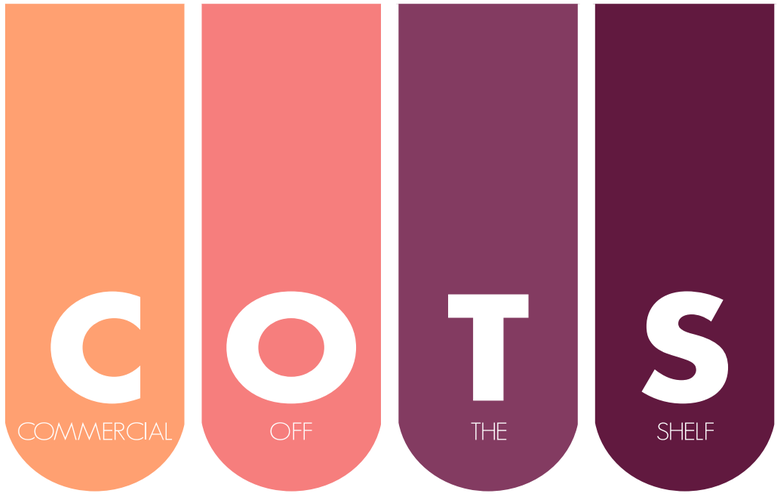
Commercial-off-the-Shelf (COTS)

It refers to hardware or software products that are ready-made and available for purchase in the commercial market.



Commercial-off-the-Shelf (COTS)

- It usually offers paid or free customer support.
- The vendors regularly release software and firmware patches to address vulnerabilities.
- Security vulnerabilities in them can introduce significant risk.
- The unavailability of source code can limit testing and monitoring.
- It helps companies validate if security testing has been performed with international standards such as Common Criteria and FIPS.
- It can perform black box and fuzzy testing.



Free and Open-Source Software (FOSS)

It is licensed to be free to use, modify, and distribute.

It allows other developers the opportunity to contribute to the development and improvement of a software.

Free and Open-Source Software (FOSS)

While FOSS is often available free of charge, its main difference from proprietary software is in the **freedom** it offers.

The **freedom to run** the program as one wishes, for any purpose

The **freedom to study** how the program works and change it, so it does compute as needed

The **freedom to redistribute** copies to help others

The **freedom to improve** the program and release modified versions to others

Free and Open-Source Software (FOSS)

Advocates of this software believe that if more users view the source code, they will eventually find all bugs and suggest how to fix them.

Linus's law states that **given enough eyeballs; all bugs are shallow.**

However, FOSS may allow hackers to find security vulnerabilities more easily than closed-source software.

Open-Source: Drawbacks

Security through obscurity isn’t enough to protect your system.

Both proprietary and open-source software have security vulnerabilities.

Many eyes on the code doesn’t guarantee timely review or patch updates.

Third-Party

Many organizations outsource software development projects to third party suppliers who:

Customize or tailor an existing commercial software

Develop new software for an organization’s business needs

Source: https://www.veracode.com/sites/default/files/pdf/resources/guides/best-practices-third-party-software-security-veracode-guide.pdf and https://apps.dtic.mil/dtic/tr/fulltext/u2/a495389.pdf

Third-Party

Risks

- Intellectual property rights dispute
- Inadequate software specification resulting in increased scope and costs
- Contract dispute which could adversely affect supplier delivery
- Poor security practices resulting in software vulnerabilities
- Lack of ongoing support

Best practices

- Third-party security policy development
- A risk-based approach to application security (threat modeling) undertaking
- Policies and security practices verification followed by the vendor
- Security metrics and SLA establishment
- Independent application security testing conduction

Quick Check

Your organization acquired proprietary software to manage customer data, but the team cannot inspect the internal code for vulnerabilities. What is the main challenge in assessing its security?

A. Inability to obtain the source code from the vendor

B. Inadequate in-house capability to perform penetration test

C. Poor software development practices followed by the vendor

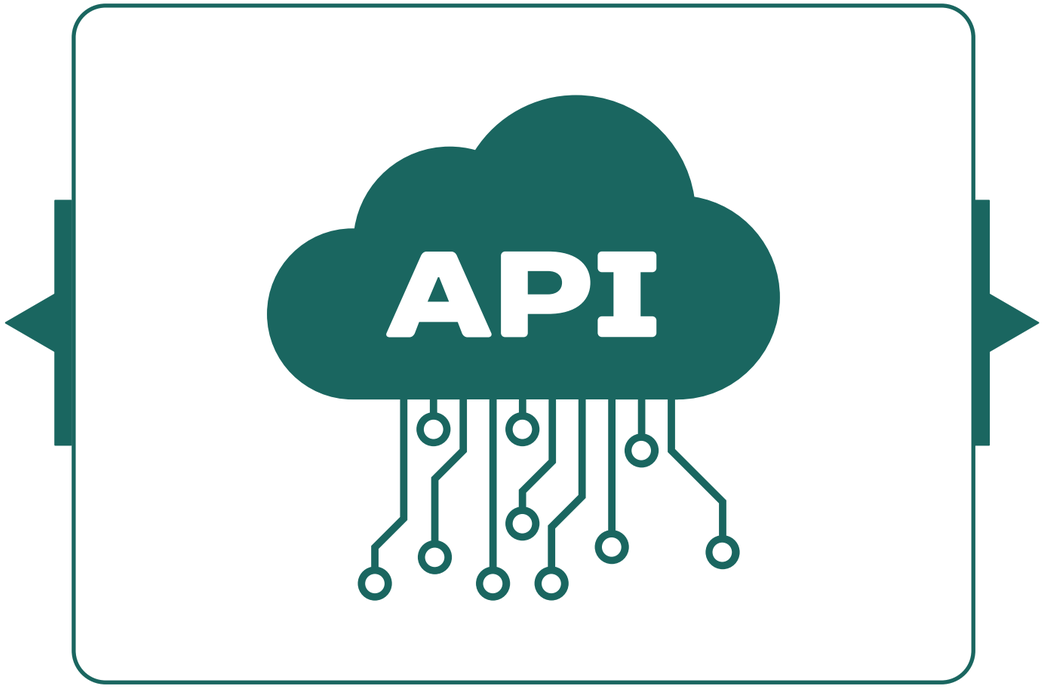
D. Unavailability of any third-party certification to validate the software

Define and Apply Secure Coding Guidelines and Standards

Security of Application Programming Interfaces (APIs)

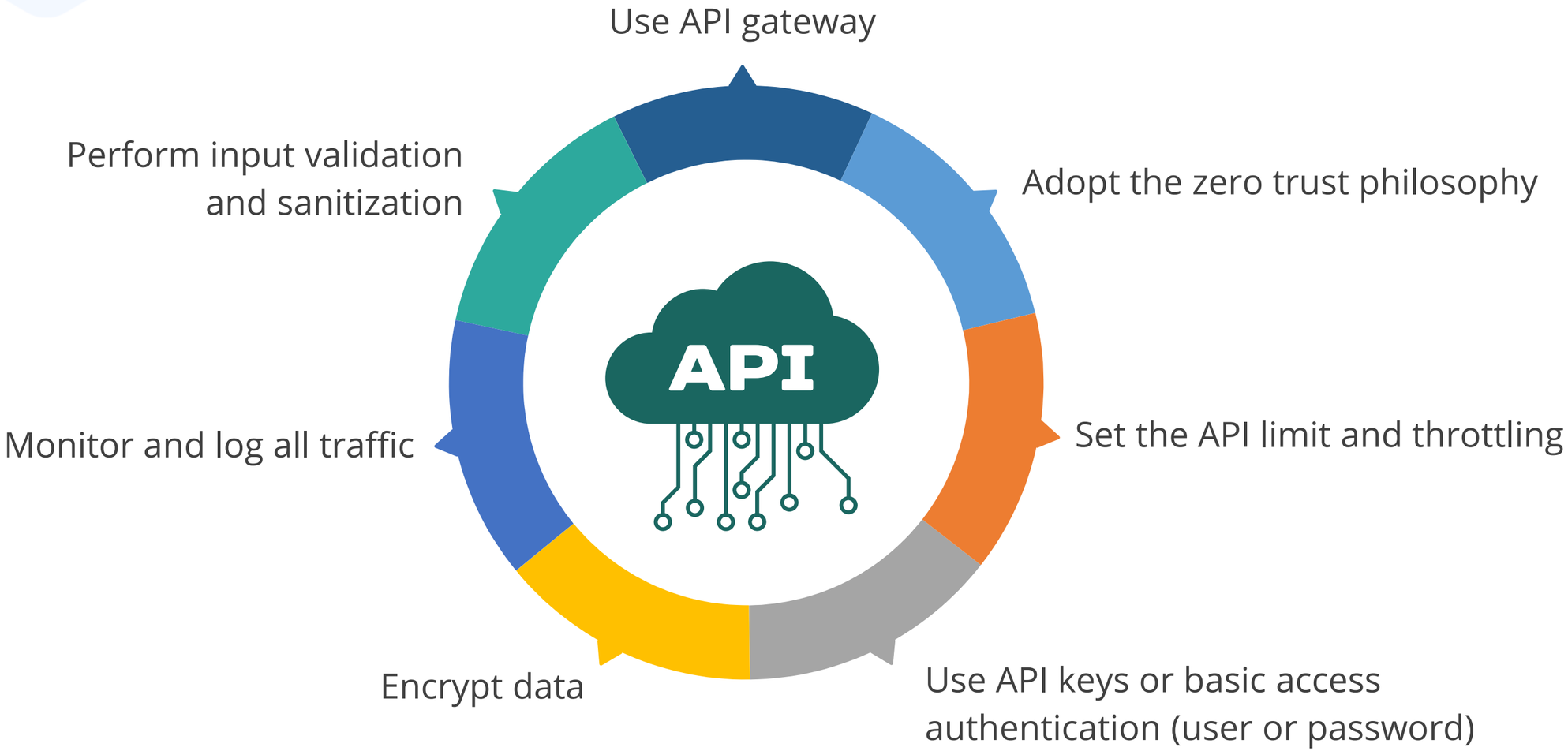
API is an interface that enables the transfer of data between two or more applications.

Simple object access protocol (SOAP):
A protocol and standard for exchanging information between web services in a structured format



Representational state transfer (REST):
A software architecture style consisting of guidelines and best practices for creating scalable web services

Security of APIs: Best Practices



API Formats

Simple object access protocol (SOAP)

- Utilizes an XML-based message protocol
- Uses SOAP envelope and then HTTP (FTP/SMTP) to transfer the data
- Supports only XML format
- Exhibits slower performance, scalability challenges, and lacks support for caching
- Is used where REST is not possible and provides WS-* features

Representational state transfer (REST)

- Follows an architectural style protocol
- Relies on simple hypertext transfer protocol (HTTP) exclusively
- Supports many data formats like JavaScript Object Notation (JSON), extensible Markup Language (XML), and Yet Another Multicolumn Layout (YAML)
- Offers improved performance, better scalability, and utilizes caching
- Is widely used

OWASP Secure Coding Practices

Input validation

Output encoding

Authentication and password management

Session management

Access control

Cryptographic practices

Memory management

General coding practices

Data protection

Communications security

System configuration

Database security

File management

Software-Defined Security (SDS)

It is a type of security model in which the information security in a computing environment is implemented, controlled, and managed by a security software.

Features

01 Abstraction of hardware

02 Centralized management and control

03 Automation and orchestration

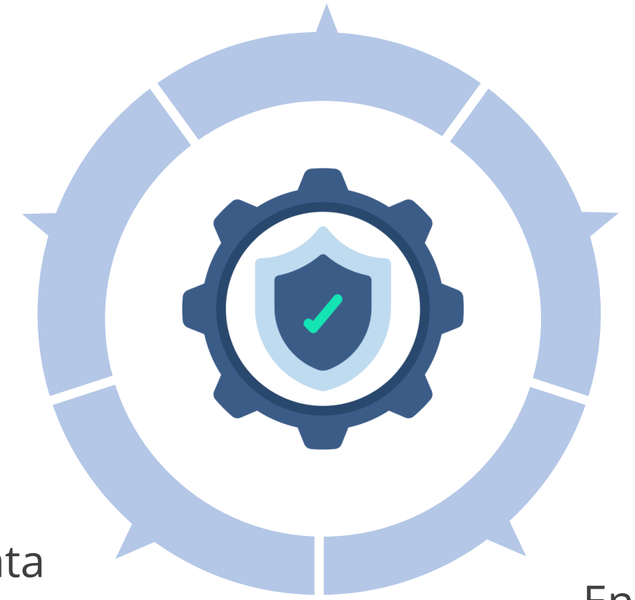
04 APIs and programmability

05 Elasticity and scalability

Software-Defined Security: Benefits

Enables automated provisioning and orchestration of security controls via policy

Removes human errors via higher levels of automation



Delegates security to protect workload and information regardless of location

Helps limit the scope of data security breaches through segmentation

Entitles security to protect dynamic cloud-based workloads

Web-Based Vulnerabilities

These are weaknesses or flaws in web applications that attackers can exploit to gain unauthorized access to data, systems, or entire networks.

These vulnerabilities can exist on the server side (the code running on the web server) or the client side (the user's web browser).

Types of Web-Based Vulnerabilities

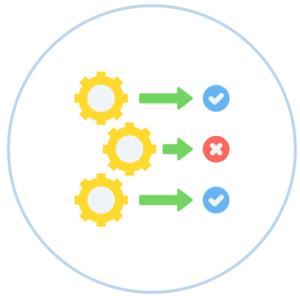
SQL injections

Cross-site scripting

Cross-site request forgery

Buffer overflow

Malicious update



Race condition

SQL Injections

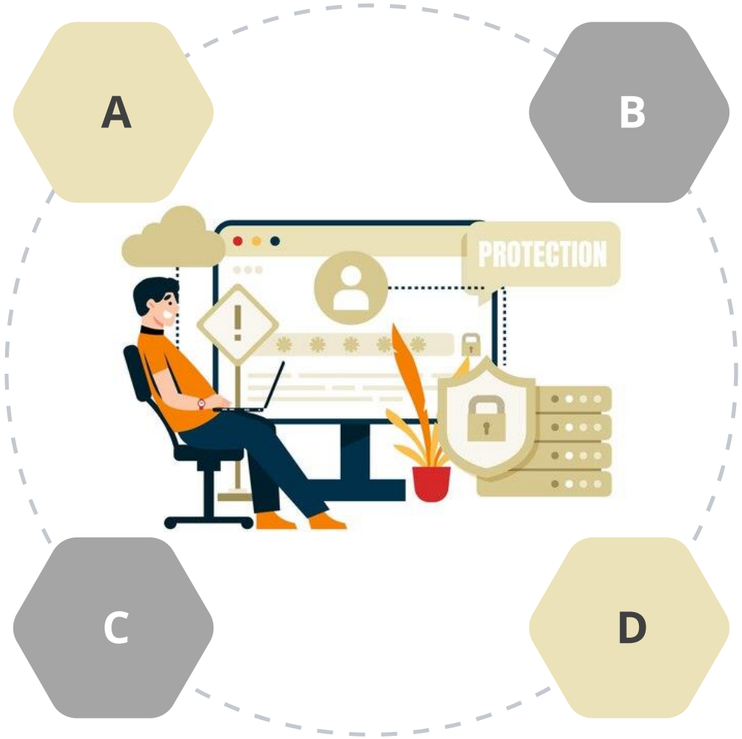
It refers to an attack where an attacker can execute malicious SQL statements to control a web application’s database server.

- These attacks allow a malicious individual to directly perform SQL transactions against the underlying database, bypassing the isolation model.
- These attacks can damage the database, retrieve information, or manipulate data.

SQL Attack Prevention

A. Update and patch

B. Perform input validation

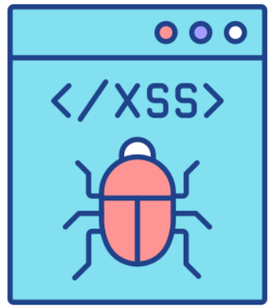


C. Limit account privilege

D. Use stored procedure

Cross-Site Scripting (XSS)

It involves injecting malicious scripts into otherwise benign and trusted websites.



- It occurs when an attacker uses a web application to send malicious code, typically in the form of a browser-side script, to a different end user.
- It enables attackers to inject client-side scripts into web pages viewed by other users.

XSS Attack: Illustration

1

A web application, abc.com, is hosted on a server with an XSS vulnerability.

2

This XSS vulnerability allows hackers to inject malicious scripts into the web server.

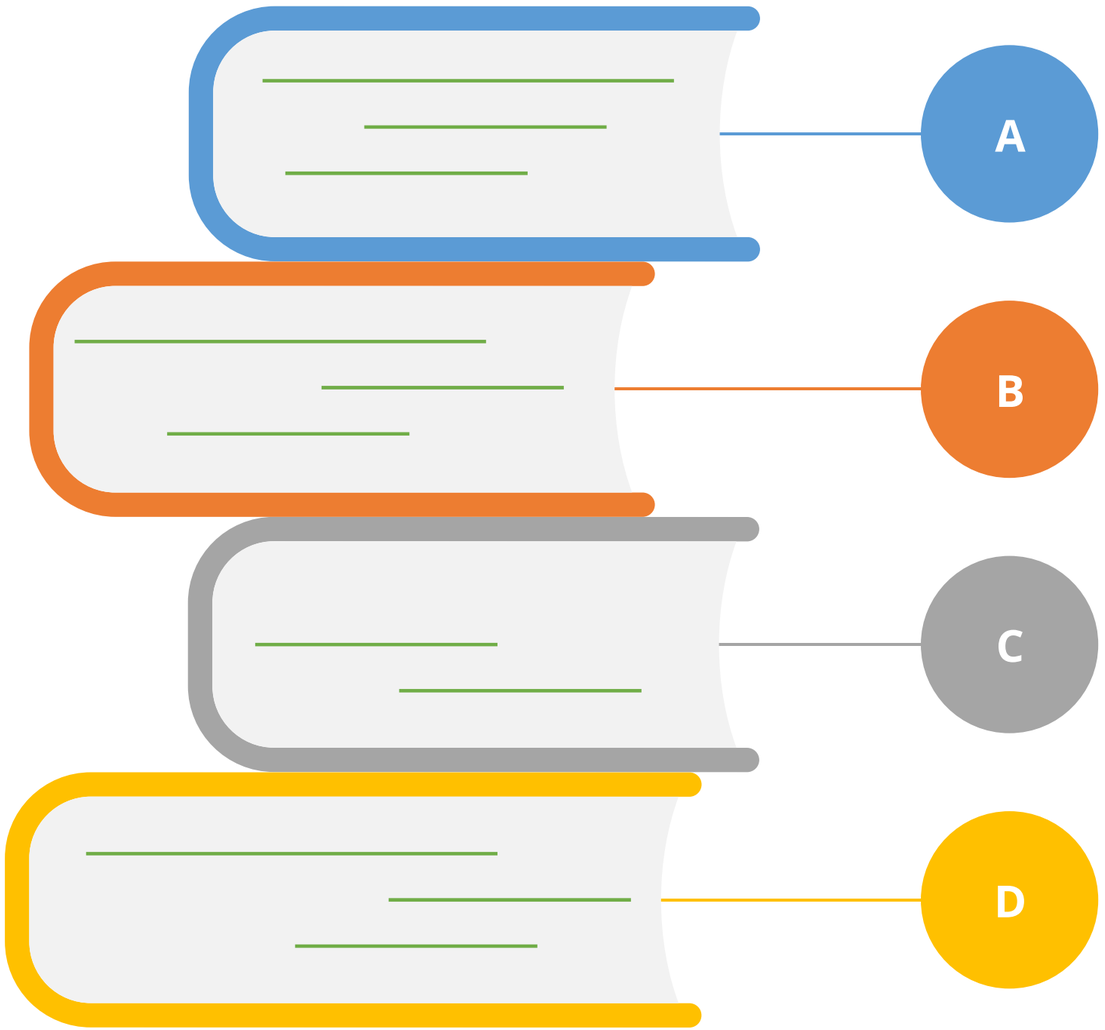
3

When a user accesses abc.com through their browser for daily tasks, the injected malicious script executes on the user's browser.

4

This execution enables the hacker to steal valuable information or perform any designated task.

Impact of XSS



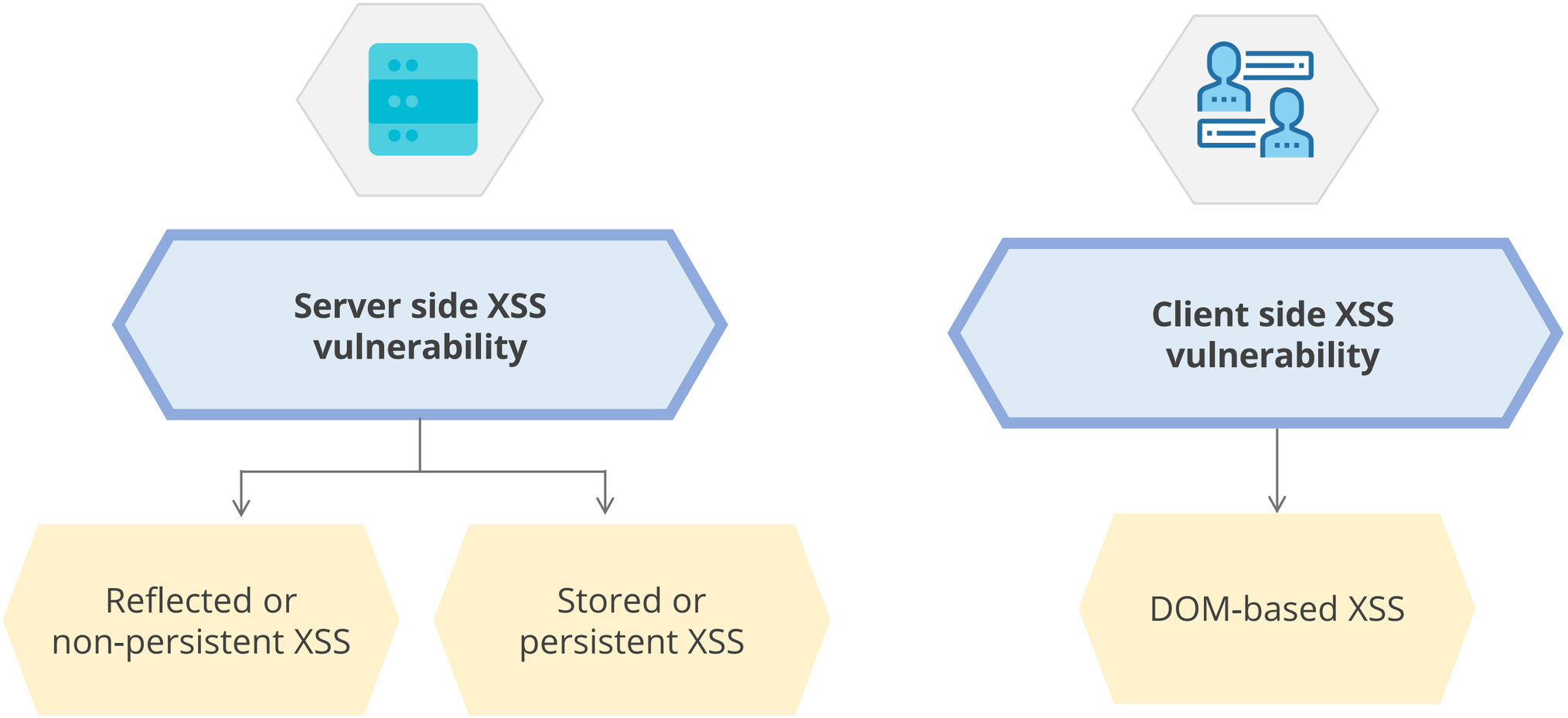
Session hijacking

Deploying hostile content

Impersonating users

Phishing

Types of XSS


Server side XSS vulnerability
Reflected or non-persistent XSS
Stored or persistent XSS
Client side XSS vulnerability
DOM-based XSS

Types of XSS

Reflected or non-persistent XSS

- It occurs when an attacker tricks a victim into processing a URL with a rogue script to steal sensitive information.
- It exploits the lack of proper input or output validation on dynamic websites.

Non-reflected or persistent XSS

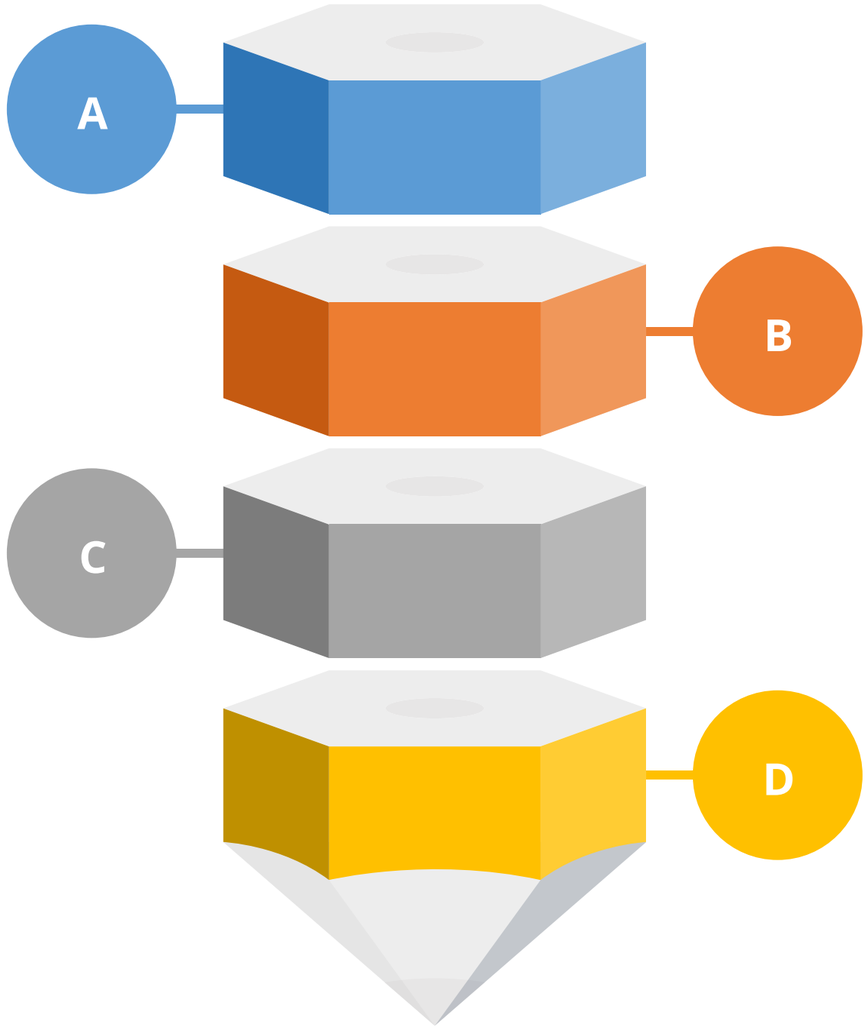
- It is also known as stored or second-order vulnerabilities and generally targets websites that allow users to input data stored in a database or another location.
- The attacker posts text containing malicious JavaScript and when users view these posts, their browsers render the page and execute the attacker's JavaScript.

DOM-based XSS

- It is a client-side vulnerability that lets attackers inject malicious scripts into a web page's document object model.
- The DOM acts like a blueprint detailing the structure, content, and interactions of elements within the browser.

Prevention of XSS

Limiting types of uploads



Proper coding practice and code review

Implementing input validation

Updating and patching

Cross-Site Request Forgery (CSRF)

These attacks exploit unintended behaviors that are legitimate within defined use but occur under unauthorized circumstances.

- They are executed against sites with authenticated users, leveraging the site’s trust from a prior authentication event.
- The attacker tricks the user’s browser into sending an HTTP request to the target site, thereby exploiting this trust.

CSRF: Illustration

01

Suppose a bank allows users to perform transactions without re-authenticating each time, such as transferring money between accounts.

02

A user opens a banking application and authenticates the banking website.

03

If the user remains logged in and does not close their browser, another browser tab could send a hidden request to the bank, resulting in a transaction that appears authorized but was not initiated by the user.

04

The transaction executes successfully because the banking application assumes it is initiated by the user, even though it originates from a malicious application in a different tab or browser.

Prevention of CSRF

Train and maintain awareness

Scan and test applications regularly

Limit authentication time

Set cookie expiration

Controls for Web Application Vulnerability

A

Validations and sanitization

B

Strong authentication

C

Regular patching

D

Secure coding practices and regular testing

Buffer Overflow

It occurs when more data is placed into a fixed-length buffer than it can handle.

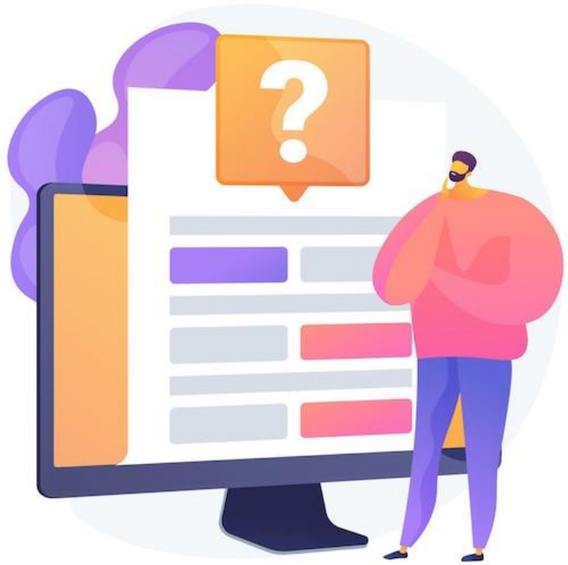
This extra information overflows into adjacent memory space, corrupting or overwriting the data held there.

Buffer Overflow: Example

The Slammer worm, also known as the SQL slammer, occurred in January 2003.

- It exploited a buffer overflow vulnerability in Microsoft SQL server.
- The worm was spread rapidly by sending a small, specially crafted data packet to vulnerable servers, causing a buffer overflow in the server's memory.
- As a result of this overflow, the worm's code was executed in the server's memory space, generating a flood of network traffic as it attempted to infect other vulnerable systems.

Reason for Buffer Overflow



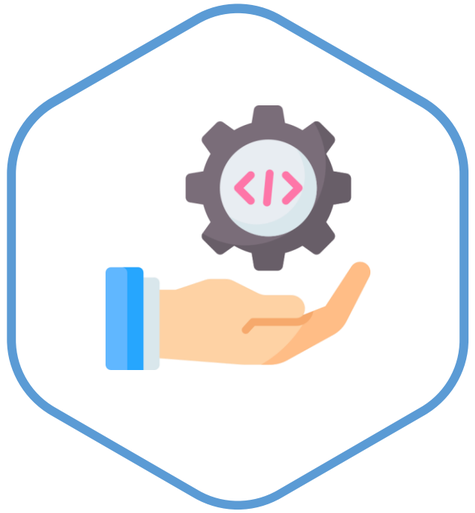
A. Poor programming practice

B. Programming language weakness

C. Lack of input validation

D. Poor management of memory allocation

Prevention of Buffer Overflow



Proper
coding practice

Proper
code review

Proper
input validation

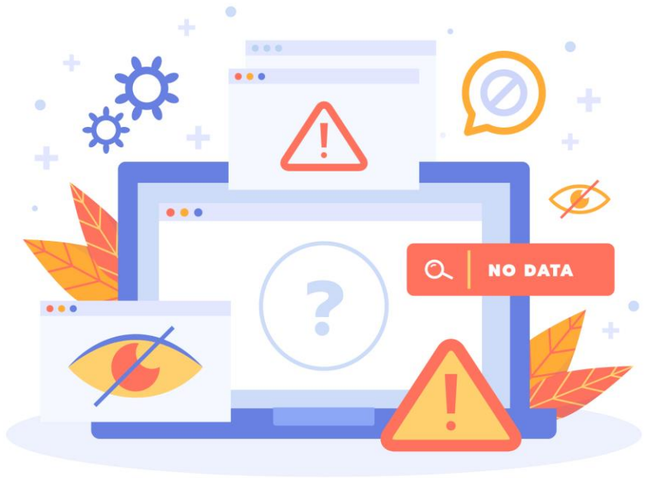
Malicious Update

It occurs when legitimate software or firmware is altered or replaced with a version containing harmful code through an update mechanism.

- Attackers disguise malicious code as a routine update, exploiting the normal update process.
- Users unknowingly install the harmful version, risking theft of sensitive information, unauthorized access, or other damage.

Malicious Update: Example

In 2017, CCleaner, a popular utility software for cleaning and optimizing computers, was compromised when hackers breached the supply chain of its parent company, Piriform.

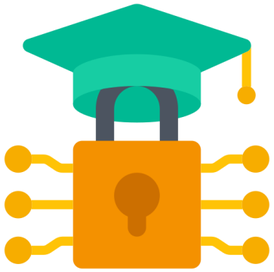


- The hackers injected malicious code into a legitimate CCleaner software update.
- This compromised update was then distributed to millions of users who trusted the software's legitimacy.
- Once installed, the hidden malware provided hackers with unauthorized access to infected systems, enabling them to collect sensitive information and deliver additional payloads for future attacks.

Prevention of Malicious Updates

Using secure channels for updates

Verifying updates with digital signatures



Educating users about the risks of updates from untrusted sources

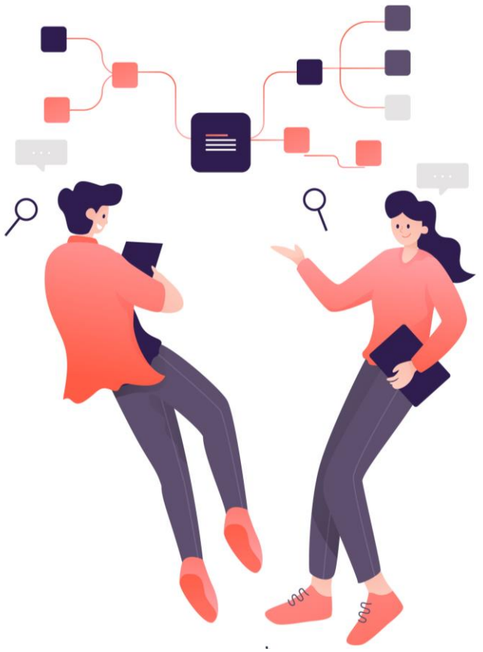
Race Condition

It occurs when two instructions from separate threads attempt to access the same data simultaneously.

Example

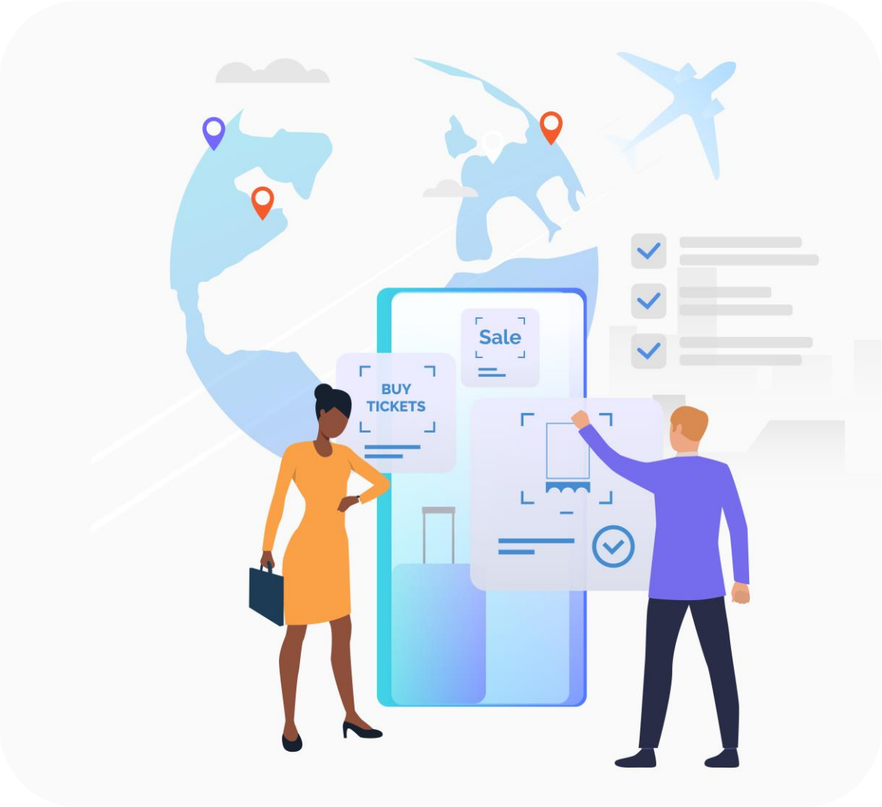
One person might view a file’s attributes while another person simultaneously accesses and modifies the same file.

This illustrates the Time of check to time of use (TOC/TOU) issue, where the person accessing the file might modify its data, inadvertently overwriting the information being viewed by the first person.



The developer must ideally program the threads to access the data sequentially.

Race Condition: Example



- Suppose two passengers, Alice and Bob, are trying to book the last available seat on a flight simultaneously.
- Alice initiates the booking process and checks the seat's availability.
- At the same time, Bob begins the booking process and sees that the seat is available.
- However, between Alice’s checking and her booking confirmation, Bob confirms his booking.
- The system processes both transactions simultaneously.
- Due to the time gap between Alice’s availability check and her booking confirmation, both bookings are accepted, resulting in an overbooked flight.

Web Application Environment Security: Specific Protection

It is important to secure the web application environment against unauthorized access and cyber threats by implementing the following measures:

- Assigning assurance sign-off process for web servers
- Hardening the OS used on such servers by:
  - Removing default configurations and accounts
  - Configuring permissions and privileges correctly
  - Keeping it up to date with vendor patches
- Extending web and network vulnerability scans prior to deployment

Web Application Environment Security: Specific Protection

- Passively assessing using:
  - Intrusion detection system (IDS)
  - Advanced intrusion prevention system (IPS) technology
- Using application proxy firewalls
- Disabling any unnecessary documentation and libraries

Web Application Environment Security: Administrative Interface Protection

- It restricts access to authorized hosts or networks and then uses strong (possibly multifactor) user authentication.
- It ensures the security of the credentials.
- It uses account lockout, extended logging, and audit and protects all authentication traffic with encryption.

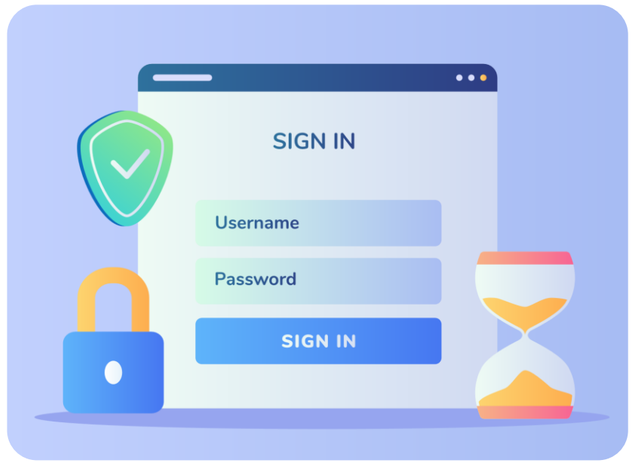
Web Application Environment Security: Input Validation

It ensures that the proxies can deal with problems of:

- Buffer overflows
- Authentication issues
- Scripting
- Submission of commands to the underlying platform and encoding issues
- URL encoding and translation

Web Application Environment Security: Sessions Protection

- The sessions or periods of any attachment to a server are controlled by other technologies, such as cookies or URL data, which must be protected and validated.
- If using cookies, always encrypt them.
- Do not use sequential, calculable, or predictable cookies, session numbers, or URL data for these purposes.
- Use random and unique indicators.



Web Application Environment Security: Web Applications Protection

The following methods should be implemented to ensure the web application is secure:

- Validate all input and output
- Use defense in depth
- Implement fail-secure (closed) system
- Use a secure network design
- Make the application or system simple

Quick Check

A web application allows users to submit forms with personal information. To ensure data integrity and security, which of the following practices should the development team implement?

A. Performing both client-side and server-side validation

B. Performing client-side validation

C. Performing server-side validation

D. Performing input validation

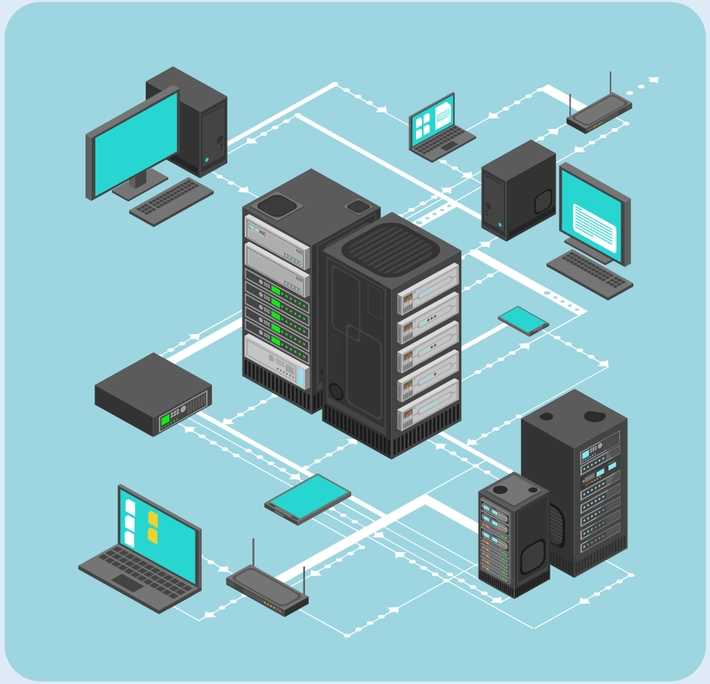
Database Security

Database

It is a structured collection of related data that allows queries, insertions, deletions, and many other functions.

The model should provide the following:

- Persistence
- Data sharing
- Recovery or fault tolerance
- Database language
- Security and integrity



Database Terms

Database	A cross-referenced collection of data
DBMS	A system that manages and controls the database
Tuple	A row in a two-dimensional database
Attributes	A column in a two-dimensional database
Primary key	Columns that make each row unique (every row of a table must include a primary key)
Foreign Key	An attribute of one table that is related to the primary key of another table
Cell	The intersection of a row and a column
Schema	The structure definition of the database

Relational Database

It is a simple model that provides flexibility and organizes data into relations or tables.

Activity code	Activity name
23	Patching
24	Overlay
25	Crack sealing

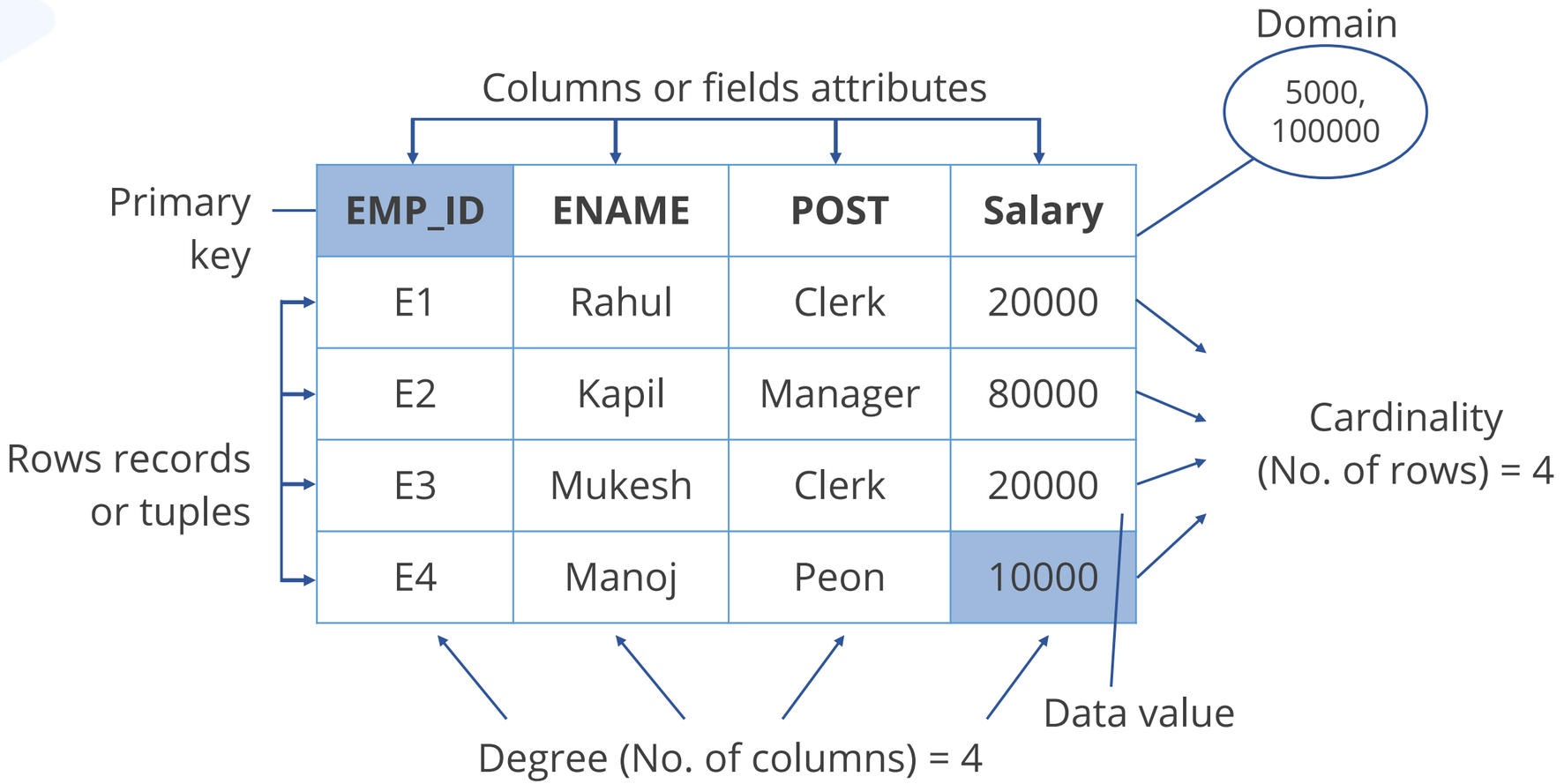
Date	Activity code	Route No.
01/12/01	24	1-95
01/15/01	23	1-495
02/08/01	24	1-66

Key = 24

Activity code	Date	Route No.
24	01/12/01	1-95
24	02/08/01	1-66

- Data can be associated across multiple tables with a key.
- The most common language used is structured query language (SQL).

Relational Database



Cardinality and degree in relational database	The number of rows in a relation is referred to as cardinality, and the number of columns is the degree.

Database Components

Data definition language: It defines the schema and structure of the database, access operations, and integrity procedures.

Data manipulation language: It contains all the commands that enable a user to view, manipulate, and use the database.

Data control language: It defines the internal language of the database.

Query language: It enables users to make requests to the database.

Report generator: It produces user-defined reports.

Database Integrity Services

Semantic integrity

It ensures structural and semantic rules are followed.

Referential integrity

It ensures no foreign key contains a reference to a primary key of a nonexistent record or null value.

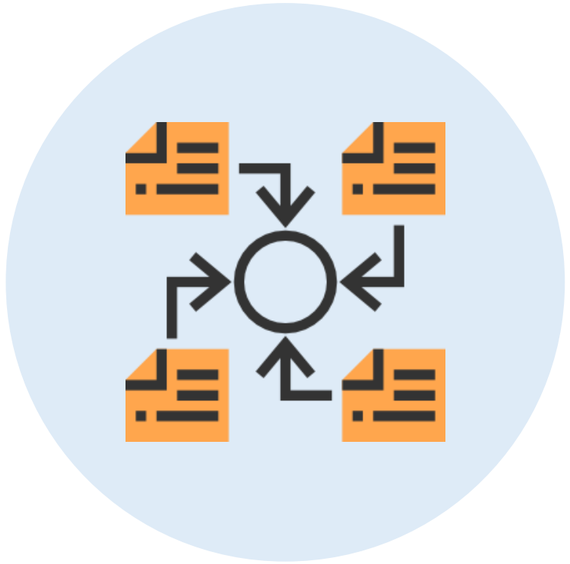
Entity integrity

It guarantees that tuples are uniquely identified by primary key values (every tuple must contain one primary value).

Aggregation

It is the act of combining information from separate sources.

- The combination of data forms new information, that the subject does not have the necessary rights to access.
- The combined data has higher sensitivity than the individual parts.
- To prevent aggregation, the subject and its proxies must be restricted from accessing the entire collection, including its components.
- The objects can be placed in level-based containers to prevent accessing subjects with lower-level permissions.
- A subject’s queries can be tracked, and context-dependent access can be enforced.
- An object’s access history can be tracked to block any suspicious aggregation attempts.

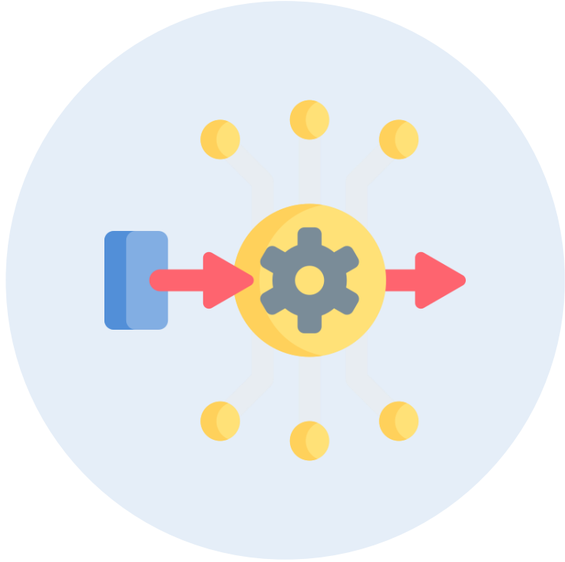


Inference

It refers to the process of extracting hidden meaning or knowledge from seemingly unconnected pieces of data to uncover a potential security threat.

Examples

- **Suspicious logins:** A sudden surge in login attempts from unknown locations can be a hacking attempt.
- **Data exfiltration:** A user downloading a large amount of sensitive data outside of normal work hours can be potential data theft.
- **Unusual activity:** A user accessing unauthorized websites or applications can be suspicious.



Database Views

These are virtual tables that act like windows into existing database tables providing a customized way to access and interact with data without modifying the underlying structure.

Using database views, databases can permit one group or a specific user to see certain information while restricting others from viewing it altogether.

Example

If a DBA wants to allow middle management members to see departments' profits and expenses but not the whole company's profits, they can implement views.

Polyinstantiation

In cybersecurity, it refers to a technique used to protect sensitive information in databases.

- It creates multiple copies of a single data record, with each copy containing different levels of detail depending on the user's access level or security clearance.
- Each copy is a version of the original data but with sensitive details removed or replaced with less sensitive information.
- Users are granted access to specific versions of the data based on their security role.

Example

A low-level employee might see limited data (name and department), while managers can see more (salary).

Online Transaction Processing (OLTP)

It is generally used when databases are clustered to provide fault tolerance and higher performance.

- It provides mechanisms that watch for problems and deal with them when they occur.
- If the process cannot be restarted, then the transaction is rolled back to prevent data corruption, or only part of a transaction is carried out.
- All transactions, valid or invalid, are logged in a transaction log for record-keeping.

Example

If a process fails, the monitor mechanisms within OLTP can detect and try to restart the process.

Database Transaction ACID Test

ACID is a set of properties that ensure data integrity and consistency in database transactions.

- Atomicity
- Consistency
- Isolation
- Durability

- Either all changes are completed, or the database is rolled back.
- Either the system completes the entire operation successfully, or the database rolls back to its prior state.

Database Transaction ACID Test

- Atomicity
- **Consistency**
- Isolation
- Durability

- Any change maintains data integrity or is canceled completely.
- All data is consistent in different databases.

Database Transaction ACID Test

- Atomicity
- Consistency
- **Isolation**
- Durability

- Transaction executes in isolation until completed, without interacting with other transactions.
- Any read or write will not be impacted by others of separate transactions.

Database Transaction ACID Test

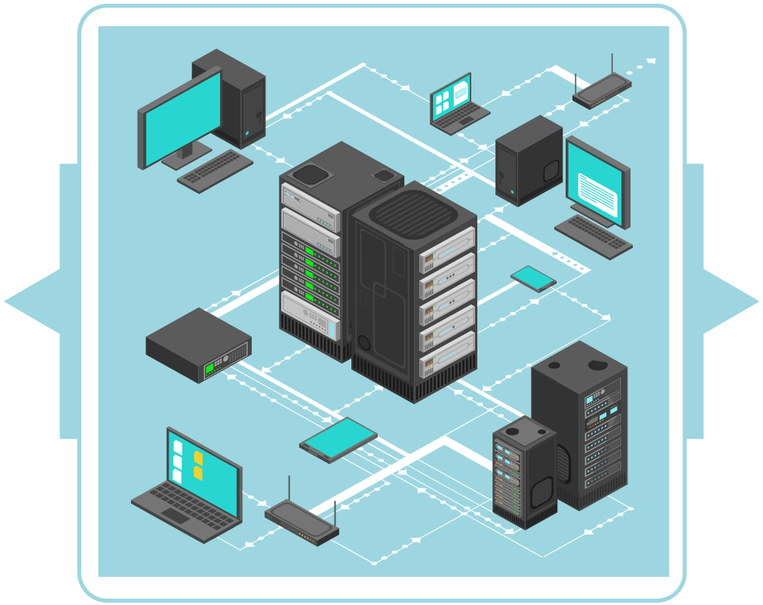
- Atomicity
- Consistency
- Isolation
- **Durability**

- Once the transaction is verified as accurate, it is committed, and the database cannot be rolled back.
- Successful transactions survive permanently.

Data Warehousing

It is a database designed to enable business intelligence activities.

It is designed for query and analysis and contains historical data derived from transaction data.



It works with data collected from multiple sources to enhance business intelligence.

Database Normalizations

It is the process of organizing the fields and tables of a relational database to minimize redundancy.

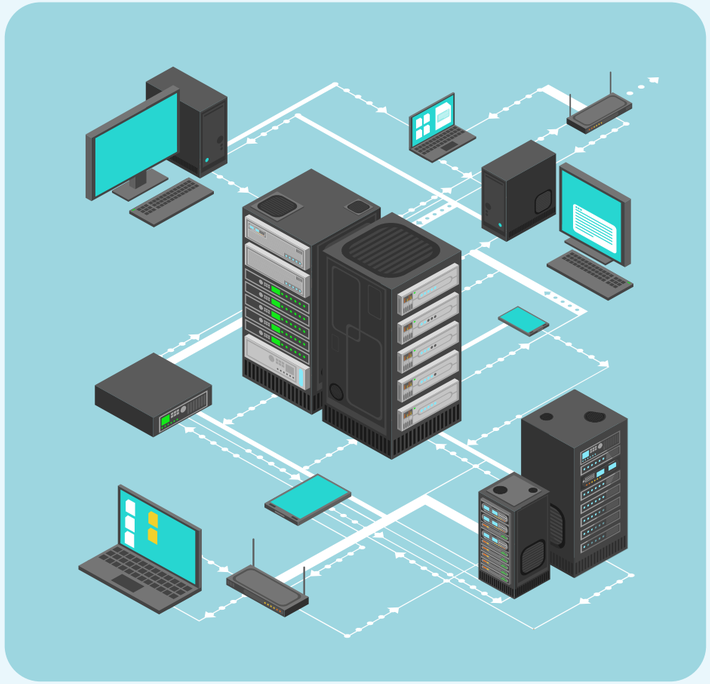
It decomposes tables to eliminate data redundancy and undesirable attributes like insertion, update, and deletion anomalies.

It is a multi-step process that puts data into tabular form, removing duplicated data from relation tables.

Data Mining

It involves processing the data stored in the data warehouse to derive more useful information.

- It is used to find an association and correlation in data to produce metadata.
- Metadata, the final output of data mining, is the result of processing data in a database and storing it in a data warehouse using tools to uncover abnormal patterns.
- The goal of data mining is to extract information to gain knowledge about activities and trends that were not visible earlier.



Quick Check

You have purchased some books from an e-commerce website. Which property of the relational database ensures that once a purchase has been made, it will remain so even in the event of power losses, crashes, or errors in the database server?

A. Atomicity

B. Consistency

C. Isolation

D. Durability

Indicators of Compromise and Indicators of Attack

Indicators

These are the signs of possible malicious actions within a system or network that indicate that an issue may exist.

- Behaviors, patterns, or anomalies indicate unauthorized access, malware infection, or other cyber threats.
- Observing and analyzing these indicators is essential for early detection and response to threats, helping to develop effective defense mechanisms.

Examples of Indicators

Account lockout

Early warnings of unauthorized access attempts, especially for privileged accounts

Concurrent usage

Sudden increases in traffic, indicating unauthorized access or a breach

Impossible travel

Multiple logins from distant locations in an unrealistically short timeframe, indicating a potential account compromise

Resource consumption

Unusual spikes in data consumption (such as high CPU or memory usage), indicating a malware infection or a DDoS attack

Resource inaccessibility

Sudden inaccessibility of critical resources, indicating cyberattacks

Out-of-cycle logging

Irregular log generation times, indicating suspicious activities and warrant investigation

Indicators of Compromise (IoC)

They serve as evidence of a cyberattack, indicating that a system is being compromised by unauthorized activity.

Artifacts detected on a network or within an OS serve as strong indicators of a computer intrusion.

Modern antimalware systems use known indicators of compromise to detect threats early, proactively protecting data and IT systems.

Indicators of Compromise (IoC)

They are used to detect:

01

Suspicious network activity

02

Anomalies in user accounts

03

System file changes

04

Suspicious emails or attachments

05

Security software alerts

Indicators of Attacks (IoA)

They indicate ongoing attacks or high risks by resembling suspicious behavior and being of a dynamic nature.

They provide early warnings of potential threats by identifying suspicious activities within a network.

They help organizations proactively defend against cyberattacks.

They identify ongoing or imminent cyberattacks, unlike IoCs that focus on evidence of a successful attack.

Indicators of Attacks (IoA)

They are used to detect:

01

Unusual login attempts

02

Escalation of user privileges

03

Abnormal data transfer patterns

04

Modification of critical system files

05

Trafficking from high-risk locations

IoC vs. IoA

Feature	Indicator of compromise	Indicator of attack
Focus	Evidence of past compromise	Signs of ongoing attack
Nature	Specific, static digital evidence	Broader, dynamic patterns of activity
Detection	Identifies already compromised systems	Proactively identifies potential attacks
Example	Known malware signature	Unusual login attempts

Malware Attacks

Malware

It consists of various harmful programs intentionally designed to execute unauthorized actions on a target system.

It includes software that disrupts or damages computers and networks.

It executes malicious actions to fulfill an attacker's goals, like data theft or system sabotage.

Impacts of Malware

It can have various detrimental effects on systems, including:

01 Stealing personal information

02 Deleting files

03 Conducting click fraud

04 Stealing software serial numbers

05 Using your computer as a relay

Types of Malware

Virus

Worms

Logic bombs

Spyware

Trojan horse

Back doors

Potentially unwanted programs

Ransomware

Adware

Bloatware

Rootkits

Keyloggers

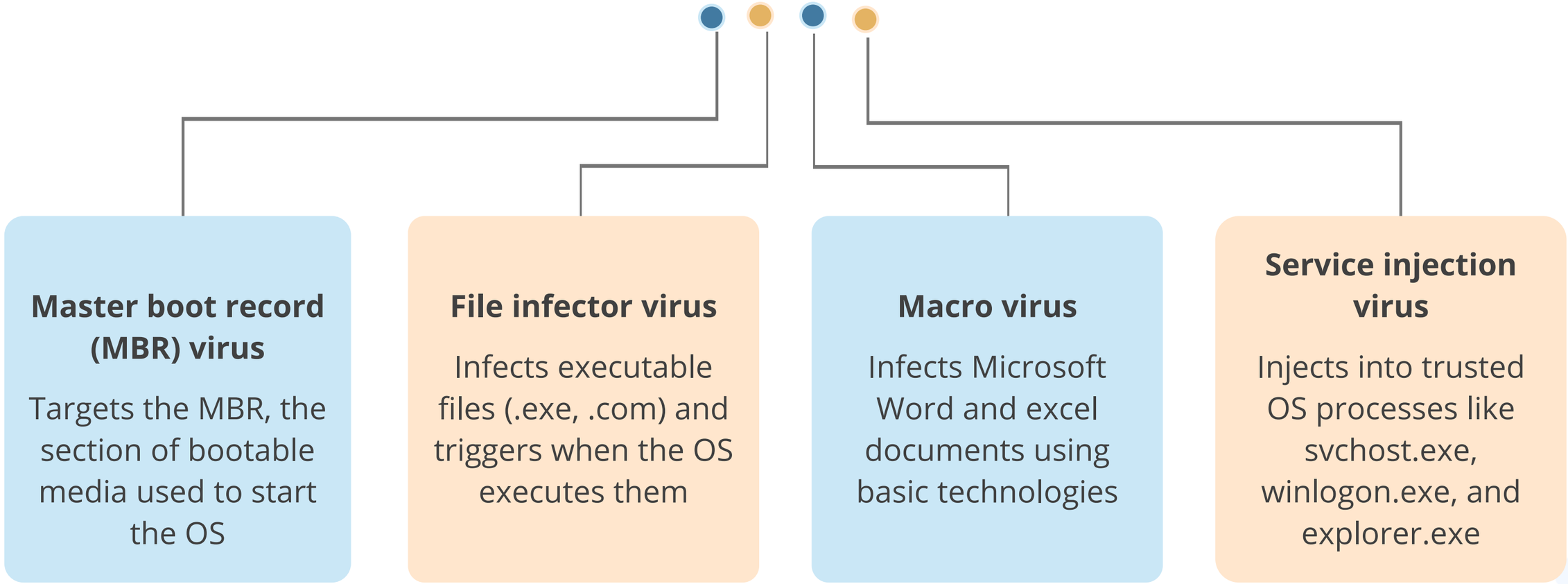
Virus

It is a malicious code that replicates by attaching itself to another piece of executable code.

- It executes when the host code runs, infecting other files and performing harmful actions.
- It can significantly slow down or crash a system, potentially leading to data loss.

Virus Propagation Techniques

Viruses use various methods to spread and infect systems. These include:



Master boot record (MBR) virus

Targets the MBR, the section of bootable media used to start the OS

File infector virus

Infects executable files (.exe, .com) and triggers when the OS executes them

Macro virus

Infects Microsoft Word and excel documents using basic technologies

Service injection virus

Injects into trusted OS processes like svchost.exe, winlogon.exe, and explorer.exe

Virus Technologies

Viruses employ various techniques to evade detection and propagate within systems. These include:

Multipartite viruses

Use multiple propagation techniques to penetrate systems

Polymorphic viruses

Modify their code as they travel from system to system

Stealth viruses

Hide by tampering with the OS to avoid detection

Encrypted viruses

Utilize cryptographic techniques to avoid detection

Worms

These are self-replicating pieces of code designed to penetrate networks and computer systems.

- They replicate and spread independently, similar to biological organisms.
- They exploit vulnerabilities to move through networks, consuming bandwidth and rapidly infecting new hosts.

Logic Bombs

They are malicious code objects that infect a system and lie dormant until triggered by specific conditions, such as:

- A specific time
- A program launch
- A website login
- Other predefined events

Spyware

It is a malicious software that secretly gathers a user’s personal information and activities.

- It installs itself without the user’s knowledge or consent.
- It transmits stolen data to a third party, posing a significant threat to privacy and security.

Trojan Horse

It is a type of malicious software that disguises itself as legitimate software to trick users.

- It appears to perform a useful function but secretly executes malicious actions.
- It is a standalone program that needs to be installed by the user, often through deception.
- It is designed to trick users into downloading or executing harmful software by mimicking legitimate files or programs.

Remote Access Trojans (RATs)

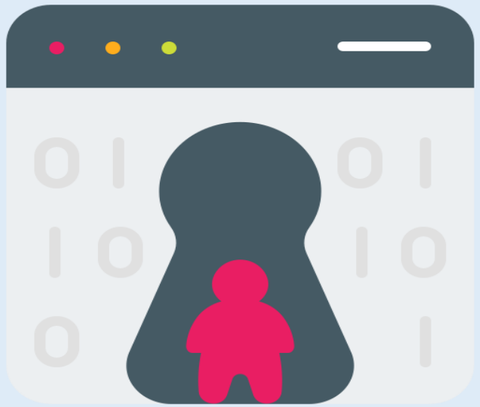
They are modern-day versions of Trojan horses in the cyber realm.

- They are stealthy infiltrators concealed within legitimate files, granting cybercriminals remote control over compromised systems.
- They allow command and control of the computer, enabling remote access for malicious activities such as data theft or malware installation.

Backdoor

It allows attackers to maintain access to a system after unauthorized entry.

- It ensures unrestricted access even if the initial entry method is blocked.
- It allows installation inadvertently by authorized users via Trojan horses.



Ransomware

It is a type of malicious software designed to extort money by encrypting the victim’s files, making them inaccessible until a ransom is paid.

- It infects systems by taking control of the victim's machine, including files and documents.
- It is typically installed through malicious email attachments, links in instant messages, or visits to compromised websites.

Adware

It is a type of malware that displays unwanted ads and pop-ups.

- It bombards the user with endless ads and pop-up windows, potentially dangerous to the device.
- It gathers personal information, though most adware is considered safe.

Bloatware

It is an unwanted software pre-installed on new devices.

It hogs storage space and slows down your device.

Examples

Trial versions of software, manufacturer-branded applications, promotional software, and unnecessary background programs

Rootkits

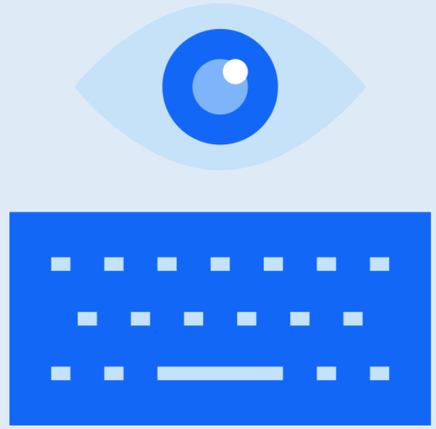
These are stealthy malwares designed to provide attackers with unauthorized, privileged access to a computer system.

- They dig deep into the system, usually at the root level, hiding their presence while giving attackers full control over the infected device.
- They alter the operation of the operating system to enable nonstandard functionality.

Keylogger

It is a type of malware designed to covertly record everything the user types on their keyboard.

- It acts like digital eavesdropper, capturing keystrokes including passwords, credit card numbers, and messages.
- It transmits stolen information to attackers, posing a significant threat to privacy and security.



Preventive Measures for Malware Control

Do not click suspicious links

Configure security settings

Enable strong passwords and two-factor authentication

Keep the software updated

Be cautious with downloads

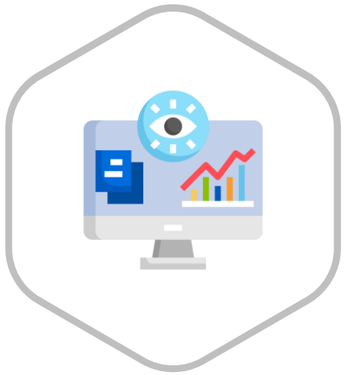
Use reputable antivirus or anti-malware programs

Beware of social engineering tactics

Detective Measures for Malware Control

These are essential to identify and mitigate malware threats. These include:

Regular malware scans



System monitoring

Security software alerts

Corrective Measures for Malware Control

They help to mitigate the impact of malware after detection. These include:

- Malware removal
- System restoration
- Attack reporting
- Vulnerability patching

Malware: Technical Controls

- Antivirus and antimalware software
- Endpoint detection and response
- Application whitelisting
- Firewalls
- Email security solutions
- Data loss prevention
- Security information and event management
- Network segmentation
- Operating system and application hardening

Quick Check

A development team is focused on building secure software. Which of the following actions does not align with best practices for software security?

A. Developing software with secure features

B. Protecting the brand and your customer's trust

C. Knowing the fundamental principles of software security

D. Deploying software with more features

Key Takeaways

- The various phases of SDLC are preparing a security plan, development or acquisition, implementation, operation or maintenance, and disposal.
- Database is a structured collection of related data that allows queries, insertions, deletions, and many other functions.
- A data warehouse is a storage facility comprising data from several databases.
- Indicators of compromise and indicators of attack are the signs of possible malicious actions within a system or network.
- Malware consists of various harmful programs intentionally designed to execute unauthorized actions on a target system.

Thank You