Security

Reading Material

PW SKILLS

Topics Covered

- Authentication and Authorization Systems
- Input Validation and Sanitization
- CORS
- Rate Limiting and Throttling APIs
- Secure Communication (Encryption, SSL/TLS)
- OWASP Top 10 Security Risks
- An Introduction to Data Protection Laws - GDRP and HIPAA
- Web Vulnerabilities - XSS
- Web Vulnerabilities - CSRF
- Web Vulnerabilities - SQL Injection
- CAPTCHA

Authentication and Authorization Systems

Authentication and authorization are key security terms that often get mixed up. Authentication verifies who you are, while authorization decides what you can access.

Methods of Authentication

- Knowledge-based Authentication (Something you know)
- Possession-based Authentication (Something you have)
- Inherence-based Authentication (Something you are)
- Location-based Authentication (Somewhere you are)
- Behavioral-based Authentication (Something you do)
- Multi-Factor Authentication (MFA)

Authorization Mechanisms

- Access Control Lists (ACLs)
- Role-Based Access Control (RBAC)
- Attribute-Based Access Control (ABAC)
- Policy-Based Access Control (PBAC)
- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC)

Question Time

Question: What is the primary difference between authentication and authorization in the context of security?
Question: Name two methods of possession-based authentication.

Input Validation and Sanitization

Introduction to Input Validation

Definition: Process of ensuring data entered into a system meets specified criteria.
Objective: Prevents invalid or malicious data from compromising application integrity and security.
Importance: Mitigates risks such as SQL injection, XSS attacks, and data corruption.

Importance of Input Validation

Functionality:
- Ensures correct data processing.
- Prevents errors (e.g., invalid credit card details).

Security:
- Blocks malicious data (e.g., SQL injection, XSS).
- Prevents unauthorized access and data breaches.

User Experience:
- Alerts users to invalid entries.
- Helps ensure valid submissions.
- Enhances user-friendliness.

Common Input Validation Techniques

- **Syntactic Validation:** Ensures inputs match expected formats (e.g., email addresses, phone numbers).
- **Semantic Validation:** Verifies that inputs make sense contextually and meet business rules (e.g., age must be a positive number).
- **Input Sanitization:** Removes or neutralizes potentially harmful characters or code from inputs to prevent security vulnerabilities.

Introduction to Input Sanitization

- **Definition:** Process of cleansing or filtering input data to prevent execution of malicious code or unintended behavior.
- **Purpose:** Mitigates risks of code injection (e.g., SQL injection, XSS) and ensures data integrity.
- **Integration with Validation:** Often used alongside validation techniques for comprehensive input security.

Input Sanitization Methods

- **Escaping:** Modifying special characters in input to neutralize their potential threat, such as converting '<' to '<' to prevent HTML injection.
- **Encoding:** Converting special characters into a safe format for transmission or storage, like converting '&' to '%26' for URL encoding.
- **Canonicalization:** Ensuring all input is in a standard, consistent format to prevent bypassing security measures through different representations (e.g., converting different forms of the same input to a single, standardized representation).
- **Using Safe APIs:** Utilizing programming interfaces designed to handle input securely, reducing vulnerabilities such as SQL injections or XSS attacks.

Preventing Injection Attacks

- **Validate Early:** Check input as soon as possible in the application workflow.
- **Use Whitelisting:** Accept only known good inputs rather than trying to filter out bad ones.
- **Sanitize and Validate Output:** Ensure outputs are also validated and sanitized to prevent injection attacks.
- **Regular Updates:** Keep validation and sanitization rules up-to-date with evolving security threats.

Question Time

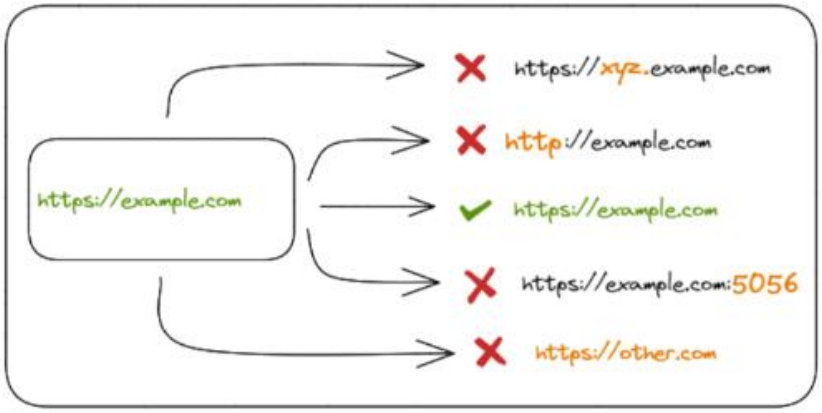
Question: What are the main purposes of input validation in an application?
Question: Give an example of a syntactic validation check and a semantic validation check.

Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing (CORS):

- A security feature implemented by web browsers.
- Allows or restricts web applications running at one origin from requesting resources from another origin.
- Controlled via HTTP headers that specify which origins are permitted.

The problem CORS is trying to solve



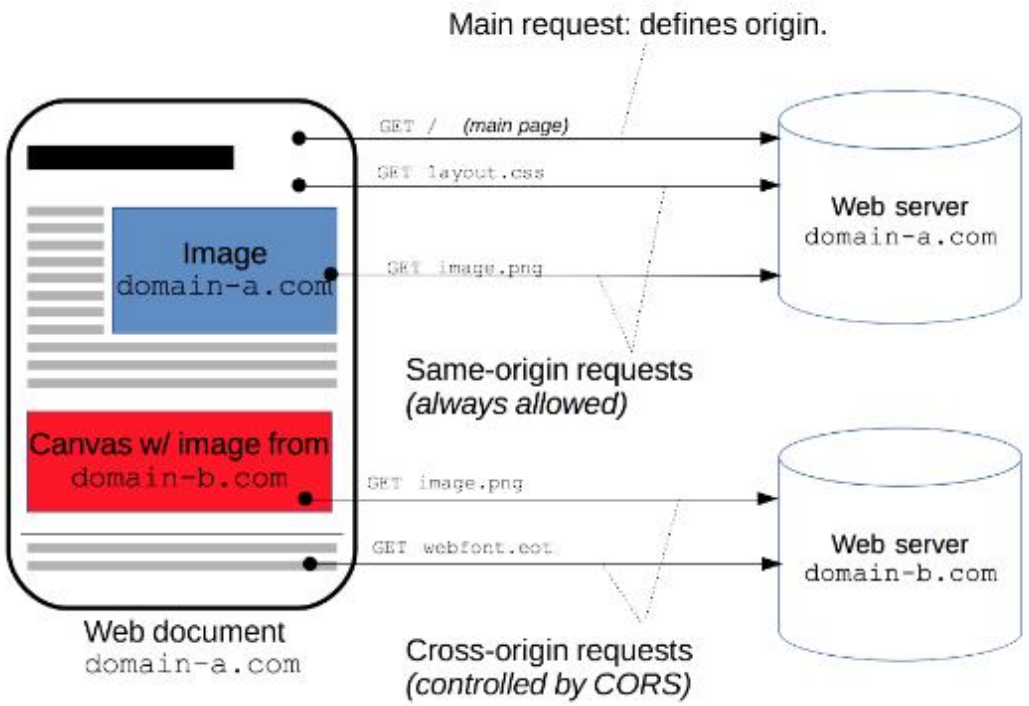
Without controls, a malicious site could make unauthorized requests to another site, potentially accessing sensitive information.

How does CORS work?

CORS Preflight Mechanism

Preflight Mechanism
- Checks if a cross-origin request is safe before sending it.
- Prevents attacks like cross-site scripting (XSS) and cross-site request forgery (CSRF).

Preflight Call
- A special HTTP OPTIONS request sent by the browser.
- Includes origin, request method, and headers.



CORS Workflow

Request Initiation
- Website on example.com requests data from api.example.org.

Preflight Request
- Browser sends HTTP OPTIONS request to api.example.org.
- Checks if the actual request is allowed.

Server Response
- Server responds with headers like:
  - Access-Control-Allow-Origin
  - Access-Control-Allow-Methods
  - Access-Control-Allow-Headers

Request Execution
- If allowed, the browser sends the actual request.
- Retrieves the requested data.

Why Are You Getting a CORS Error?

Top Reason for CORS Errors:

- The server you are requesting from does not include the expected Access-Control-Allow-Origin header in its responses.
- The server includes the header, but your frontend app's URL is not in the list of approved origins.

Few ways to resolve CORS errors:

Server Configuration

Include headers:
- Access-Control-Allow-Origin
- Access-Control-Allow-Methods
- Access-Control-Allow-Headers

Proxying Requests
- Use a proxy server or service to relay requests.
- Configure web server (e.g., Nginx, Apache) for proxying.

Browser Extensions
- Chrome: "Allow CORS: Access-Control-Allow-Origin"
- Firefox: "CORS Everywhere"

JSONP (JSON with Padding)
- Suitable for GET requests to bypass CORS.

Same-Origin Policy
- Keep requests within the same domain.

Enable CORS on External APIs
- Ensure third-party APIs support and enable CORS.

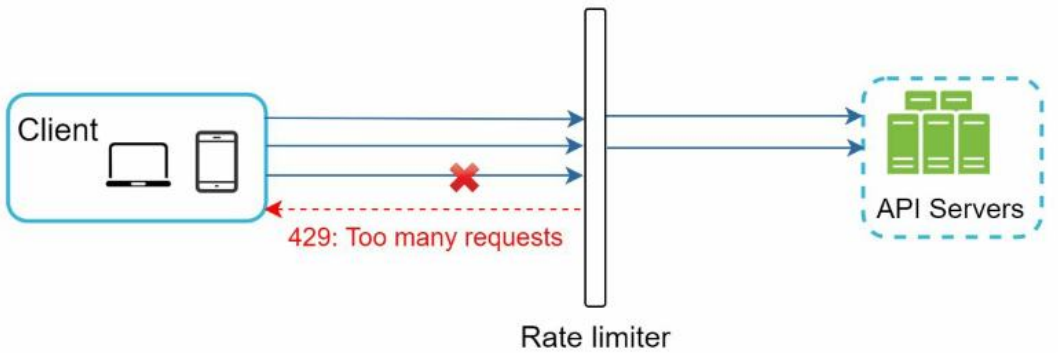
web security policies:
- Secure Communication :
- Input Validation :
- Access Control and Authorization:
- Authentication and Password Policies :
- Session Management :
- Content Security Policy (CSP) :
- Regular Security Updates :
- User Education and Awareness :

Question Time!

Question: What problem does CORS (Cross-Origin Resource Sharing) solve in web development?
Question: What is the purpose of the CORS preflight mechanism?

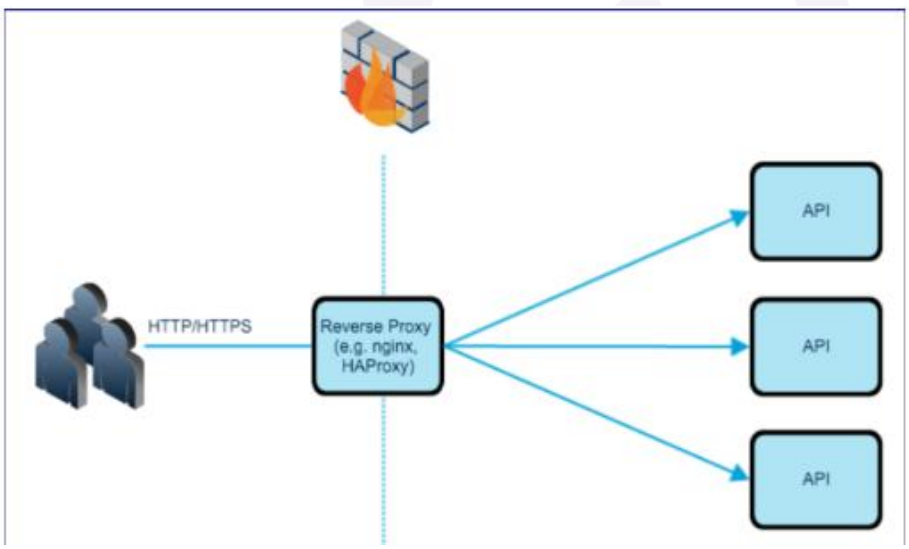
Rate Limiting and Throttling APIs

Rate limit



Rate limiting controls the number of API requests a client can make within a specified time frame. This prevents overuse or abuse of the API, ensuring that resources are available to all clients.

Throttling APIs to Prevent Abuse:



Throttling controls the rate at which a client can make requests to the API over time. It can either delay the requests or reject them if they exceed the threshold.

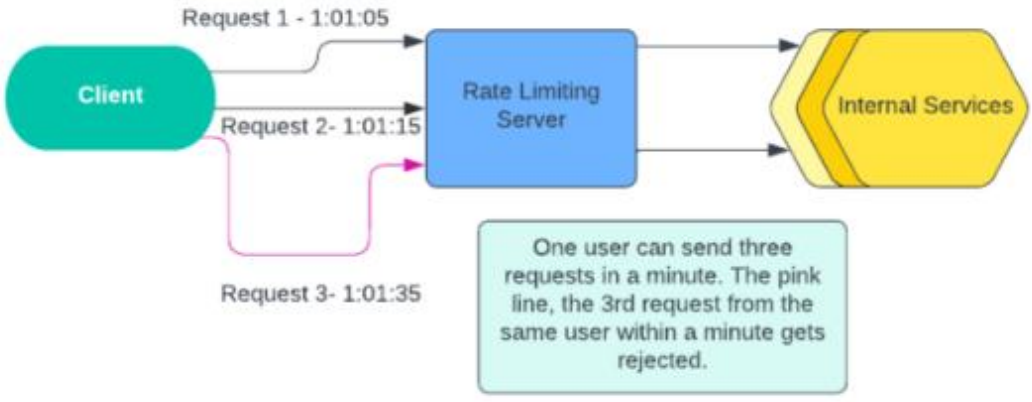
Why Rate Limiting and Throttling?

- Reasons for Implementation
- **Prevent Abuse:** Protects APIs from excessive use or attacks.
- **Ensure Fair Usage:** Ensures all users have fair access to resources.
- **Maintain Performance:** Helps maintain server performance and stability.

How Rate Limiting Works

Mechanism

- **Set a Limit:** Define a maximum number of requests allowed.
- **Track Requests:** Monitor the number of requests made by each user.
- **Enforce Limits:** Block or slow down requests that exceed the limit.



Types of Rate Limiting

Common Types

- **Fixed Window:** Limits requests in a fixed time period (e.g., 100 requests per hour).
- **Sliding Window:** Limits requests over a rolling time window.
- **Token Bucket:** Uses tokens to allow requests until tokens run out.
- **Leaky Bucket:** Limits request rate by processing requests at a constant rate.

How Throttling Works

Mechanism

- **Define Limits:** Set maximum request rates.
- **Monitor Requests:** Keep track of the rate at which requests are made.
- **Control Flow:** Slow down or pause request handling to stay within limits.

Tools Rate Limiting

- AWS API Gateway
- Apigee
- Kong
- NGINX

Libraries for Rate Limiting

- Python
  - Flask-Limiter
  - Django-ratelimit
- Node.js
  - express-rate-limit
  - Rate-limiter-flexible
- Java
  - Bucket4j
  - Resilience4j

Question Time

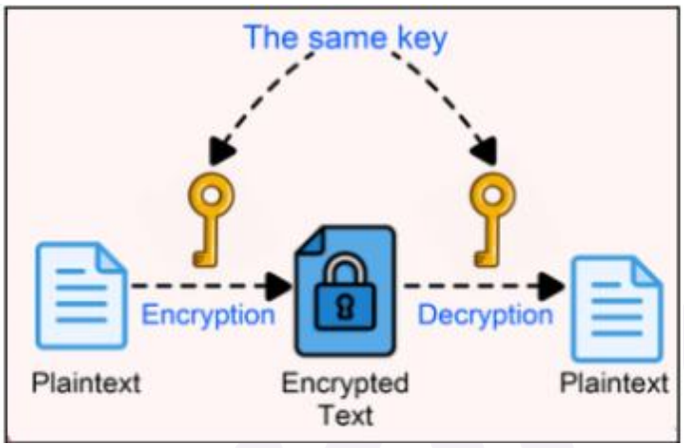
Question: Why is rate limiting necessary when throttling APIs, and what purpose does it serve?
Question: Name a tool or library commonly used for implementing rate limiting in web applications.

Secure Communication (Encryption, SSL/TLS)

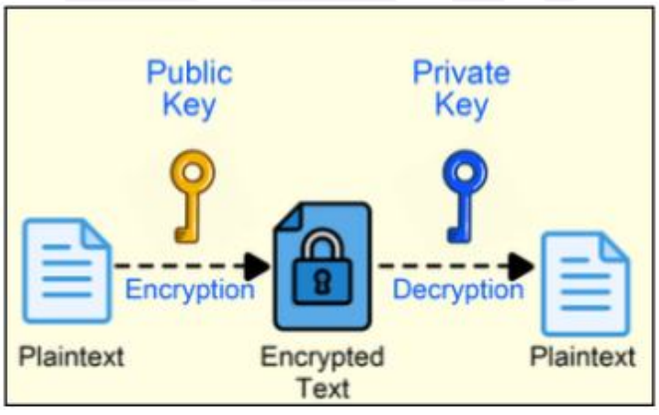
Secure communication is crucial in modern computing and internet interactions, ensuring data transmitted between parties stays confidential and unaltered.

Encryption: Encryption is the process of converting data into a coded format to prevent unauthorized access, ensuring data privacy and security.

Symmetric Encryption:



Asymmetric Encryption:



Secure Communication (Encryption, SSL/TLS)

Introduction to Secure Communication

Why Secure Communication Matters

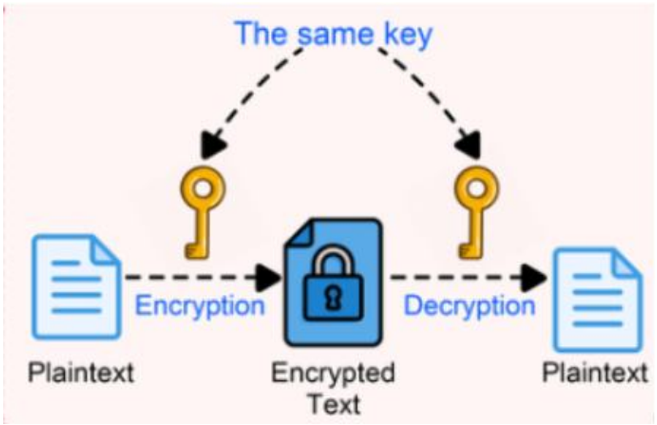
- Protects sensitive data during transmission.
- Prevents unauthorized access and tampering.
- Ensures trust and integrity in digital interactions.

Encryption

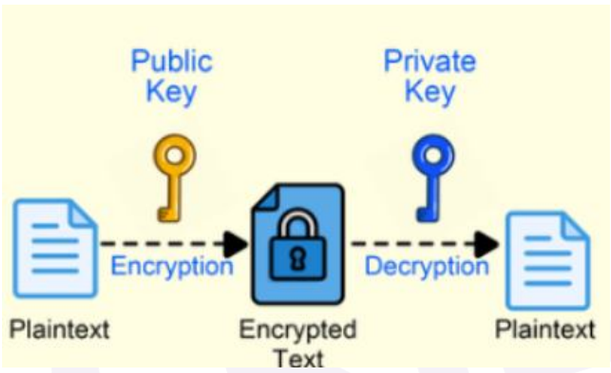
Encryption is the process of converting data into a coded format to prevent unauthorized access, ensuring data privacy and security.

Types of Encryption Algorithms

- **Symmetric Encryption:** Uses a single key for both encryption and decryption (e.g., AES).



Asymmetric Encryption: Uses public and private key pairs (e.g., RSA).



Introduction to SSL/TLS

What is SSL/TLS?

- SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols.
- Ensure secure communication over a computer network.
- Essential for securing data transmission over the internet.

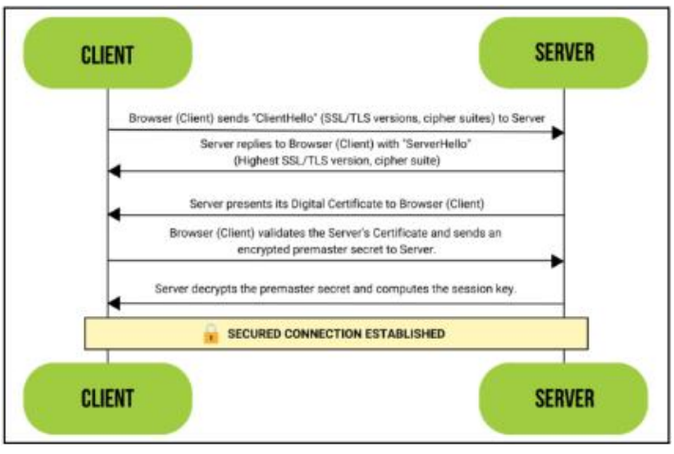
Purpose of SSL/TLS

- **Encryption:** Encrypts data to prevent unauthorized access during transmission.
- **Authentication:** Verifies the identity of parties involved in the communication.
- **Integrity:** Ensures data remains intact and unaltered during transmission.
- **Trust:** Establishes trust between communicating parties through certificates.

SSL/TLS Handshake Process

Overview

- **Client Hello:** Initiates the handshake by sending supported cryptographic algorithms.
- **Server Hello:** Responds with selected cryptographic algorithms and server certificate.
- **Key Exchange:** Client and server agree on session keys for encryption.
- **Encryption:** Secure data transmission begins using agreed-upon keys.

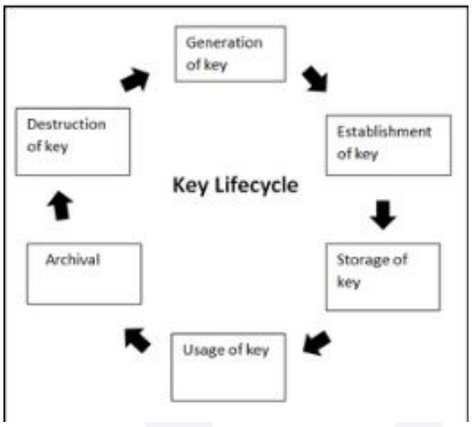


Components of SSL/TLS

Key Components

- **Certificates:** Digital certificates issued by Certificate Authorities (CAs) to verify identities.
- **Cipher Suites:** Sets of encryption algorithms and key exchange methods.
- **Protocols:** Versions such as SSL 3.0, TLS 1.0, TLS 1.2, TLS 1.3 (latest).

Certificates and PKI (Public Key Infrastructure):



Question Time

Question: What are the two main types of encryption used to ensure secure communication?
Question: What is the purpose of the SSL/TLS handshake protocol in secure communication?

Overview of OWASP

What is OWASP?

- Open Web Application Security Project (OWASP) is a nonprofit foundation.
- **Mission:** Improving software security by providing free resources and tools.
- **Community:** Global community of security professionals and volunteers.

OWASP Top 10

- **OWASP Top 10:** List of top 10 web application security risks.
- **Updated Regularly:** Reflects current security threats and vulnerabilities.
- **Guidance:** Provides guidance on mitigating risks in software development.

Lists the top ten web application security risks.

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Broken Access Control
- Security Misconfiguration
- Cross-Site Scripting (XSS)
- Insecure Deserialization
- Using Components with Known Vulnerabilities
- Insufficient Logging and Monitoring

OWASP Projects

Diverse Initiatives

- **Tools:** Security testing tools and frameworks.
- **Guides:** Development and deployment guides.
- **Training:** Online and offline training resources.
- **Research:** Research papers and documentation.

Question Time

Question: What is the OWASP Top 10, and why is it important?
Question: Give an example of an OWASP Top 10 security risk and a prevention method.

An Introduction to Data Protection Laws

What is Data Protection?

- **Definition:** Ensuring the privacy, confidentiality, and security of personal data.
- **Importance:** Protects individuals' rights and promotes trust in digital transactions.

Why Data Protection Laws Matter

- **Privacy Rights:** Safeguards individuals' privacy and personal information.
- **Compliance:** Ensures organizations handle data responsibly and legally.
- **Trust and Reputation:** Builds trust with customers, clients, and stakeholders.

General Data Protection Regulation (GDPR)

Overview

- **EU Regulation:** Enforced in May 2018.
- **Key Principles:** Lawful processing, data minimization, transparency, and accountability.
- **Rights:** Rights of data subjects (e.g., right to access, right to erasure).

Data Protection Laws in India

- **Personal Data Protection Bill, 2019:** Proposed comprehensive legislation.
- **Key Features:** Data localization, consent, rights of data subjects.
- Purpose: Enhance protection of personal data and regulate data processing.

Key Features of Personal Data Protection Bill, 2019

- Proposed Legislation
- **Data Localization:** Storage of personal data within India.
- **Consent:** Clear and informed consent for data processing.
- **Rights of Data Subjects:** Similar to GDPR rights (e.g., right to access, right to erasure).
- **Penalties:** Fines for non-compliance and data breaches

Question Time

Question: What is the main objective of data protection laws?
Question: Name two key data protection regulations and describe their focus.

Web Vulnerabilities

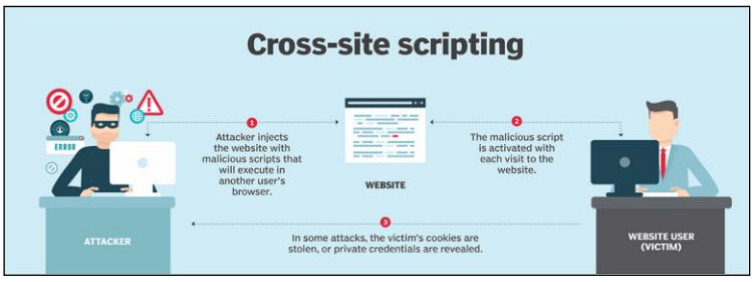
What are Web Vulnerabilities?

Definition: Weaknesses or loopholes in web applications that can be exploited by attackers.
Importance: Pose security risks such as unauthorized access, data breaches, and service disruptions.
.

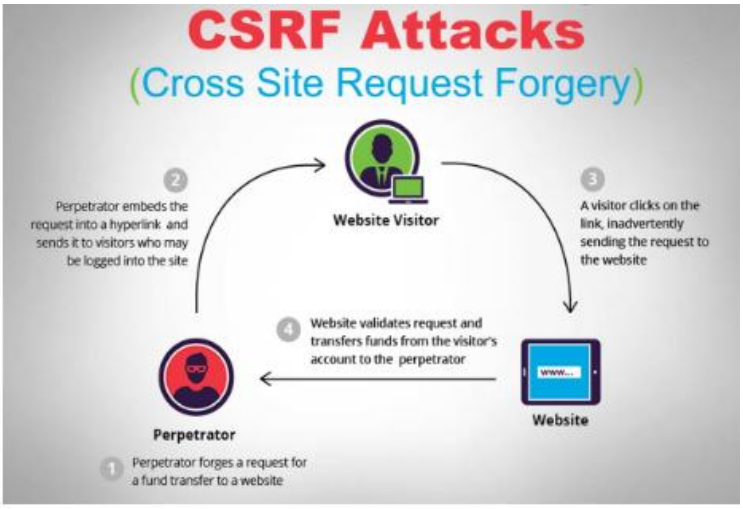
Common Web Vulnerabilities

- **Injection Attacks:** SQL Injection, Command Injection.
- **Broken Authentication:** Weak credentials, session management flaws.
- **Sensitive Data Exposure:** Insecure storage or transmission of sensitive information.
- **XML External Entities (XXE):** Exploiting XML parsers.
- **Broken Access Control:** Improper access restrictions.
- **Security Misconfiguration:** Default configurations, unnecessary features.
- **Cross-Site Scripting (XSS):** Injection of malicious scripts.
- **Insecure Deserialization:** Manipulation of serialized data.
- **Using Components with Known Vulnerabilities:** Exploiting outdated libraries.
- **Insufficient Logging and Monitoring:** Inadequate visibility into security events.

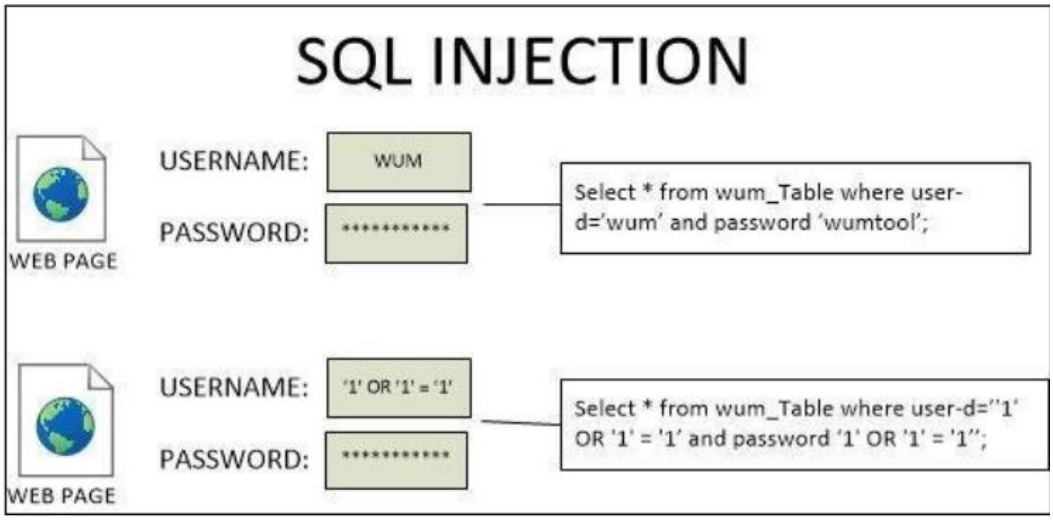
Cross-site scripting : Cross-site scripting (XSS) is a security vulnerability that allows attackers to inject malicious scripts into webpages viewed by other users.



CSRF (Cross-Site Request Forgery) is a security vulnerability where an attacker tricks a user into performing actions on a website without their consent.



SQL injection is a security vulnerability that allows attackers to manipulate a website's database by injecting malicious SQL code through input fields.



Importance of Web Application Security

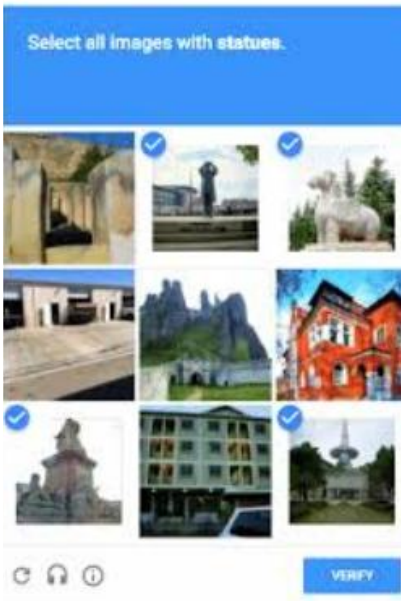
- **Protects Users:** Safeguards personal and financial information.
- **Maintains Trust:** Enhances user trust and credibility.
- **Compliance:** Meets regulatory requirements (e.g., GDPR, PCI DSS).

Question Time
Question: What is Cross-Site Scripting (XSS) and how does it exploit web applications?

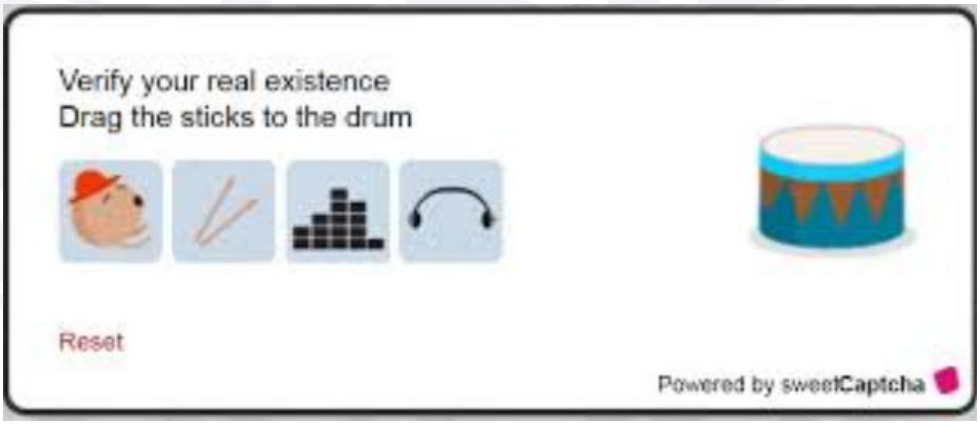
CAPTCHA

CAPTCHA is a security measure that distinguishes between humans and bots. It uses challenges easy for humans but hard for automated scripts, ensuring user authenticity.
Text-based CAPTCHA is a security measure that requires users to enter characters from a distorted image to prove they are human and not automated bots.

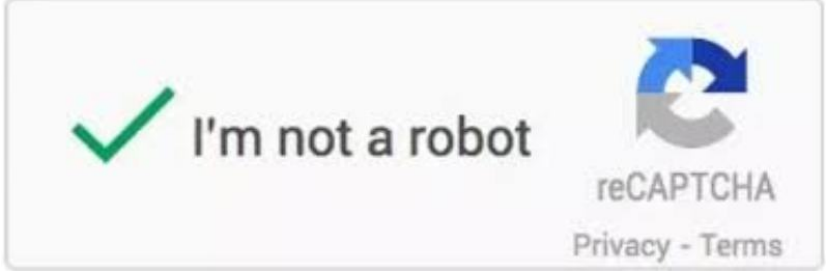
Image-based CAPTCHA is a security mechanism that requires users to identify and select specific objects or patterns within an image to verify their identity as human users rather than automated bots.



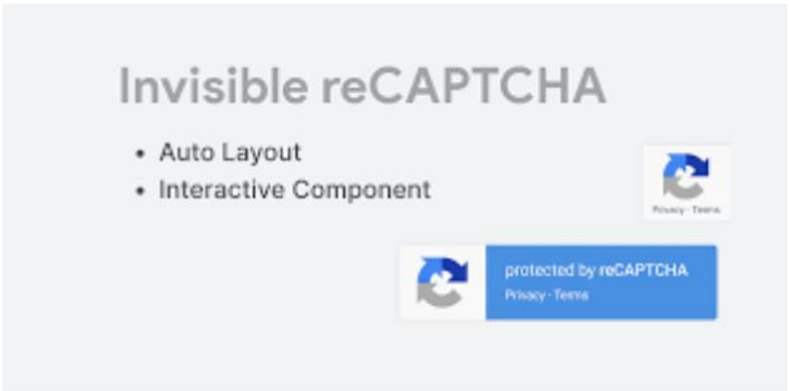
Logical CAPTCHA typically involves presenting users with a challenge that requires logical reasoning or understanding of a concept rather than visual identification. It may ask users to solve a puzzle, answer a simple question, or perform a task that requires human intelligence to complete, thereby verifying their identity as a human user.



Checkbox CAPTCHA is a type of CAPTCHA that presents users with a checkbox to confirm their identity as a human user. Users simply need to check the box to indicate they are not a robot, eliminating the need for complex image or text recognition tasks.



Invisible CAPTCHA is a type of CAPTCHA that verifies users without requiring them to solve any interactive puzzles or challenges. It operates in the background and detects bot activities based on user behavior, such as mouse movements and interactions with the webpage, providing a seamless user experience while maintaining security.



Working of CAPTCHA

CAPTCHA Mechanism

- **Challenge:** Presents a task that is easy for humans but difficult for bots.
- **Response:** User completes the task to prove human identity.
- **Validation:** Server verifies the response to grant access.

Alternatives to CAPTCHA

Other Approaches

- **Biometric Verification:** Fingerprint, facial recognition.
- **Behavioral Analysis:** Mouse movements, typing patterns.
- **Two-Factor Authentication (2FA):** SMS codes, authenticator apps.

Question Time
Question: What is CAPTCHA, and how does it distinguish between humans and bots?

Interview points:

Q1 : What are the main challenges associated with password-based authentication? How can these challenges be mitigated?

Q2 : Explain the difference between Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC). When would you choose one over the other?

Q3 : Why is input validation and sanitization crucial for web application security, and how do they work together to protect against attacks like SQL injection and Cross-Site Scripting (XSS)?

Q4 : What are the potential security implications of improperly configured Cross-Origin Resource Sharing (CORS) policies?

Q5 : How does rate limiting differ from throttling when managing API requests, and why are both techniques essential for API security and performance?

Q6 : What are the key differences between symmetric and asymmetric encryption, and when would you choose one over the other in a secure communication setup?

Q7 : Can you explain the role of digital certificates and Public Key Infrastructure (PKI) in ensuring secure communication over the Internet?

Q8 : What are the key differences between the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA), and how do these regulations impact organizations differently?

Q9 : Describe SQL Injection and explain how it can be prevented in web applications.

Q10 : Different types of CAPTCHA