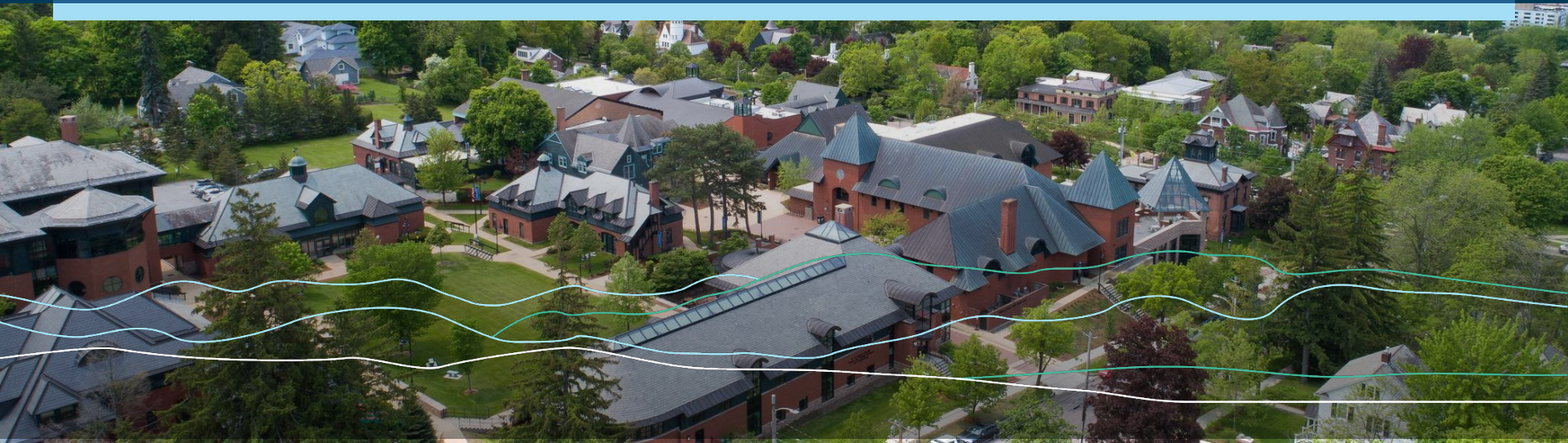
CHAMPLAIN COLLEGE

Mod 12 – Web Security

SEC-110



What is Web Security?

The practice of protecting websites, web applications, and web services from cyber threats.

CIA Triad in Web Security

Confidentiality

Preventing **unauthorized** access to information

How?

- **Encryption** for data at rest/in transit
- **Access** Controls
- **Authentication** Methods

Integrity

Ensuring data is **accurate**, **consistent**, and has not been **tampered** with.

How?

- **Checksums** and **digital signatures**
- **Version** control
- **Access** Logs

Availability

Making sure systems and data are **accessible** to authorized users.

How?

- **Backup** systems
- Disaster recovery plans
- **Load** balancing
- Controls against **DDos** attacks

OWASP Top 10

- **Industry-standard** list of most critical web application security risks
- Updated regularly to **reflect current threat landscape**

OWASP Top 10

Current list as of October 28th, 2025:

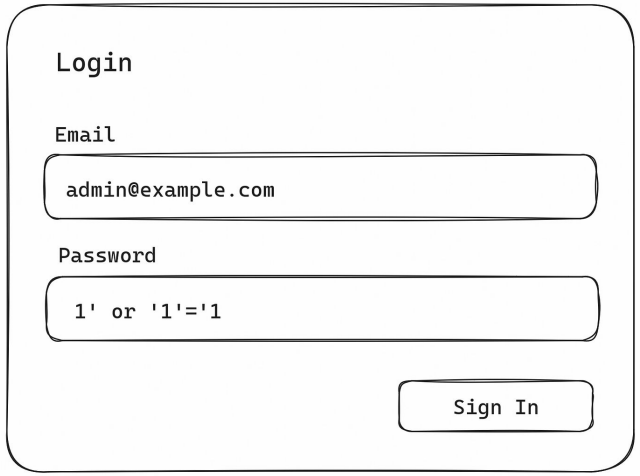
- 1. Broken Access Control
- 2. Cryptographic Failures
- 3. 2021 Injections
- 4. Insecure Design
- 5. 2021 Security Misconfiguration
- 6. 2021 Vulnerable and Outdated Components
- 7. 2021 Identification and Authentication Failures
- 8. Software and Data Integrity Failures
- 9. Security Logging and Monitoring Failures
- 10. Server-Side Request Forgery

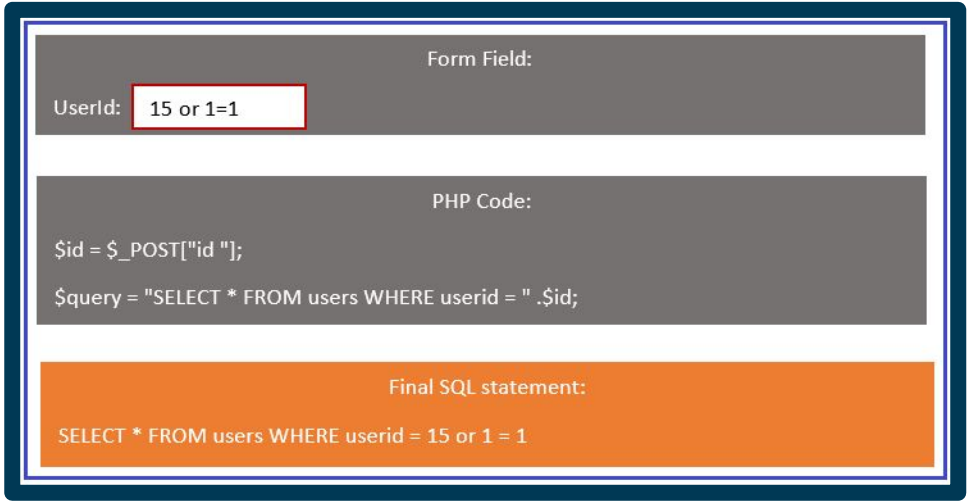
- *https://owasp.org/www-project-top-ten/*

Injection Attacks: SQL

SQL Injection (SQLi):

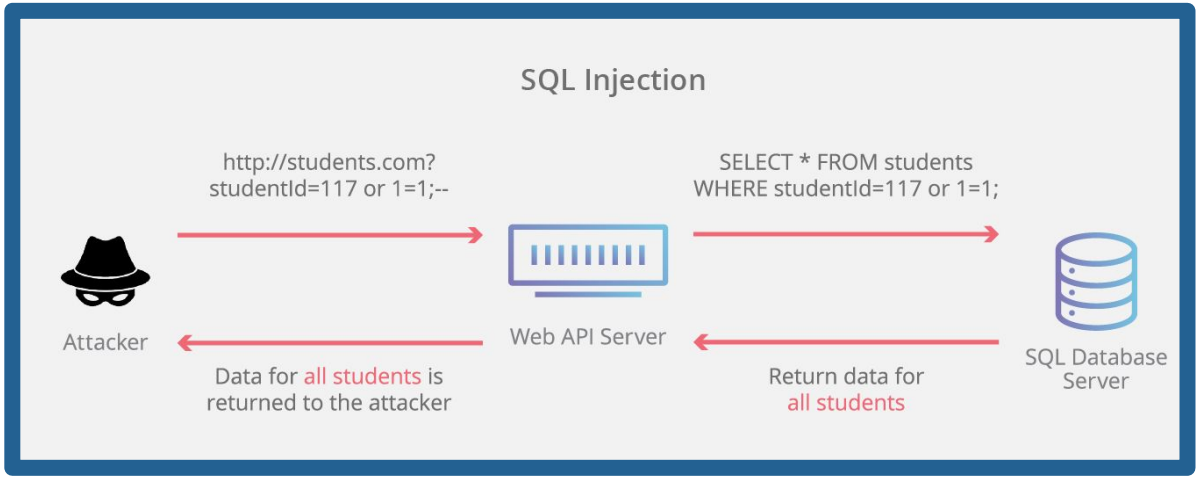
- Attackers insert malicious SQL code into input fields
- SQLi uses conditional logic to trick insecure SQL databases
- Can access, modify, or delete database information.





SQL Injection Example

This example uses SQLi to get all student records in the database



This is what the backend code works to get student records based off a student ID

```
studentId = getRequestString("studentId");
lookupStudent  = "SELECT * FROM students WHERE studentId = " +
studentId
```

SQL Injection Example

This is what the normal SQL query looks like to get a student record based on a student ID

```
SELECT * FROM students WHERE studentId = 117;
```

This is what is added to the query to get all student records. 1=1 means that the statement will always be true!

```
SELECT * FROM students WHERE studentId = 117 OR 1=1;
```

Final result:

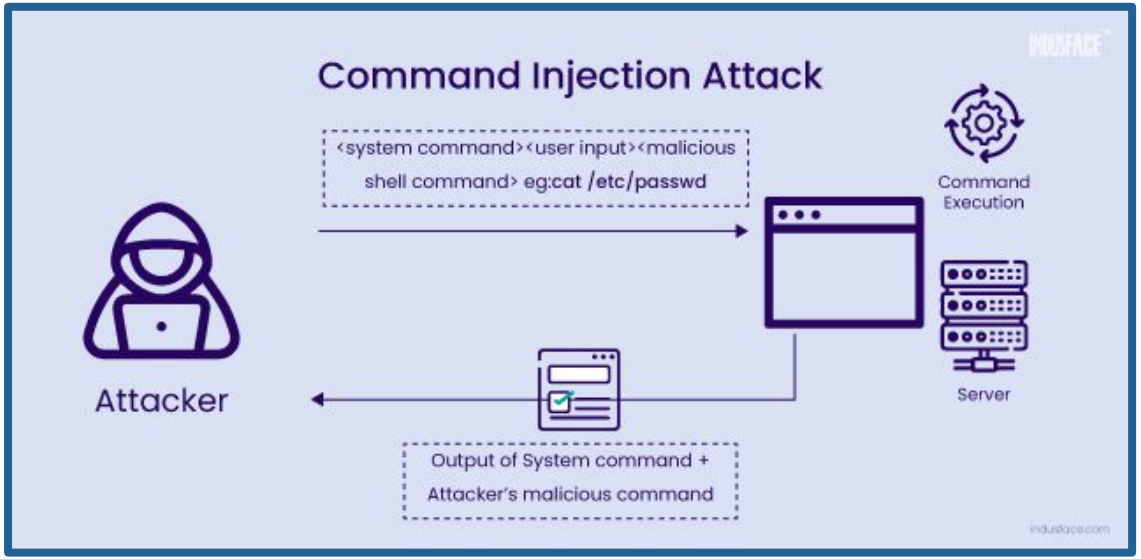
```
All student records returned!
```

Learn more here! https://www.cloudflare.com/learning/security/threats/sql-injection/

Injection Attacks: Command

Command Injection:

- An attacker submits input that gets executed on the server, allowing them to run arbitrary OS commands

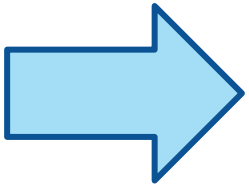


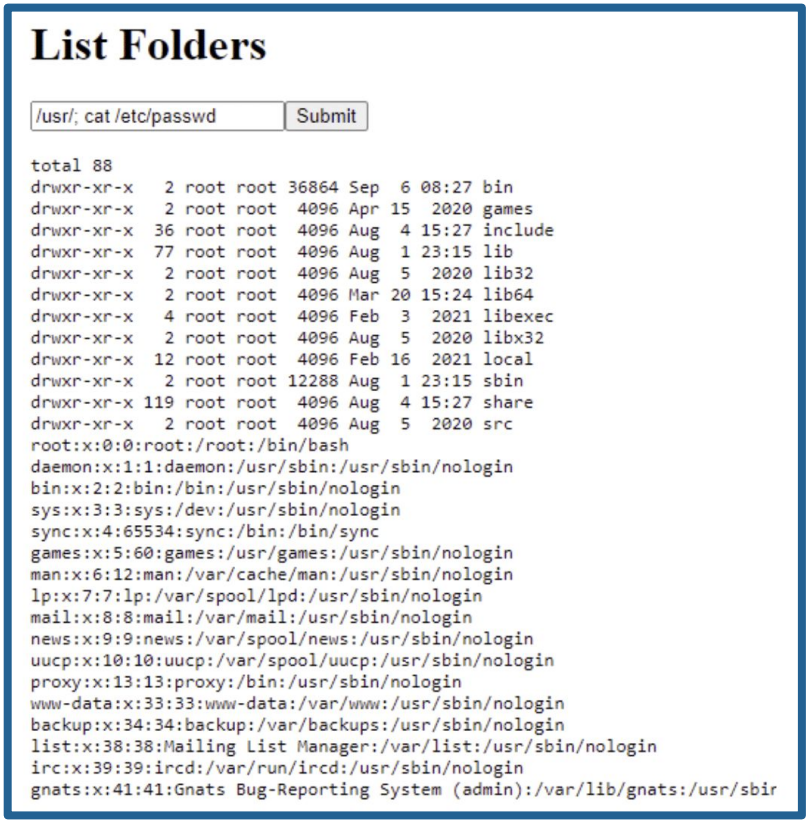
Command Injection: Example

In this example we **exploit the input field** by injecting additional commands to break out of the intended operation.

By using a command separator **(like ; or &&)**, we can chain a second command (cat) that executes on the system!

This command **outputs the contents of the /etc/passwd file,** which contains user account information on Linux systems.





Cross-Site Scripting (XSS)

Definition: a vulnerability that allows an attacker to inject malicious client-side scripts into a web page

Types:

- **Stored** - Malicious script saved in database
- **Reflected** - Script immediately returned in response
- **DOM-based** - Vulnerability in client-side Javascript

XSS Example

1. The attacker injects malicious script

text=<script>alert(1)</script>

2. Malicious script stored in database

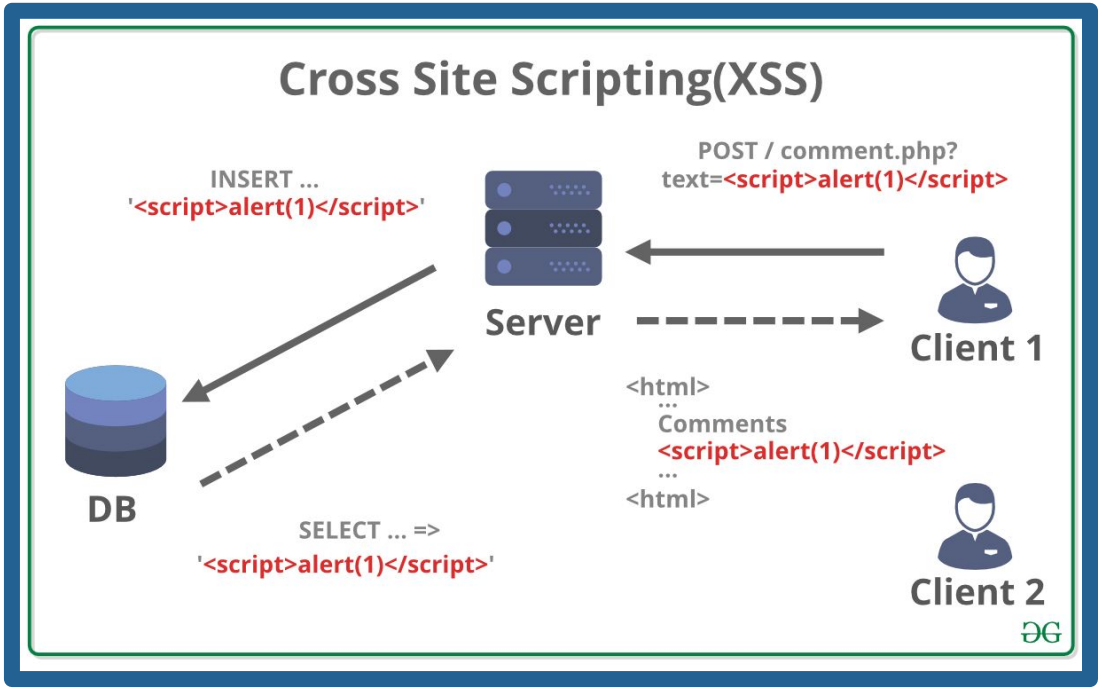
Using POST method

3. Server delivers malicious content

Returns HTML with malicious script

4. The script executes in victim's browser

alert(1) will display as a pop up in the users browser



Cross-Site Request Forgery (CSRF)

An attack that tricks authenticated users into executing **unwanted actions** on a web application where they're currently logged in.

How it works:

- User logs into trusted banking site
- Attacker tricks user into clicking malicious link
- Browser automatically sends authenticated request using the victim's **cookies**.
- Server executes unauthorized money transfer.

This works since the server can't distinguish between legitimate user requests and forged requests from the attacker's page.

Weak Authentication

Weak Authentication Vulnerabilities

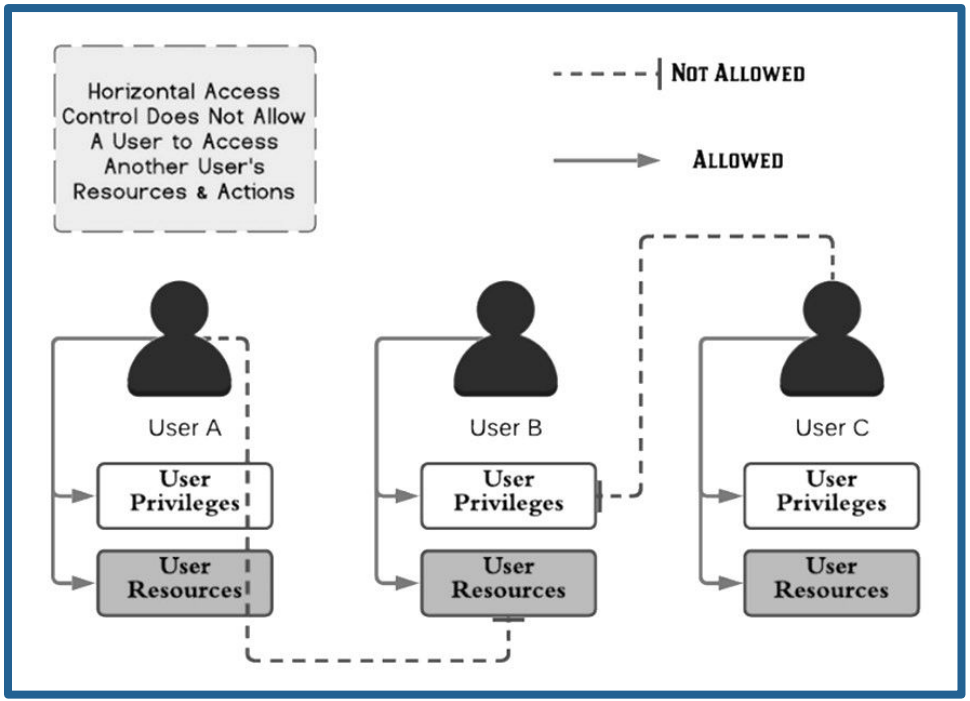
- Weak password requirements
- Session fixation attacks
- Predictable session IDs
- Missing MFA
- Improper session timeout

Broken Access Control

Broken Access Control

Users accessing resources they shouldn't

Example: *a non-admin user accessing admin accounts*



Other Vulnerabilities

Security Misconfiguration

Common configuration errors include:

- Default creds not changed
- Unnecessary features enabled
- Missing security headers
- Outdated software and unpatched vulnerabilities

Insufficient Logging/Monitoring

You can't detect or respond to breaches without proper logging!

- **What to log:**
  - Login attempts
  - Access to sensitive data
  - Admin actions
- **Best practices include** real-time alerts, regular log review, and secure log storage.

How Can We Secure Our Websites?

Request Filtering

Scan incoming HTTP requests and block/allow traffic based on predefined rules.

Prevents:

- Path Traversal attacks
- Known attack signatures

Allowlists and Denylists

Have a ruleset of allowed URLs or a ruleset of blocked URL formats.

Prevents:

- Malicious file uploads
- Dangerous input characters
- Unauthorized API access

How Can We Secure Our Websites?

Input Validation and Sanitization

Harden input fields against injection.

Prevents:

- SQL Injection
- Cross-Site Scripting
- Command Injection

CSRF Token Implementation

Value generated by a server used to prevent a client from performing unauthorized actions.

Prevents:

- Cross-Site Request Forgery
- Unauthorized state changes

HTTP vs HTTPS

Hyper **T**ext **T**ransfer **P**rotocol

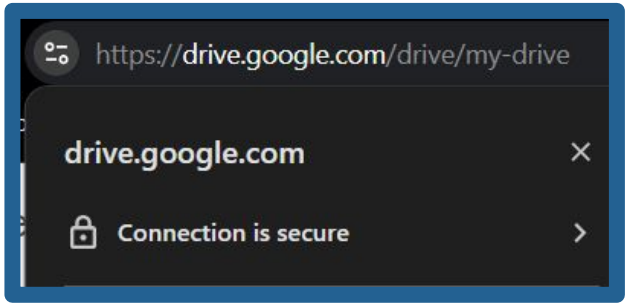
Data Transmitted in **Plain text**

Vulnerable to eavesdropping and **man-in-the-middle** attacks

Hyper **T**ext **T**ransfer **P**rotocol **S**ecure

Encrypted communication using **SSL/TLS**

Protects data **confidentiality** and **integrity**



Secure Password Practices

Bad Passwords

DanielR22 LPatnodeChamp

Password123 DanNetflix

[petname]22 [favorite food]22

Reasons:

- Names are easily guessable
- Common passwords
- Info findable with OSINT

Good Passwords

k&,Xf2:/DWs !! W.F6oH@fV#C

335AdmAD#2!7 W3lln3ss!ID2312

MfcWaT1n2009* M4kwtaK@1PM

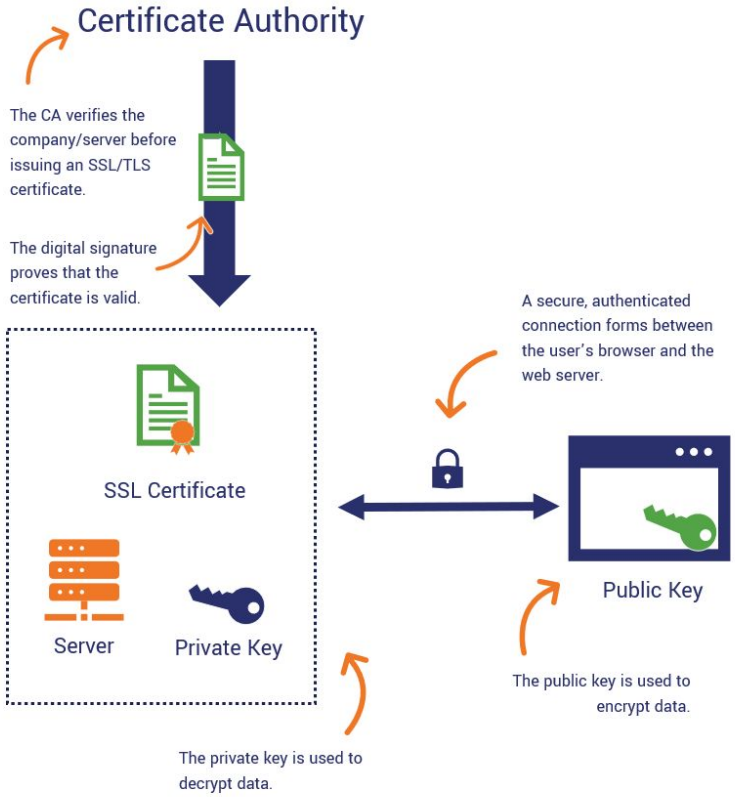
Reasons:

- Random, not guessable
- Special characters, numbers and capitalization
- At least 12 to 15 characters

CA certificates

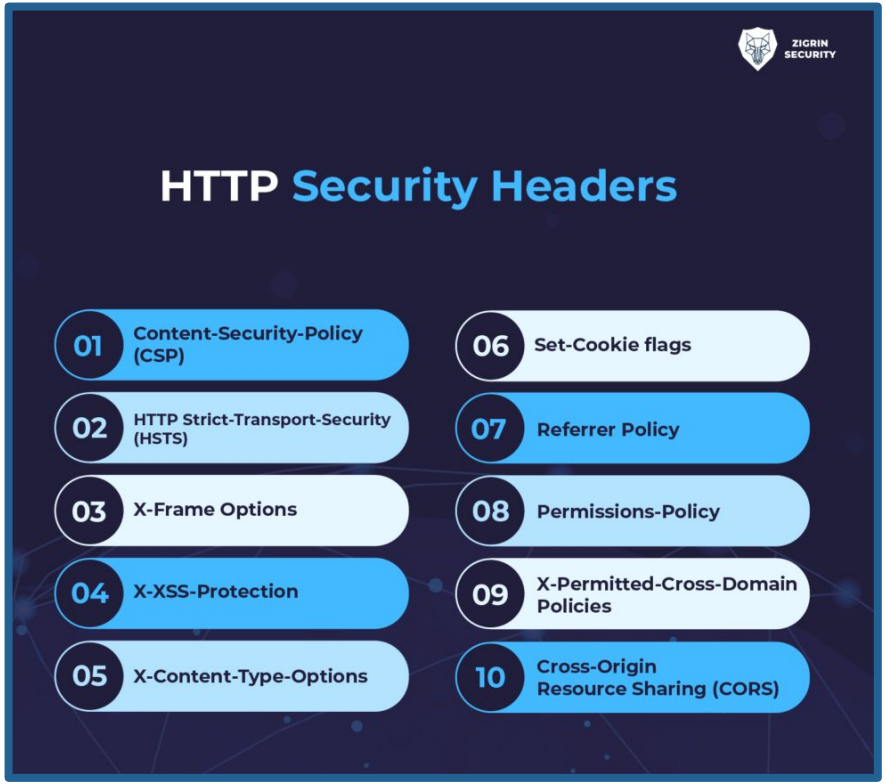
A certificate authority (CA) is a **trusted third-party organization** that issues digital certificates to verify the identity of devices.

It is most commonly used with web servers using a CA like Amazon Trust Services CA or Google's WR2



HTTP Security Headers

An HTTP security header is part of a **website's response** that instructs the browser on how to **handle content securely** and protect against common web attacks.



HTTP Security Headers

- **HTTP Strict Transport Security (HSTS)**
  - Enforces the use of HTTPS and ensures all communication occurs over a secure, encrypted connection.
- **Content Security Policy (CSP)**
  - Mitigates Cross-Site Scripting (XSS) and other code injection attacks by restricting the sources
- **Set-Cookie (with Secure and HttpOnly flags)**
  - Ensures Cookies are only transmitted over HTTPS and cannot be accessed by client-side scripts.

```
# HTTP Security Headers
<IfModule mod_headers.c>
Header set Strict-Transport-Security "max-age=31536000; includeSubDomains"
Header set X-Content-Type-Options "nosniff"
Header set X-Xss-Protection "1; mode=block"
Header set X-Frame-Options "SAMEORIGIN"
Header set Referrer-Policy "strict-origin-when-cross-origin"
Header set Permissions-Policy "geolocation=self"
</IfModule>
```

Secure Development Lifecycle

Requirements

Planning

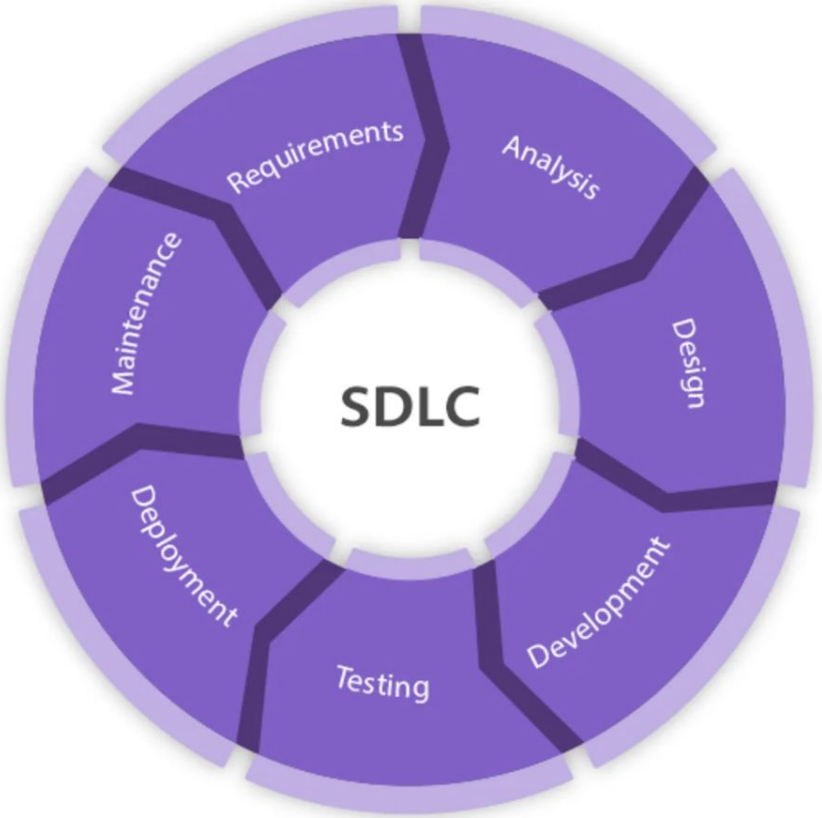
Design

Implementation

Testing

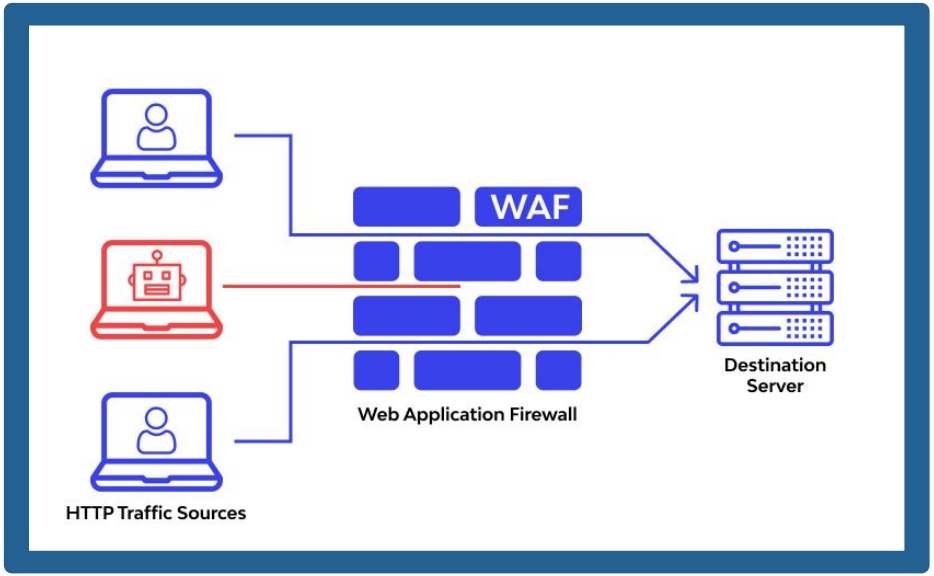
Deployment

Maintenance



Web Application Firewalls (WAF)

Filtering, monitoring, and blocking malicious HTTP traffic between a web application and the internet



Functions:

- Monitors HTTP traffic
- Blocks malicious requests
- Protects against OWASP Top 10

Examples:

- ModSecurity
- Cloudflare WAF
- AWS WAF

Security Testing

Penetration Testing

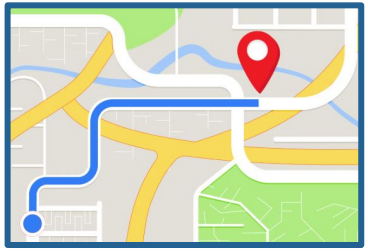
- a simulated cyberattack on a computer system, network, or application to find and exploit security vulnerabilities

Bug Bounty Programs

- Big organizations offer bounties to ethical hackers for finding vulnerabilities

Static/Dynamic Security Testing

- Static analysis (SAST) checks the application's source code without running it
- Dynamic analysis treats the running application as a black box, meaning a pentester is running blind.



Web Security Tools

Burp Suite

- Works as a proxy to intercept, inspect, and modify traffic between a browser and a web server.

ZAP

- Stands between the tester's browser and the web application so that it can inspect and modify messages sent between browser and web application.

Nmap

- Network mapping/discovery tool, can show services and open ports on devices.

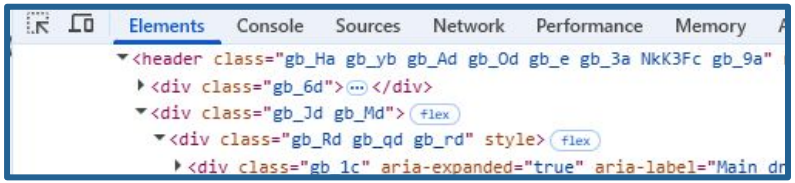
Wireshark

- A packet inspection tool which records and displays intercepted packets.

Browser Developer Tools

- Inspect requests, cookies, storage.

Right-click page , inspect



Best Practices for Everyone!

- Follow principle of least privilege
- Never trust user input
- Keep all software updated
- Implement proper error handling
- Regular security training and code reviews

- Use strong passwords
- Enable MFA
- Keep browsers and plugins updated
- Be cautious of phishing attempts
- Use HTTPS
- Review permissions requested by web applications