Web Application Penetration Testing

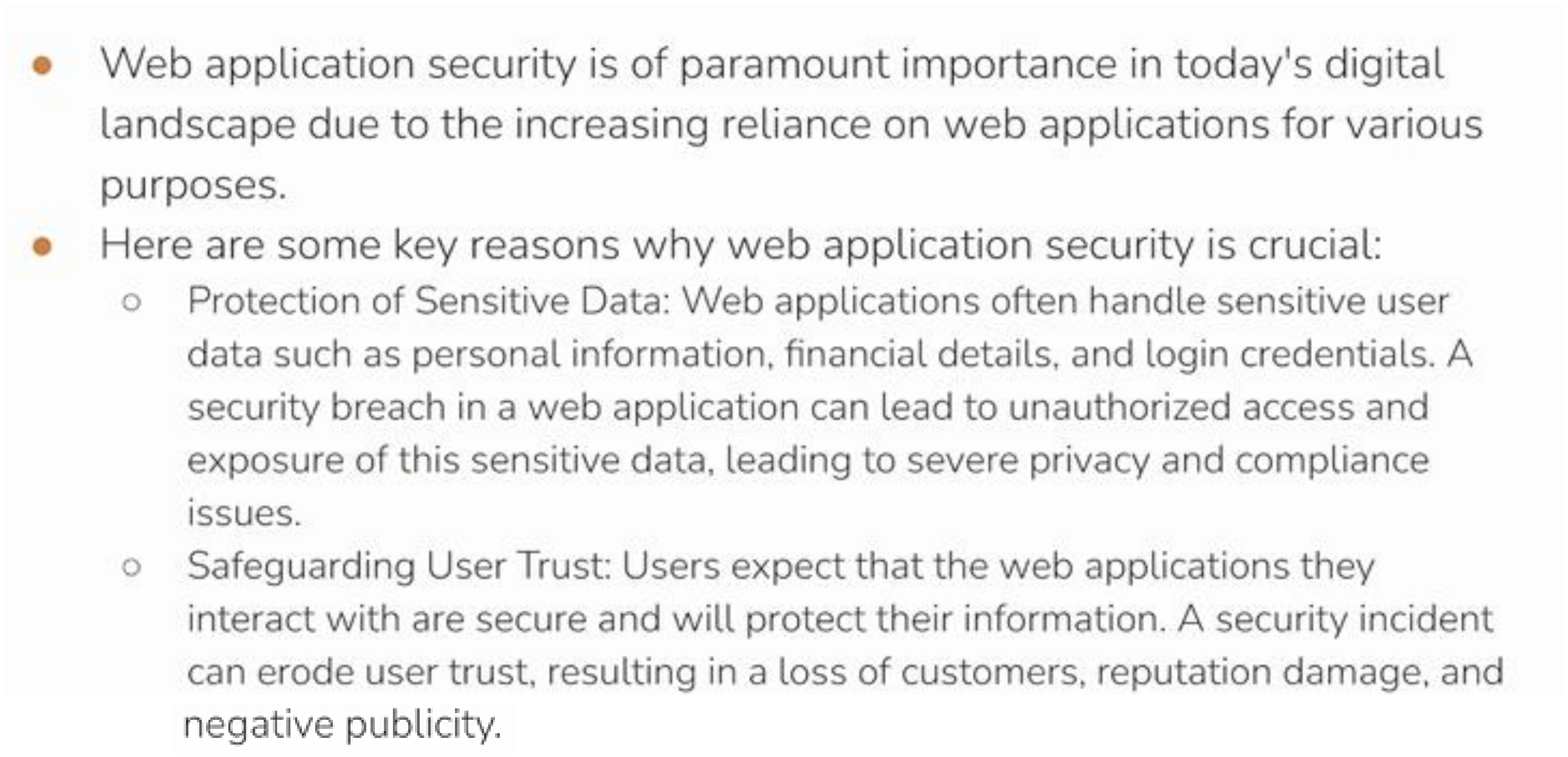
Bappe Sarker

Web Application Fundamentals

What is Web Applications?

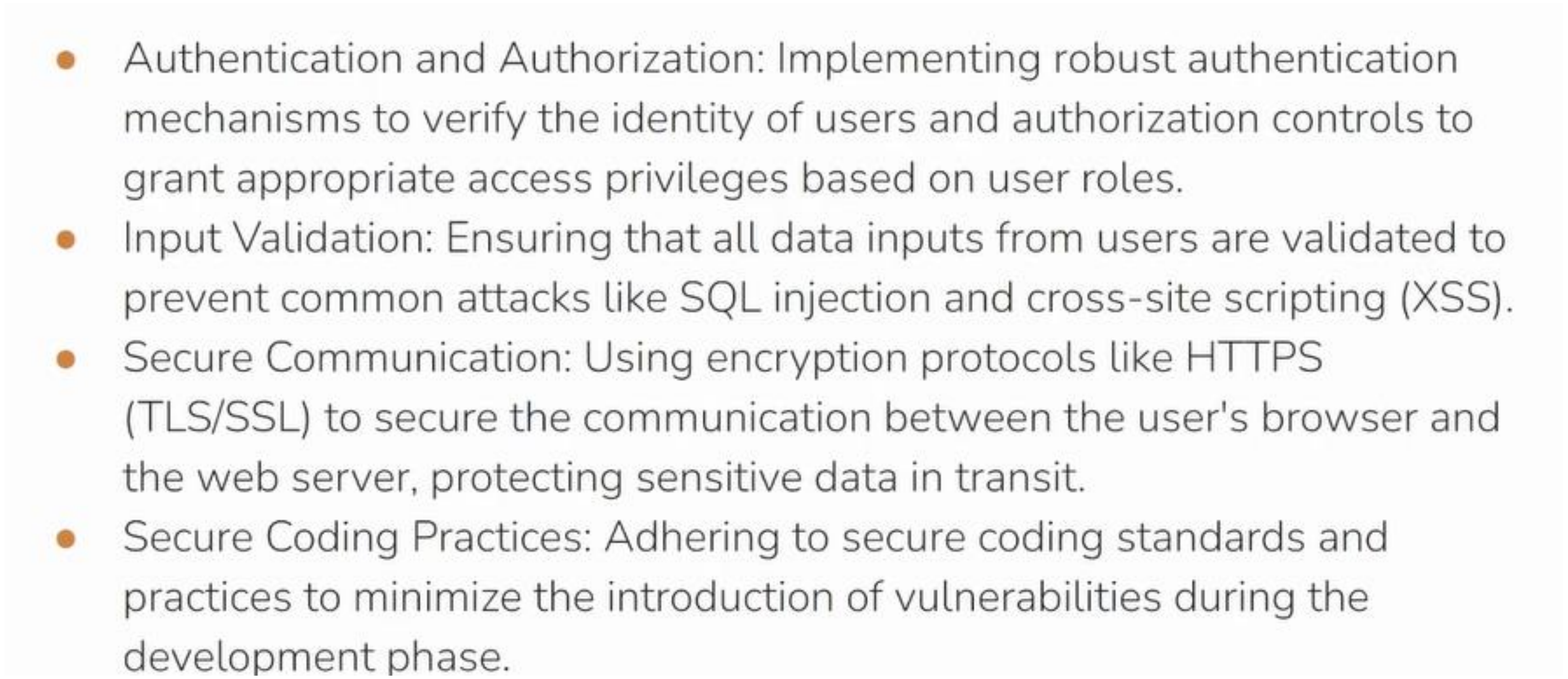
- Web applications are software programs that run on web servers and are accessible over internet through web browsers.
- They are designed to provide integrative and dynamic functionality to users, allowing them to perform various tasks, access information and interact with data online.
- Examples: Online email services (e.g.. Gmail, Outlook)

Why web application security?



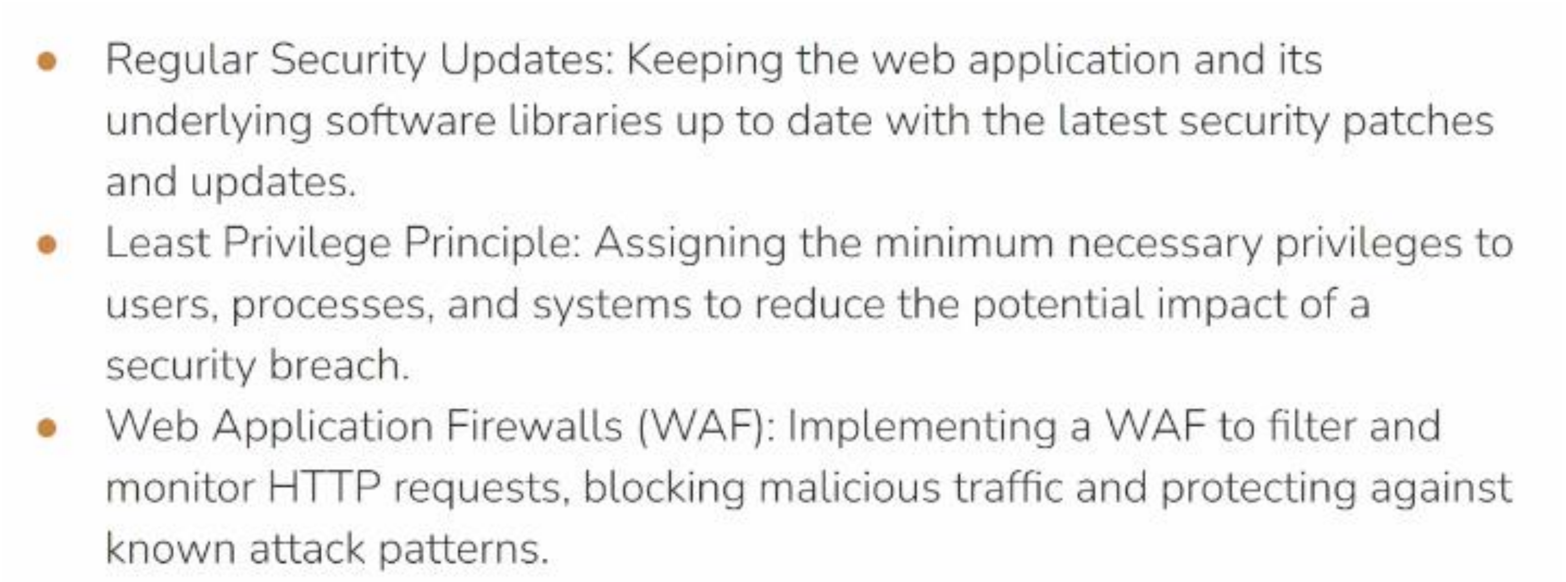
- Web application security is of paramount importance in today's digital landscape due to the increasing reliance on web applications for various purposes.
- Here are some key reasons why web application security is crucial:
    - Protection of Sensitive Data: Web applications often handle sensitive user data such as personal information, financial details, and login credentials. A security breach in a web application can lead to unauthorized access and exposure of this sensitive data, leading to severe privacy and compliance issues.
    - Safeguarding User Trust: Users expect that the web applications they interact with are secure and will protect their information. A security incident can erode user trust, resulting in a loss of customers, reputation damage, and negative publicity.

Web application security practices



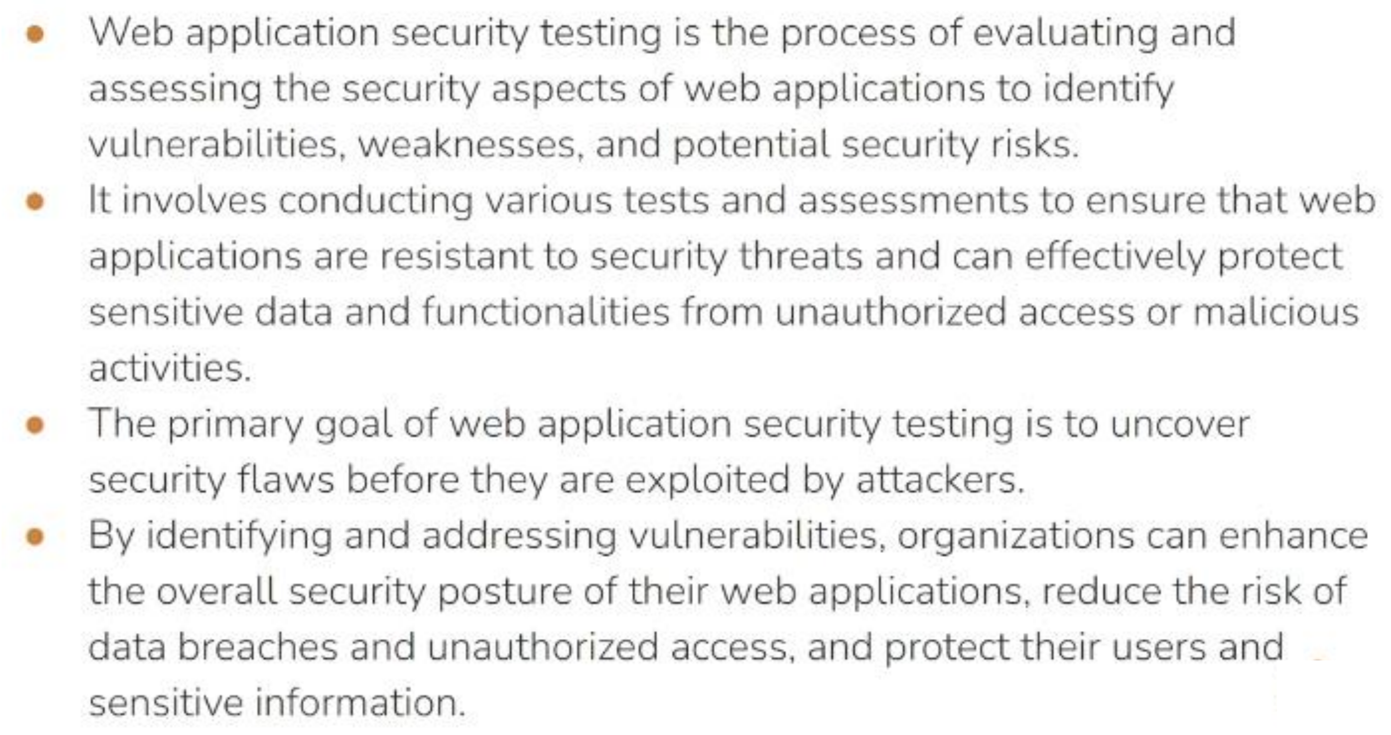
- Authentication and Authorization: Implementing robust authentication mechanisms to verify the identity of users and authorization controls to grant appropriate access privileges based on user roles.
- Input Validation: Ensuring that all data inputs from users are validated to prevent common attacks like SQL injection and cross-site scripting (XSS).
- Secure Communication: Using encryption protocols like HTTPS (TLS/SSL) to secure the communication between the user's browser and the web server, protecting sensitive data in transit.
- Secure Coding Practices: Adhering to secure coding standards and practices to minimize the introduction of vulnerabilities during the development phase.

Web application security practices



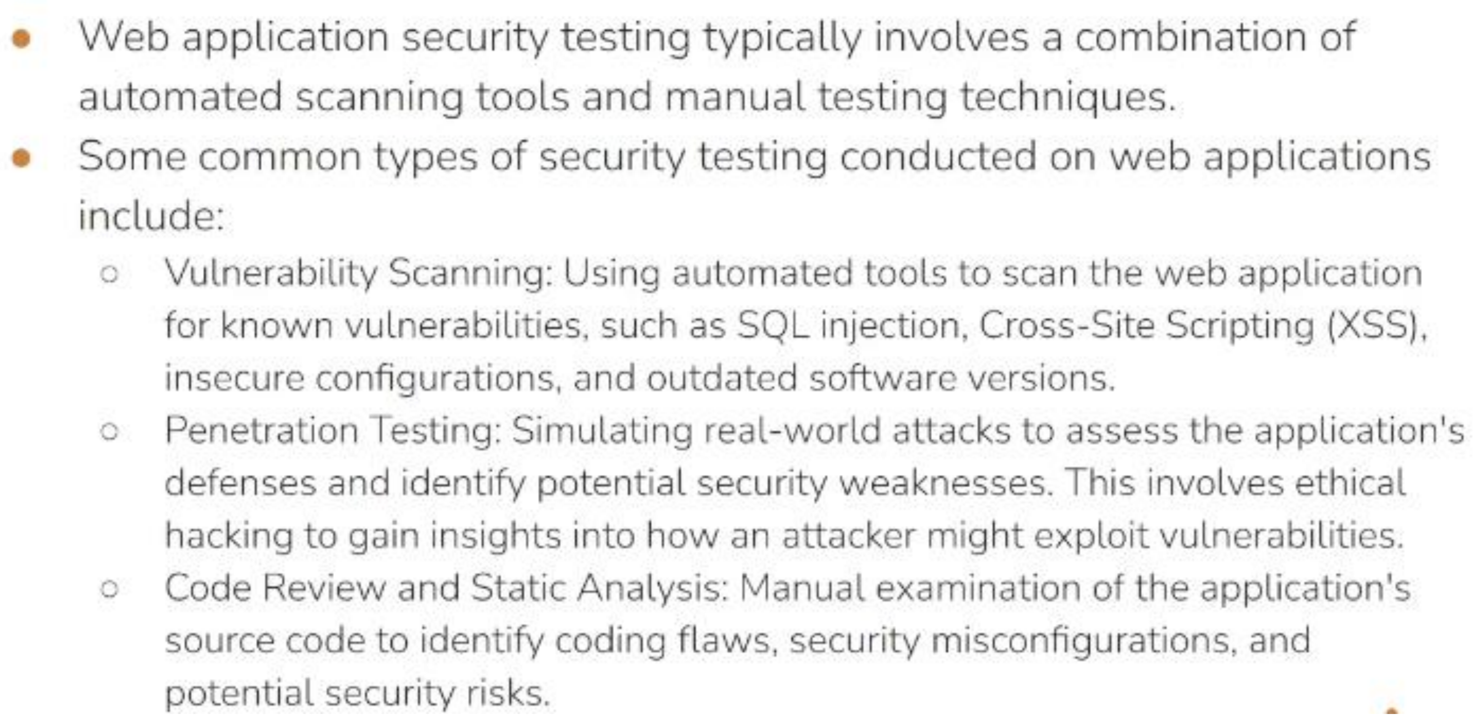
- Regular Security Updates: Keeping the web application and its underlying software libraries up to date with the latest security patches and updates.
- Least Privilege Principle: Assigning the minimum necessary privileges to users, processes, and systems to reduce the potential impact of a security breach.
- Web Application Firewalls (WAF): Implementing a WAF to filter and monitor HTTP requests, blocking malicious traffic and protecting against known attack patterns.

Web application security Testing



- Web application security testing is the process of evaluating and assessing the security aspects of web applications to identify vulnerabilities, weaknesses, and potential security risks.
- It involves conducting various tests and assessments to ensure that web applications are resistant to security threats and can effectively protect sensitive data and functionalities from unauthorized access or malicious activities.
- The primary goal of web application security testing is to uncover security flaws before they are exploited by attackers.
- By identifying and addressing vulnerabilities, organizations can enhance the overall security posture of their web applications, reduce the risk of data breaches and unauthorized access, and protect their users and sensitive information.

Web application security Testing Types



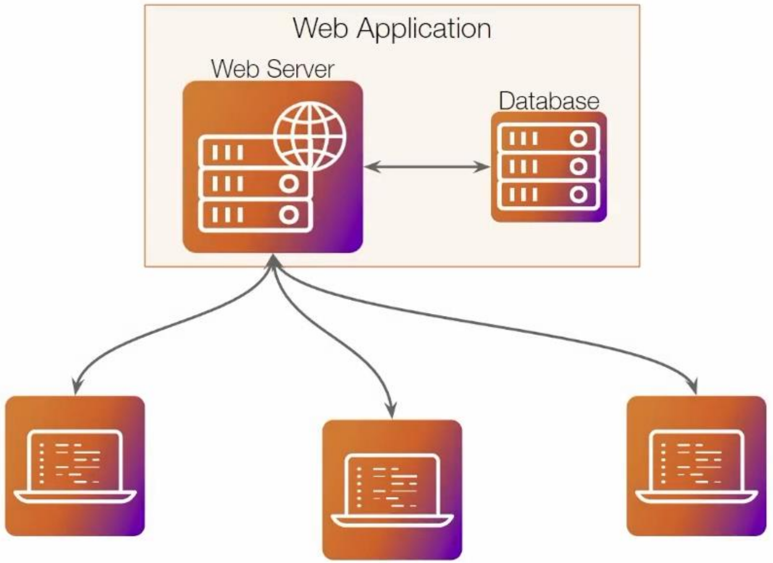
- Web application security testing typically involves a combination of automated scanning tools and manual testing techniques.
- Some common types of security testing conducted on web applications include:
  - Vulnerability Scanning: Using automated tools to scan the web application for known vulnerabilities, such as SQL injection, Cross-Site Scripting (XSS), insecure configurations, and outdated software versions.
  - Penetration Testing: Simulating real-world attacks to assess the application's defenses and identify potential security weaknesses. This involves ethical hacking to gain insights into how an attacker might exploit vulnerabilities.
  - Code Review and Static Analysis: Manual examination of the application's source code to identify coding flaws, security misconfigurations, and potential security risks.

Web application security Testing Types

- Authentication and Authorization Testing: Evaluating the effectiveness of authentication mechanisms and access control features to ensure that only authorized users have appropriate access levels.
- Input Validation and Output Encoding Testing: Assessing how the application handles user inputs to prevent common security vulnerabilities like XSS and SQL injection.
- Session Management Testing: Verifying how the application manages user sessions and related tokens to prevent session-related attacks.
- API Security Testing: Assessing the security of APIs (Application Programming Interfaces) used by the web application for data exchange and integration with other systems.

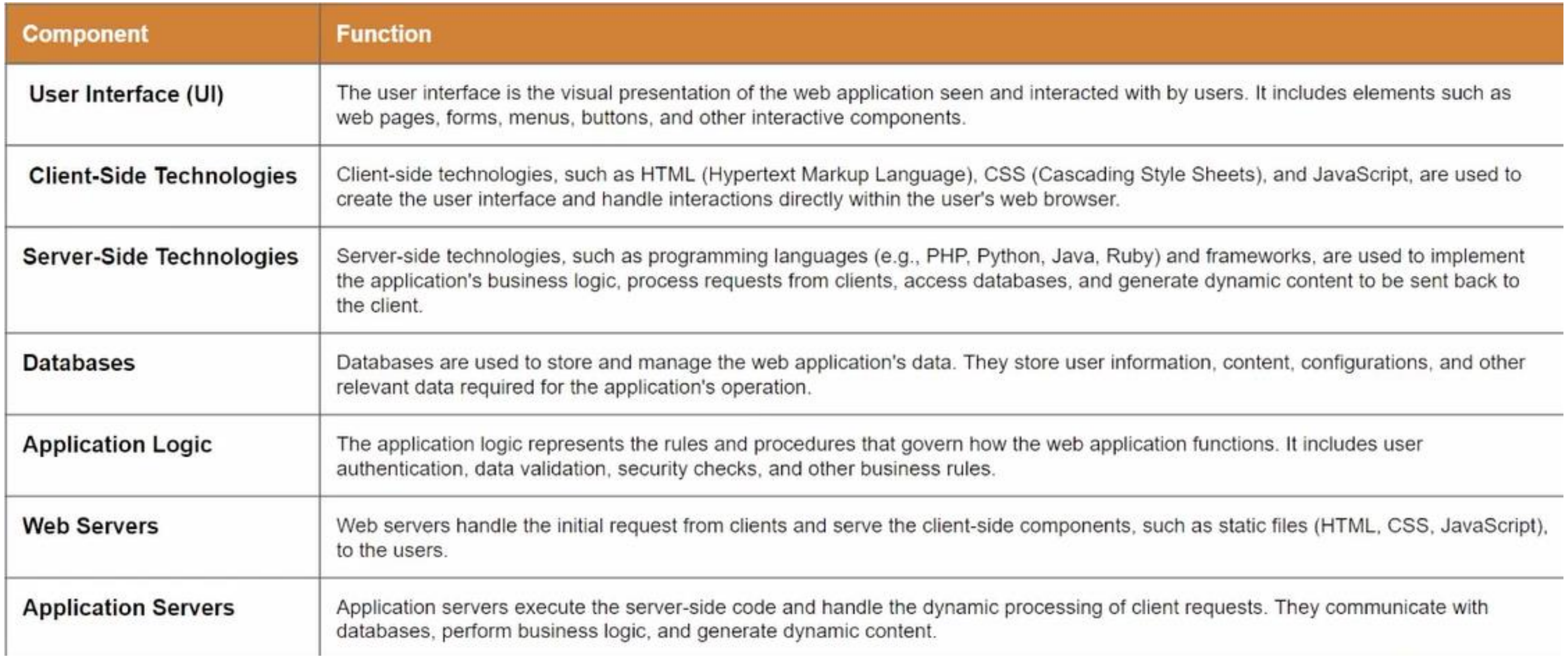
Web Application Fundamentals

- Web application architecture model: Web application can be deployed in various ways like:
  - Client-Server Model



Web Application Fundamentals

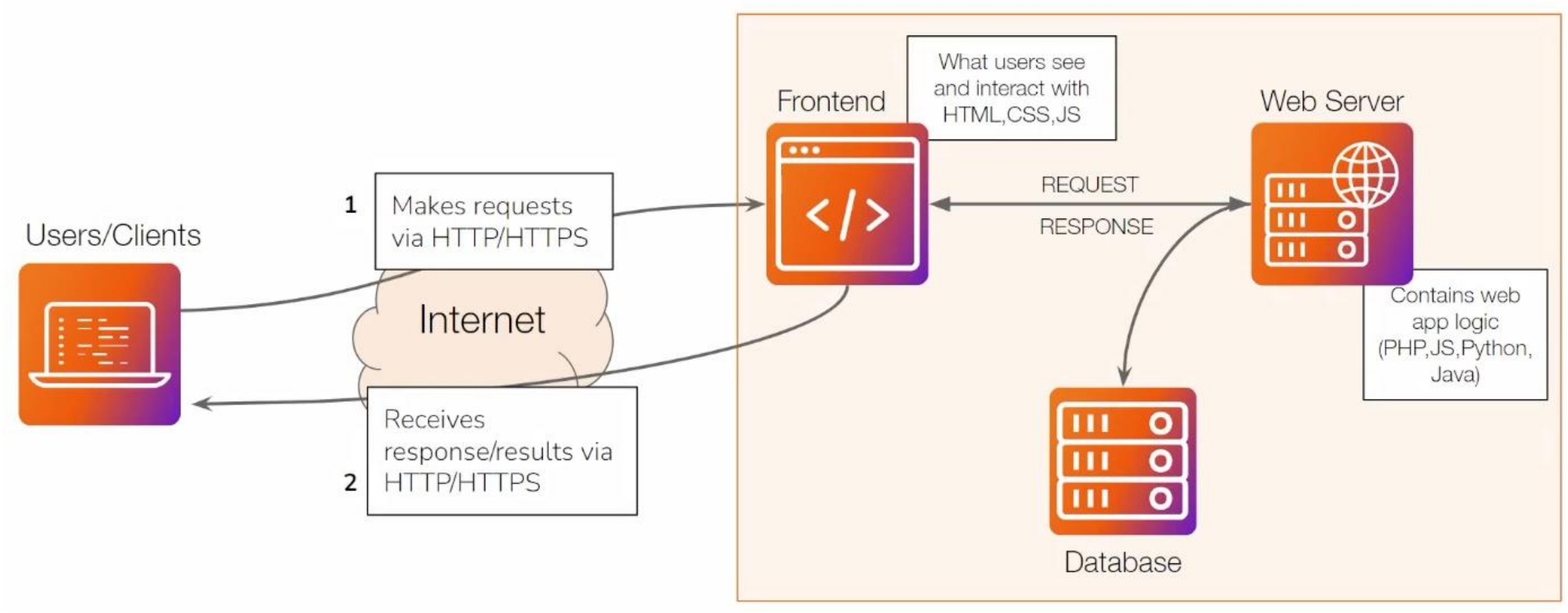
- Web application components



Component	Function
User Interface (UI)	The user interface is the visual presentation of the web application seen and interacted with by users. It includes elements such as web pages, forms, menus, buttons, and other interactive components.
Client-Side Technologies	Client-side technologies, such as HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript, are used to create the user interface and handle interactions directly within the user's web browser.
Server-Side Technologies	Server-side technologies, such as programming languages (e.g., PHP, Python, Java, Ruby) and frameworks, are used to implement the application's business logic, process requests from clients, access databases, and generate dynamic content to be sent back to the client.
Databases	Databases are used to store and manage the web application's data. They store user information, content, configurations, and other relevant data required for the application's operation.
Application Logic	The application logic represents the rules and procedures that govern how the web application functions. It includes user authentication, data validation, security checks, and other business rules.
Web Servers	Web servers handle the initial request from clients and serve the client-side components, such as static files (HTML, CSS, JavaScript), to the users.
Application Servers	Application servers execute the server-side code and handle the dynamic processing of client requests. They communicate with databases, perform business logic, and generate dynamic content.

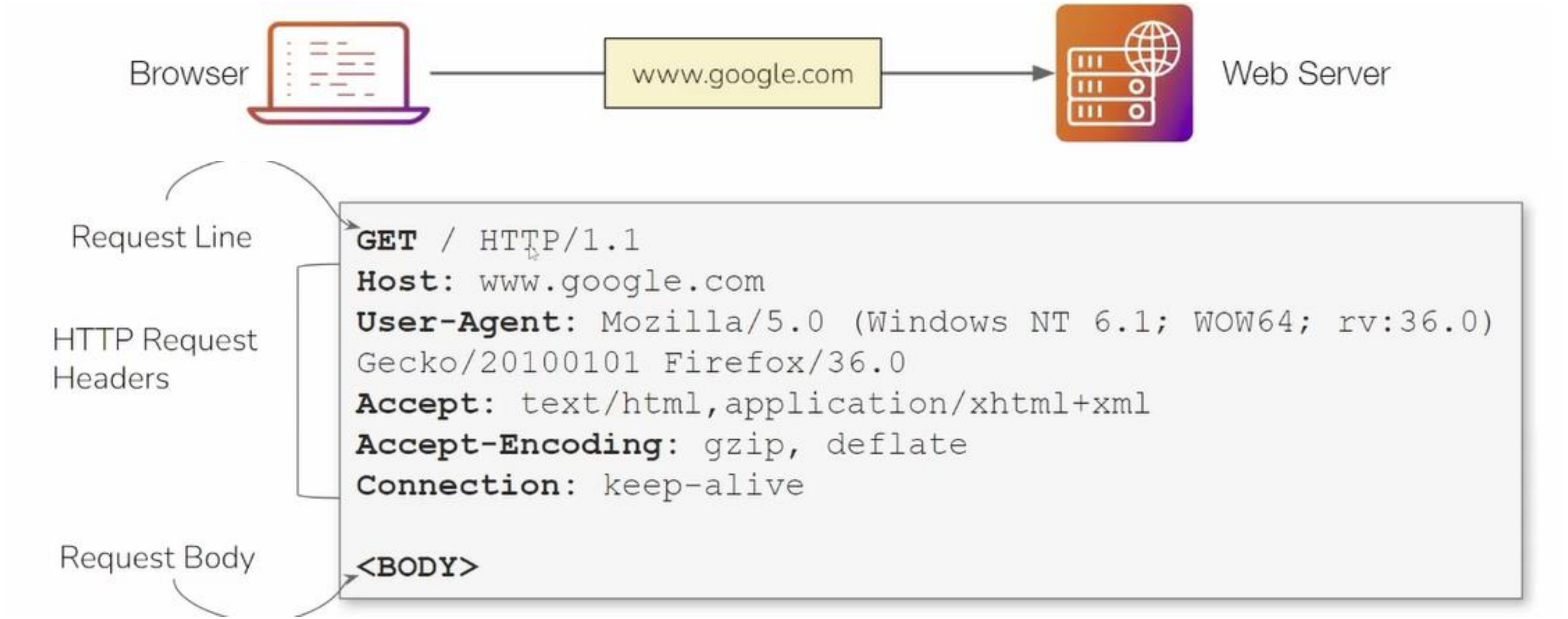
Web Application Fundamentals

- Web application components


Users/Clients
1
Makes requests via HTTP/HTTPS
Internet
2
Receives response/results via HTTP/HTTPS
Frontend
What users see and interact with HTML,CSS,JS
REQUEST
RESPONSE
Web Server
Contains web app logic (PHP,JS,Python, Java)
Database

HTTP Protocol

HTTP Request:

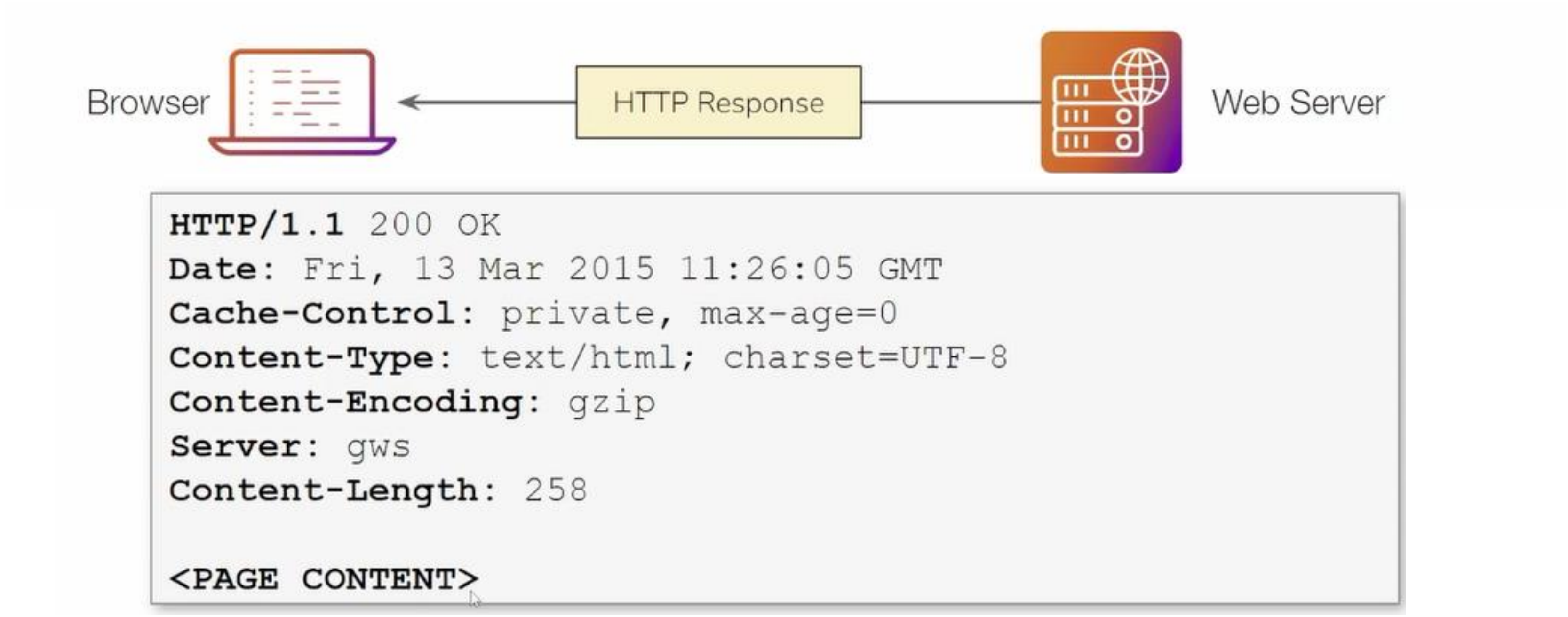


```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:36.0)
Gecko/20100101 Firefox/36.0
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate
Connection: keep-alive

<BODY>
```

HTTP Protocol

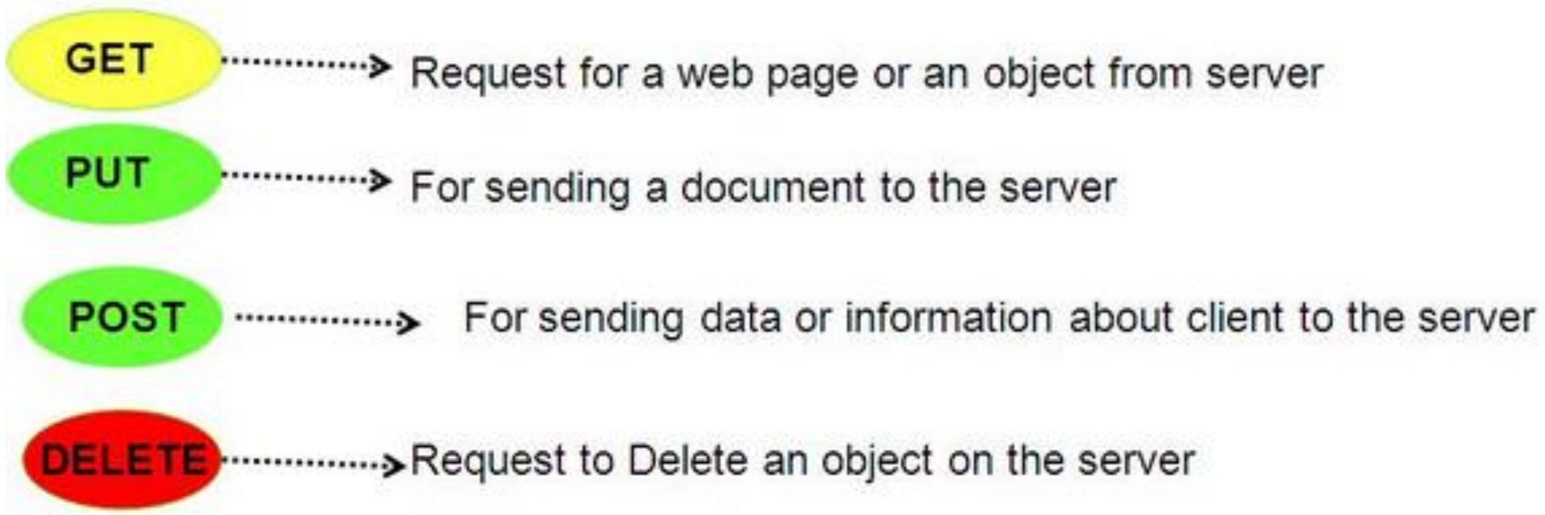
HTTP Response:



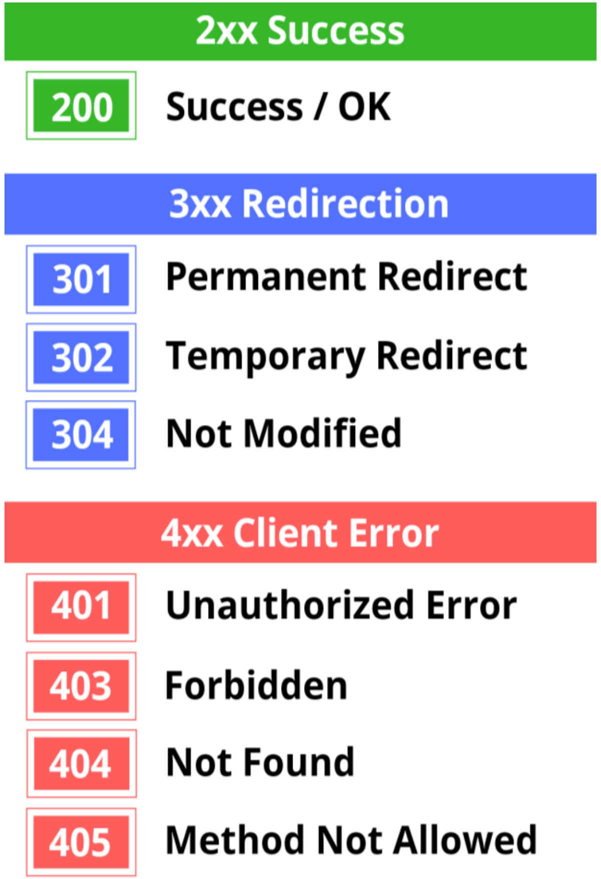
```
HTTP/1.1 200 OK
Date: Fri, 13 Mar 2015 11:26:05 GMT
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 258

<PAGE CONTENT>
```

HTTP Methods


GET
Request for a web page or an object from server
PUT
For sending a document to the server
POST
For sending data or information about client to the server
DELETE
Request to Delete an object on the server

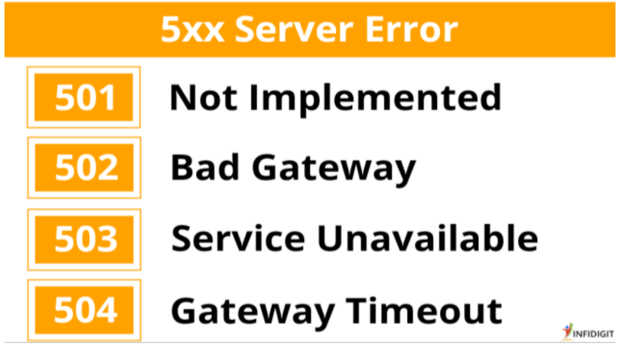
HTTP Status Code



2xx Success	
200	Success / OK

3xx Redirection	
301	Permanent Redirect
302	Temporary Redirect
304	Not Modified

4xx Client Error	
401	Unauthorized Error
403	Forbidden
404	Not Found
405	Method Not Allowed

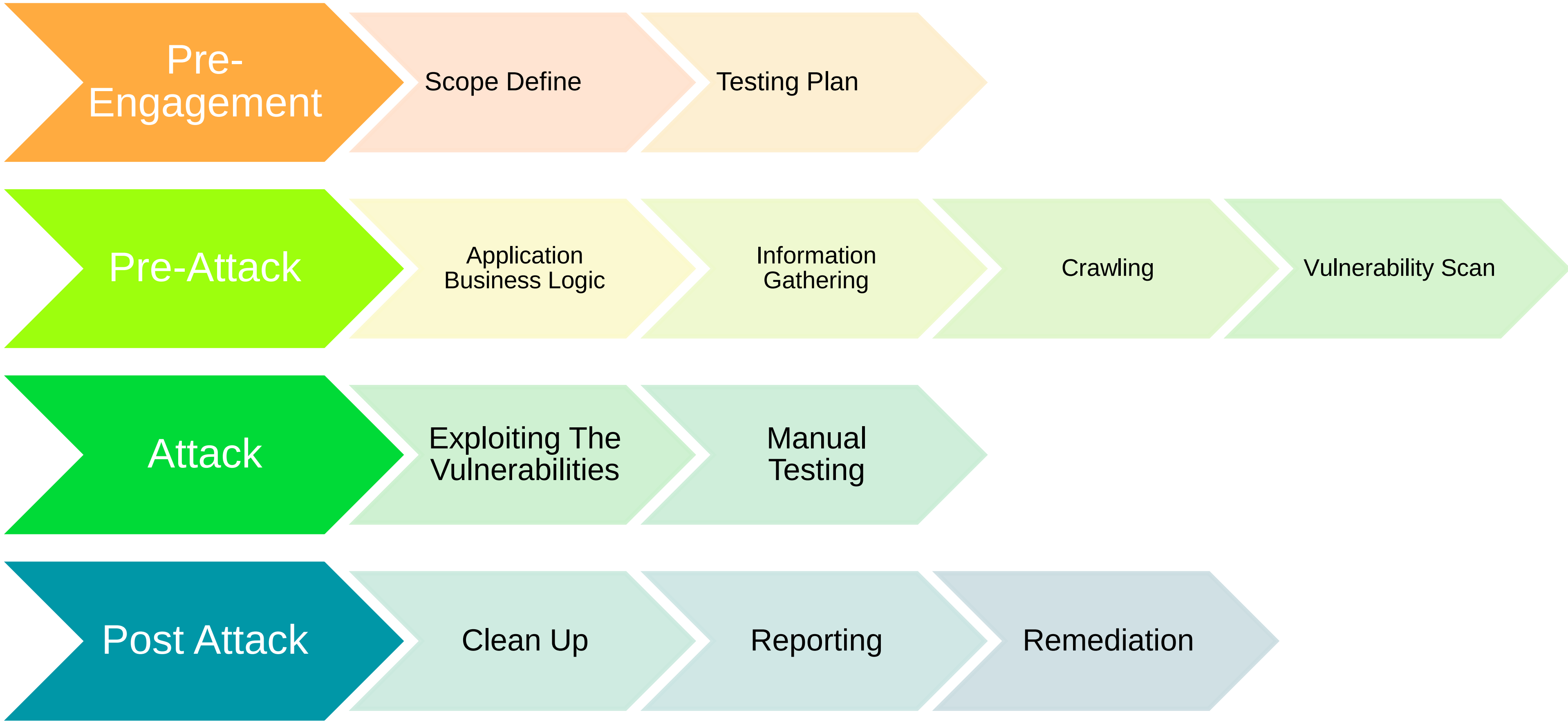


5xx Server Error	
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

Web Application Penetration Test

- Web application penetration testing is a comprehensive security assessment aimed at identifying vulnerability and weaknesses in web applications.
- It involves simulating real-world attacks to evaluate the application’s security posture and provide recommendations for remediation.
- Using a methodology for web application penetration testing is vitally important as it provides a structured and systematic approach to conduct through security assessment.

Web Application Hacking Life Cycle



Assessment Step

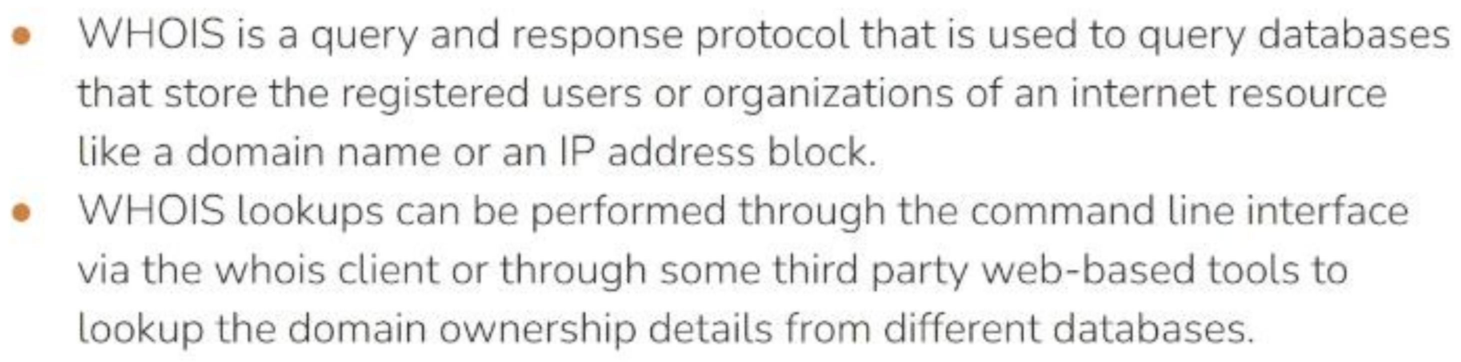
Phase	Target	Tools
Information Gathering	Whois lookup, Banner Grabbing, Sub-Domain Enumeration, Specific URLs, HTTP Options	Whois tool, whatweb / Wapalyzer, Sublist3r, amass, Google Dorks, Burp Suite
Crawling / Spidering	Directory discover, File discovery, Web site structure	Dirbuster, Gobuster,Drib, OWASP Zap, Burp Suite
Security Assessment / Audit	Vulnerability Assessment	Tools- Netsparker, Acunetix Checklist- OWASP/SANS
Penetration Test	Exploitation	Burp Suite, SQLmap, Open Source Toolset.

Information Gathering

Information gathering is a process to collects those information which helps to identify the weakness or exploit the application vulnerabilities. Information gathering includes:

- Finding the website ownership and IP address
- Search Engine Discovery
- Source code analysis
- Crawling and spidering
- Web server footprinting
- Domain and subdomain enumeration

Whois Lookup



- WHOIS is a query and response protocol that is used to query databases that store the registered users or organizations of an internet resource like a domain name or an IP address block.
- WHOIS lookups can be performed through the command line interface via the whois client or through some third party web-based tools to lookup the domain ownership details from different databases.

Tools – whois (kali linux command), Online whois lookup sites.

Objectives – Domain owner or IP owner details.

Finding real IP of a site

Real IP is important to perform some attacks, such as – DDoS attack.

Tools –

https://securitytrails.com/

https://search.censys.io/

Google dorks to find useful info

We can identify vulnerable location or vulnerable endpoints of an application or websites by using dorks.

Tools –

https://www.exploit-db.com/google-hacking-database

Banner Grabbing

We can search backend technologies of server or sites, such as – web server, language, frameworks etc.

Tools –

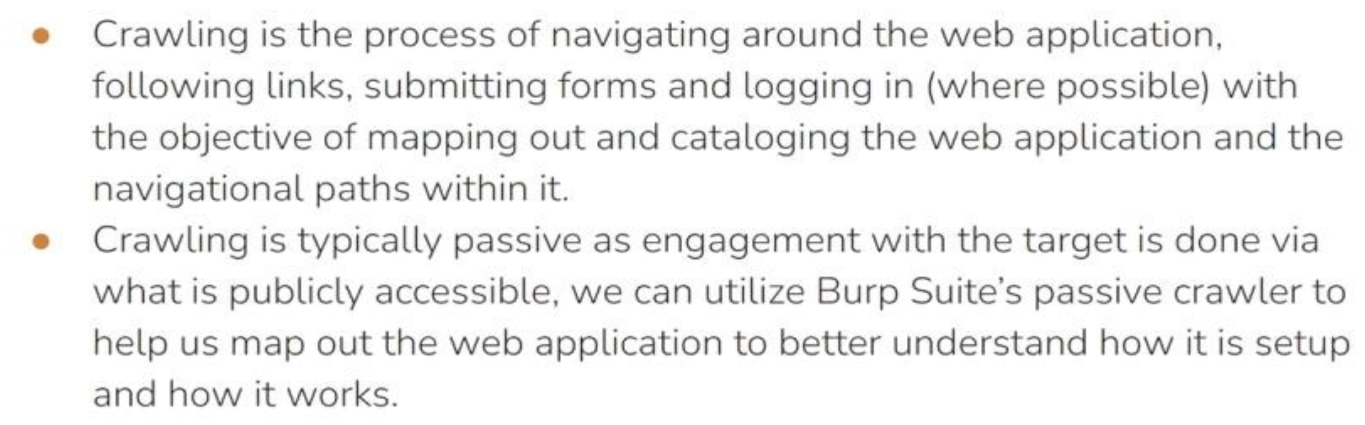
- Whatweb (kali)
- Wappalyzer(browser extension)

Web Server Footprinting

By performing web server footprinting, we can identify server’s open ports, running services, basic configuration etc.

Tools- Nmap, Nikto

Crawling and Spidering



- Crawling is the process of navigating around the web application, following links, submitting forms and logging in (where possible) with the objective of mapping out and cataloging the web application and the navigational paths within it.
- Crawling is typically passive as engagement with the target is done via what is publicly accessible, we can utilize Burp Suite's passive crawler to help us map out the web application to better understand how it is setup and how it works.

Tools –

○ Burp suite / ZAP

Domain / Subdomain Enumeration

We can perform domain enumeration to findout domain records and subdomains for target domain.

Tools- Amass, subrute, dig, dnsenum etc.

Hidden Directory Busting

We can discover hidden file or folder by using different directory bruteforce tools.

Tools- Gobuster, dirbuster, Feroxbuster etc.

OWASP Top 10 - 2021

OWASP Top 10

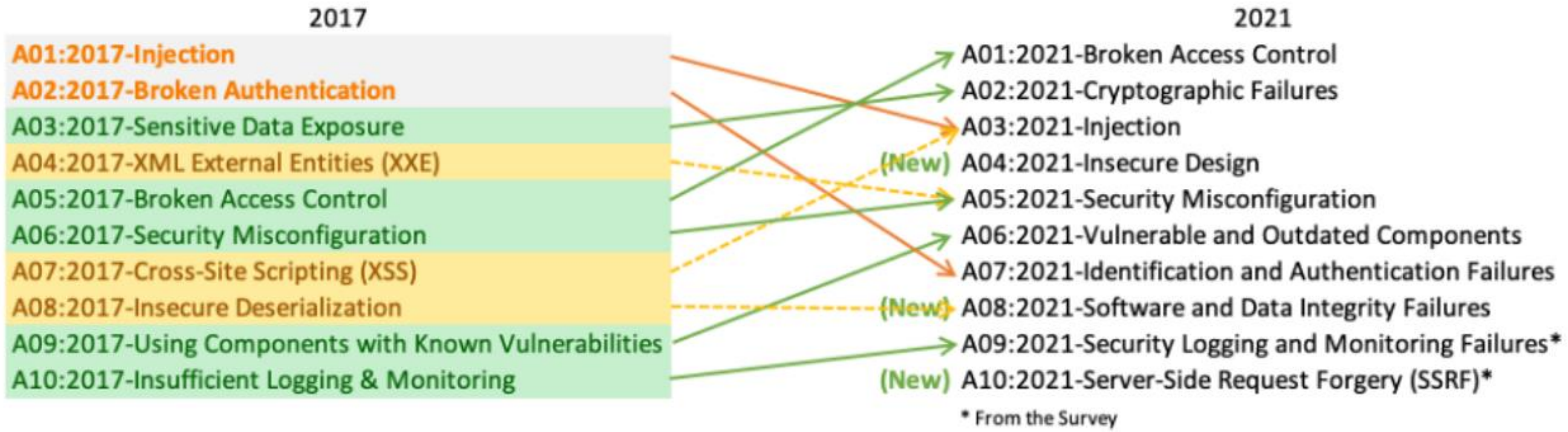
The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications

Globally recognized by developers as the first step towards more secure coding.

OWASP Top 10 over Time

Top 10 Web Application Security Risks

There are three new categories, four categories with naming and scoping changes, and some consolidation in the Top 10 for 2021.

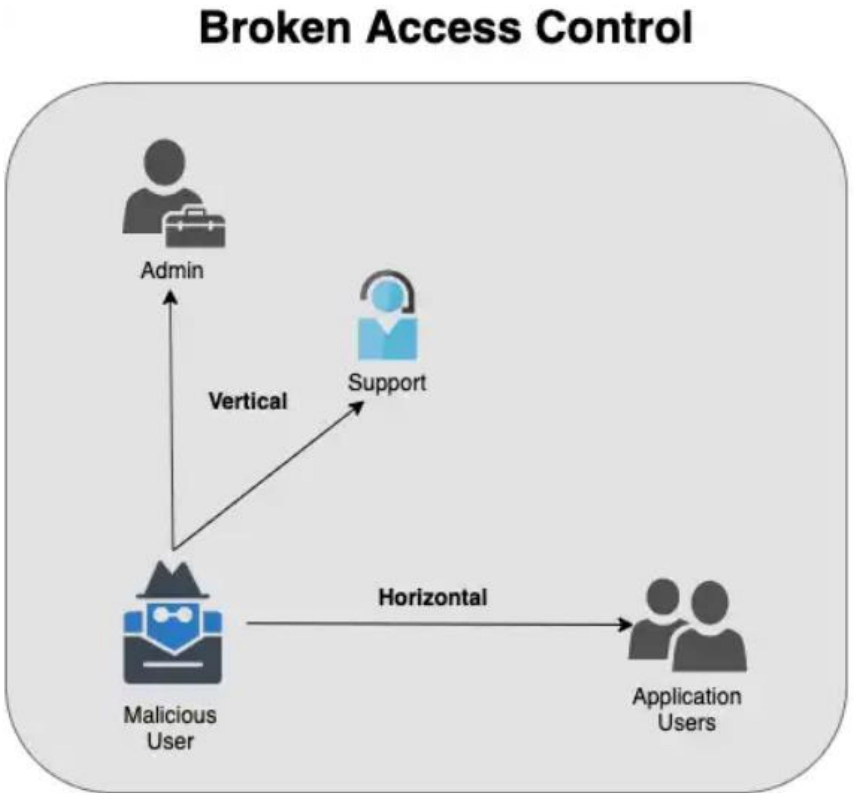

2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

How to Use OWASP top 10

- Understand the classes of web application vulnerabilities
- Prioritize remediation
- Educate application developer

A01:2021 – Broken Access Control

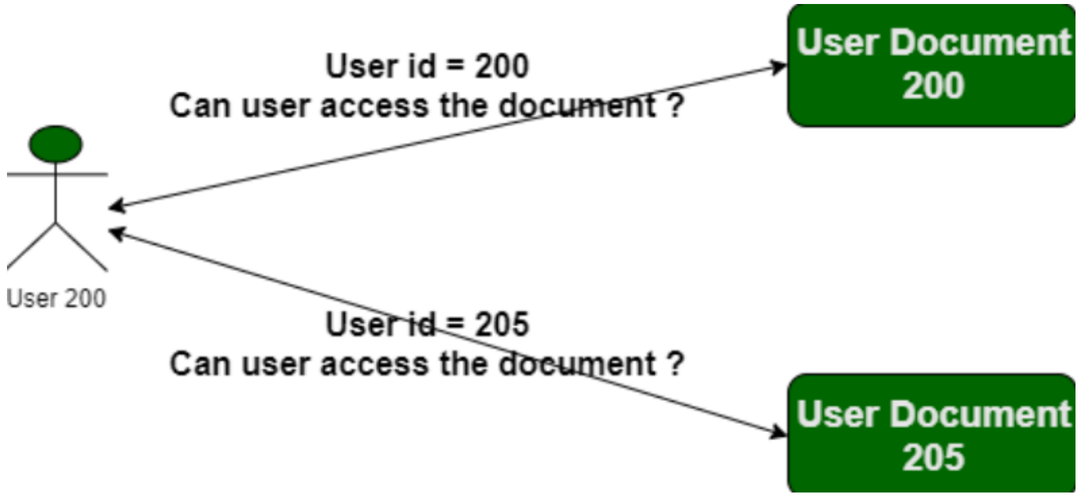
Access controls are designed to prevent users from acting outside their intended permissions. Users can take actions beyond the scope of their authorized permissions if there are vulnerabilities in these controls. This may allow attackers to steal information from other users, modify data and perform actions as other users. **Arises from too much trust in client side code.**



A01:2021 – Broken Access Control

Another types of broken access control is **IDOR (insecure direct object reference)**, When applications fail to secure access from a user’s scope.

Example – view another user profile information by changing the user id / profile id parameter.



A02:2021 – Cryptographic Failures

Cryptography offers tools that can be used to safeguard sensitive data and securely transfer it across the internet.

Any misuse or lack of cryptographic security solutions.

Example – HTTP instead of HTTPS, Weak password hashing.

A03:2021 - Injection Flaws

Injection flaws are web application vulnerabilities that allow **untrusted, malicious** or **arbitrary data** as input from **user or malicious actor**.

The main reason is **input validation/sanitization**.

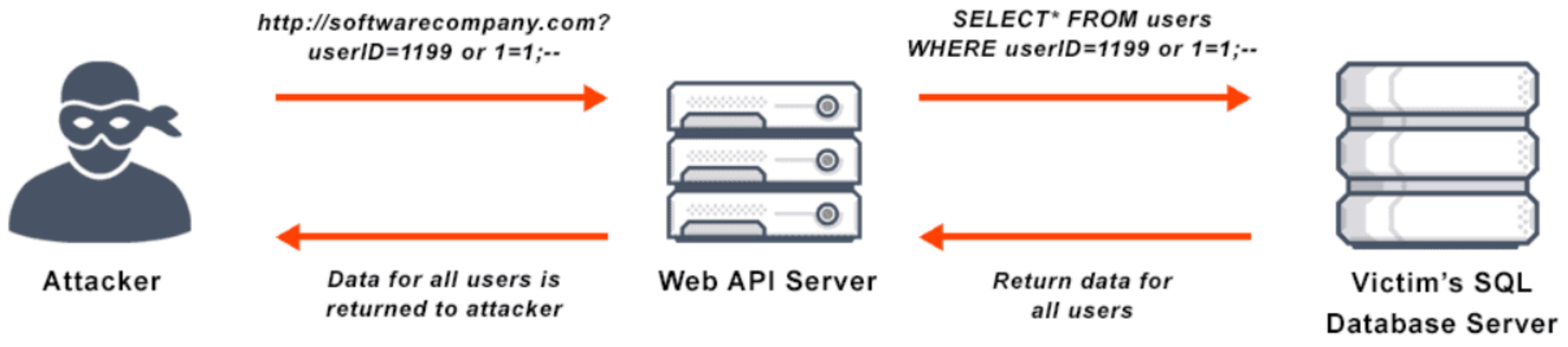
Hacker can exploit injection vulnerabilities by executing malicious commands or quires that is resulted as –

- SQL injection
- Command Injection
- Cross Site Scripting (XSS)
- LDAP Injection

SQL injection

SQL injection is a technique used to **retrieve, write or remove data** from **SQL database** via manipulating the application's backend **SQL queries** by input **malicious inputs**.

SQL injection is critical vulnerability in all time !


http://softwarecompany.com?
userID=1199 or 1=1;--
SELECT* FROM users
WHERE userID=1199 or 1=1;--
Attacker
Data for all users is returned to attacker
Web API Server
Return data for all users
Victim's SQL Database Server

SQL injection Location and Types

Using SQL injection, hacker can do –

- Read/write data in DB
- Bypass Login
- Remote Code Execution

There are two types of SQL injection mainly

1. In Band SQL injection
2. Out of Band SQ Injection

SQL injection

In Band SQL injection

- Error Based
- Union Based
- Blind SQL injection
  - Boolean based
  - Time Based

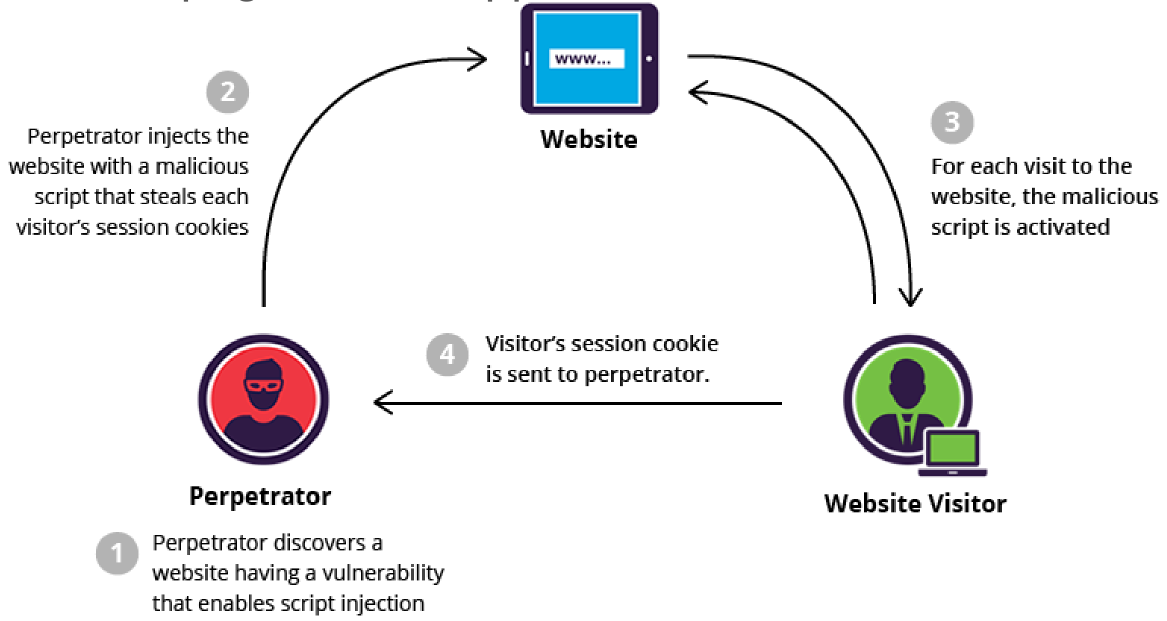
LAB-A01: Injection-1: SQL

Lab objective: Retrieve data via SQL injection.

Lab Requirements: OWASP Juice Shop Lab VM, Burp Suite

Cross Site Scripting (XSS)

Cross-site Scripting (XSS) is a client-side code injection attack. The attacker aims to execute malicious scripts in a web browser of the victim by including malicious code in a legitimate web page or web application.


www...
Website
2
Perpetrator injects the website with a malicious script that steals each visitor's session cookies
3
For each visit to the website, the malicious script is activated
4
Visitor's session cookie is sent to perpetrator.
Perpetrator
Website Visitor
1
Perpetrator discovers a website having a vulnerability that enables script injection

Cross Site Scripting - XSS

There are 3 types of XSS

1. Reflected XSS
2. Stored / permanent XSS
3. DOM Based XSS

Where ?

- Input field (search bar, URL, Parameter)
- Comment / Blog and DOM (HTML DOM)

LAB-A02: Injection-2: XSS

Lab objective: Inject Malicious script into the application.

Lab Requirements: OWASP Juice Shop Lab VM, Burp Suite

A04:2021: Insecure Design

Insecure design is any vulnerabilities in application architecture that create attack opportunities. Insecure design covers a wide range of issues and no single attack vector.

Example – CSRF, missing captcha in input filed etc.

A05:2021 – Security Misconfiguration

Misconfiguration vulnerabilities are configuration weaknesses that might exist in software subsystems or components. For instance, web server software might ship with default user accounts that a cybercriminal could utilize to access the system, or the software might have a known set of standard configuration files or directories, which a cybercriminal could exploit.

Example –

- Default accounts
- Missing hardening / Unnecessary features
- XXE (XML External Entity)

A05:2021 – Security Misconfiguration

- XML sent as data can include “external entities”
- If supported by application, then it may can disclose local files on the server
- In certain configurations, can lead to RCE.

```
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "http://myinternalwebsite"> ]>

<item id="1">

<title>&xxe;</title>

</item>
```

A06:2021: Vulnerable and Outdated Comp.

Most web applications that use components such as libraries, frameworks or plugins always execute then with full privileges and flaws in any component can result in serious impact.

If those components contains bugs/vulnerability then the application may compromise without any vulnerability in application.

As example – wordpress plugins are vulnerable.

A07:2021: Ident. and Authent. Failures

Identification and authentication failures can occur when functions related to a user's identity, authentication, or session management are not implemented correctly or not adequately protected by an application. Attackers may be able to exploit identification and authentication failures by compromising passwords, keys, session tokens, or exploit other implementation flaws to assume other users' identities, either temporarily or permanently.

- Brute force / Credential staffing
- Session Hijacking / Session guessing
- User Enumeration

LAB-A03: Identification & Authentication Failure

Lab objective: Enumerate valid user and dictionary attack .

Lab Requirements: OWASP Juice Shop Lab VM, Burp Suite

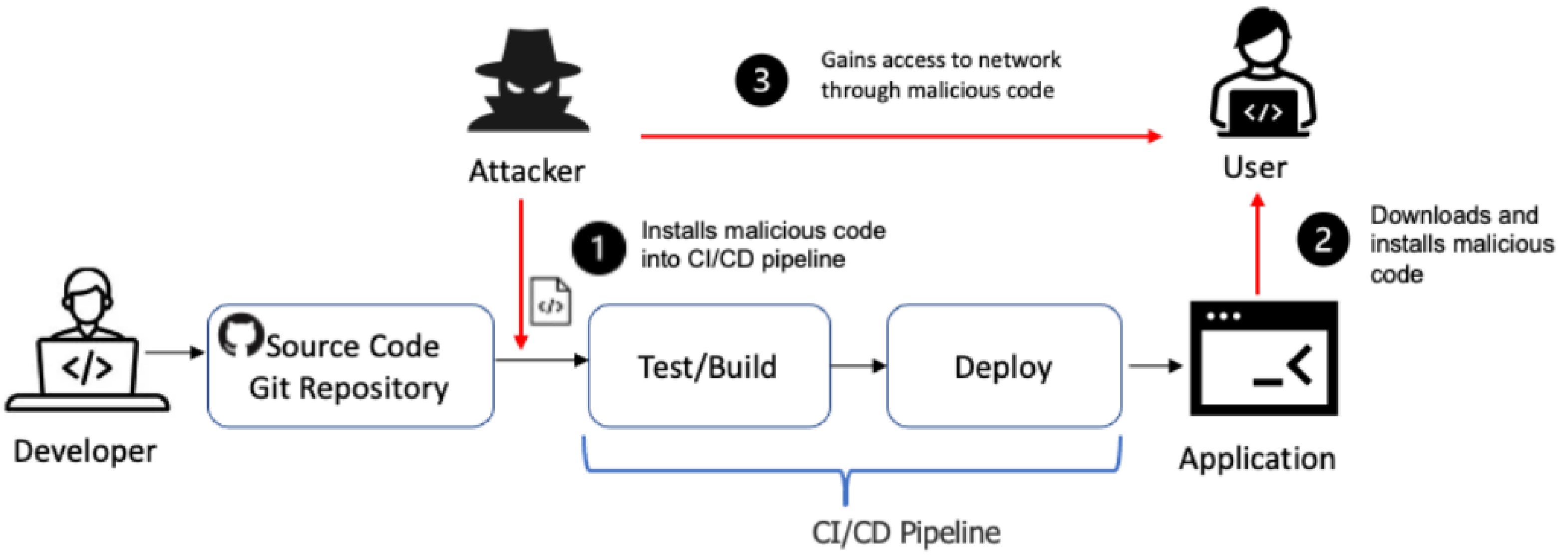
A08:2021: Software Data & Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. This can occur when you use software from untrusted sources and repositories or even software that has been tampered with at the source, in transit, or even the endpoint cache (**supply chain attack**). Attackers can exploit this to potentially introduce unauthorized access, malicious code, or system compromise as part of the following attacks:

- Cache Poisoning
- Code injection
- Command execution
- Denial of Service

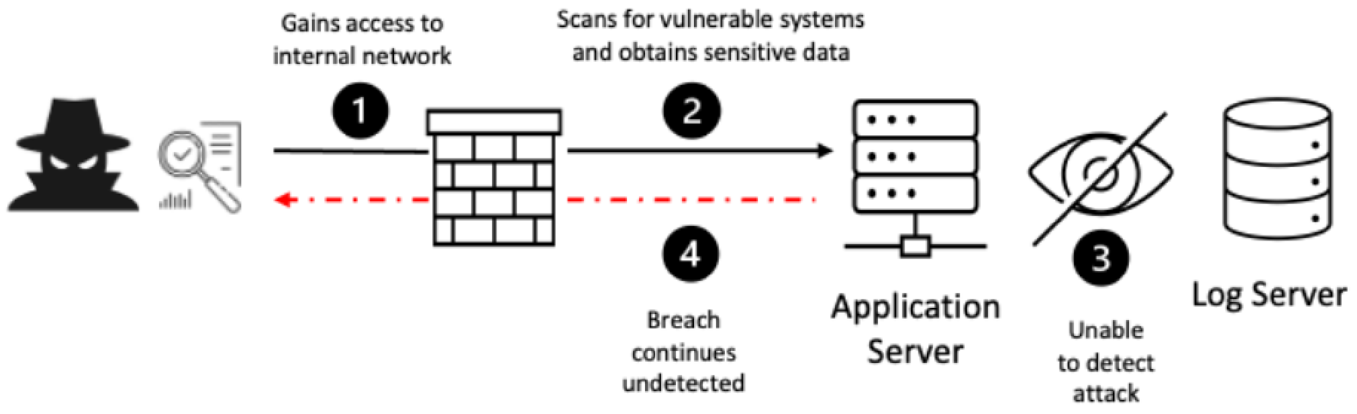
A08:2021: Software Data & Integrity Failures

The SolarWinds Orion attack in which highly targeted malicious updates were distributed to more than 18,000 organizations is one of the most significant breaches of this nature.


Attacker
3
Gains access to network through malicious code
User
1
Installs malicious code into CI/CD pipeline
2
Downloads and installs malicious code
Developer
Source Code Git Repository
Test/Build
Deploy
Application
CI/CD Pipeline

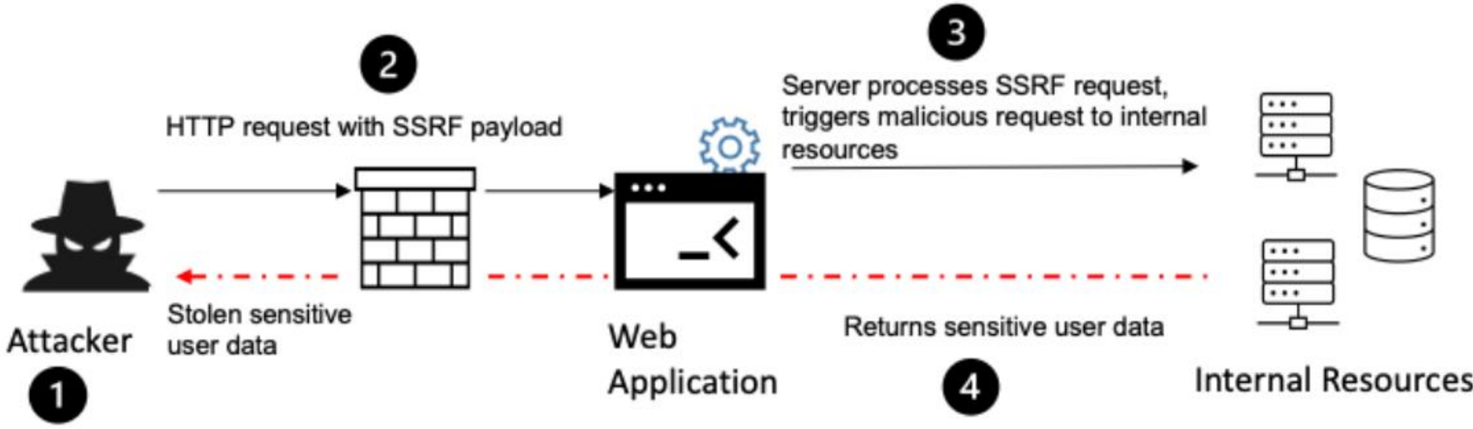
A09:2021:Security Logging & Monitoring

Failure to sufficiently log, monitor, or report security events, such as login attempts, makes suspicious behavior difficult to detect and significantly raises the likelihood that an attacker can successfully exploit your application. For example, an attacker may probe your application or software components for known vulnerabilities over a period. Allowing such probes to continue undetected increases the likelihood that the attacker ultimately finds a vulnerability and successfully exploits the flaw.


Gains access to internal network
1
Scans for vulnerable systems and obtains sensitive data
2
4
Breach continues undetected
Application Server
3
Unable to detect attack
Log Server

A10:2021 – Sever Side Request Forgery (SSRF)

Server-side request forgery (SSRF) flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. The vulnerable web application will often have privileges to read, write, or import data using a URL. To execute an SSRF attack, the attacker abuses the functionality on the server to read or update internal resources. The attacker can then force the application to send requests to access unintended resources, often bypassing security controls



2
HTTP request with SSRF payload
3
Server processes SSRF request, triggers malicious request to internal resources
Attacker
1
Stolen sensitive user data
Web Application
Returns sensitive user data
4
Internal Resources

Demo – DAST Scan and Report

Thank You