Software Security

Web Application Security

Dr. Zeyad Safaa Younus Saffawi

Introduction

Web Application Security is the discipline of **protecting web-based systems, web applications, and APIs from cyber threats and unauthorized access**.
Unlike traditional desktop applications, **web applications are**:

- **Publicly internet accessible**
- **Highly interactive**
- **Dependent on user input**
- **Connected to databases and external services**

- **The primary objective** is to ensure that **web applications continue** to **operate securely and reliably** while **protecting sensitive information** from **Cyber damage, data theft, unauthorized access, fraud and unethical competition, and service disruption**.
- **Web applications** are particularly **vulnerable** because they **are exposed to the public internet**. This means **anyone** including **malicious users** can **interact with them**.
- One of the **biggest challenges in web security is handling user input**. **security challenges** arise mainly from:
  - **Improper input validation**
  - **Weak authentication mechanisms**
  - **Misconfiguration**

Web Application

- A **web application** is a **software application** executed by a **web server** that responds to **dynamic web page requests over HTTP**.
- **web application** typically **consists** of:
  - **collection of scripts**
  - **Web server components**
  - **Databases**
  - **APIs**
  - **Dynamic content generation**
- Using internet infrastructure, **web applications allow users and service providers** to **exchange and manipulate information in a platform-independent manner**.
- Examples:
  - Online banking systems
  - E-commerce websites
  - Social media platforms
  - Cloud-based services

Why are Web Applications Easy Targets?

1. **Publicly accessible over the Internet**, making them exposed to attackers worldwide.
2. **Accept large amounts of user input**, increasing the risk of injection attacks.
3. **Complex architectures** (frontend, backend, databases, APIs) expand the attack surface.
4. **Misconfiguration and human errors** introduce unintended vulnerabilities.
5. **Continuous exposure** to automated scanning and bot attacks.

Web Application Architecture

- **Client Side (Frontend)**
  - Runs in the **user's browser** (HTML, CSS, JavaScript).
  - Vulnerable to attacks such as Cross-Site Scripting (XSS).
- **Server Side (Backend)**
  - Processes HTTP requests, implements business logic, and interacts with databases.
  - Vulnerable to attacks such as SQL Injection and authentication bypass.
- **Databases**
  - Store sensitive and critical information such as user credentials and financial data.
- **Application Programming Interface (APIs)**
  - Enable communication between applications and external services.
  - Improper validation may expose data or allow unauthorized access.
- **Sessions & Cookies**
  - Maintain user authentication, identity, and session state.
  - If not properly secured, they can be hijacked or stolen.


OWASP TOP 10
1 Broken Access Control
2 Cryptographic Failures
3 Injections
4 Insecure Design
5 Security Misconfigurations
6 Vulnerable and Outdated Components
7 Identification and Authentication Flaws
8 Software and Data Integrity Failure
9 Security Logging and Monitoring Flaws
10 Server-Side Request Forgery (SSRF)

OWASP Top 10 Vulnerabilities

The **Open Worldwide Application Security Project (OWASP) Top 10** is a global list of the most critical security risks affecting web applications.

1. **Broken Access Control:** Restrictions on user access are not properly enforced, allowing unauthorized information disclosure or modification. (*Mitigation: Apply Role-Based Access Control (RBAC) and enforce the least privilege principle)*
2. **Cryptographic Failures** Formerly Sensitive Data Exposure Failure to properly protect sensitive data using encryption. (*Mitigation:* Encrypt sensitive data at storage and transition *using strong cryptographic algorithms and secure protocols)*
3. **Injection:** Occurs when untrusted input is interpreted as code or commands. (*Mitigation:* Apply strict input validation and cleanup)
4. **Insecure Design:** Focuses on risks related to design flaws, requiring more "secure by design" principles. (*Mitigation:* Apply threat modeling, secure design principles, and risk assessment during the SDLC)
5. **Security Misconfiguration:** Improper system, server, database, or application settings that expose vulnerabilities. (*Mitigation:* Disabled unused features)

OWASP Top 10 Vulnerabilities

6. **Vulnerable and Outdated Components:** Using libraries or frameworks with known vulnerabilities. (Mitigation: Perform regular vulnerability scanning and maintain an updated inventory of components*)*
7. **Identification and Authentication Failures:** Weaknesses allowing attackers to compromise passwords, keys, or session tokens. *(Mitigation: Enforce strong password policies, implement multi-factor authentication (MFA), secure session management, and protect credentials using hashing and salting techniques.)*
8. **Software and Data Integrity Failures:** Failure to verify the integrity of software updates or data. *(Mitigation: Verify digital signatures and checksums and ensure code integrity validation.)*
9. **Security Logging and Monitoring Failures:** Lack of proper logging and monitoring of security events allows breaches to go undetected *(Mitigation: Implement centralized logging, enable real-time monitoring and alerting, retain logs securely, and conduct regular security audits)*
10. **Server-Side Request Forgery (SSRF):** The web application fetches a remote resource without validating the user-supplied URL. *(Mitigation: Validate and sanitize user input URLs, restrict outbound network access, implement allowlists, and disable unnecessary URL schemas.)*

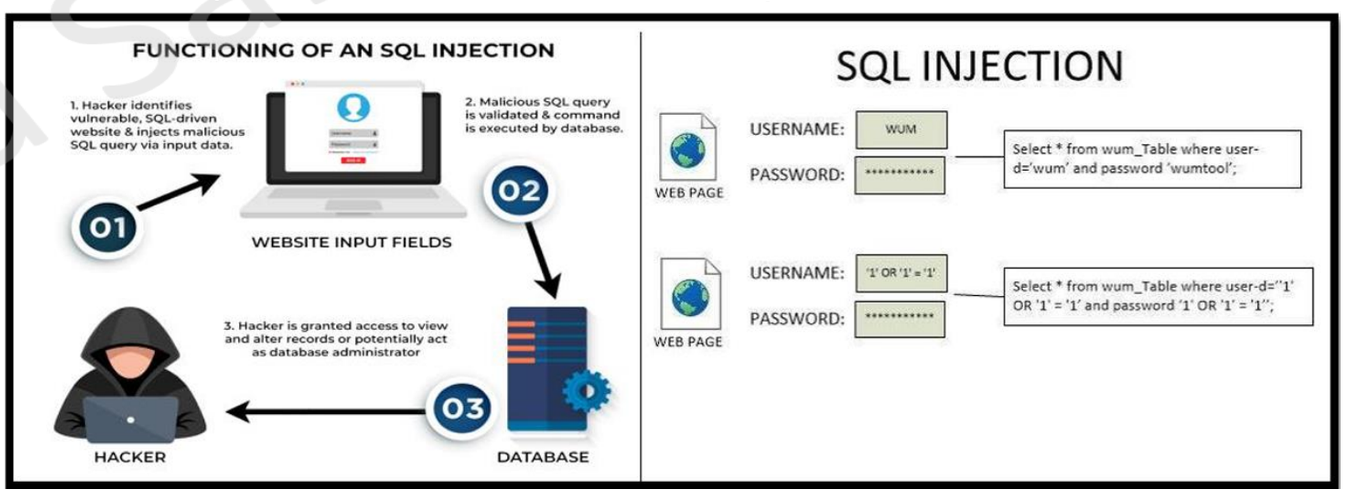
Common Web Application Attack

- **SQL Injection** is one of the **most dangerous web vulnerabilities**
- Occurs when **attackers inject malicious SQL command into input fields** that are **not properly validated and compromise the security of a web application**.
- This **happens** when:
  - **User input is directly inserted into SQL statements.**
  - **Input is not filtered or strongly typed**.

Example: **SELECT * FROM users WHERE username = 'admin' --'**

Impact: can allow an attacker to alter SQL statements passed to the database as parameters.

- **Unauthorized database access**
- **Data theft**
- **Data modification or deletion**
- **Complete database compromise**



Prevention:

- **Parameterized queries**
- **Input validation**
- **Least privilege database accounts**

A **database** is **vulnerable** to **SQL injections** when **user input** is either **incorrectly filtered** for **string literal escape characters embedded** in **SQL statements** or when **user input is not strongly typed**.

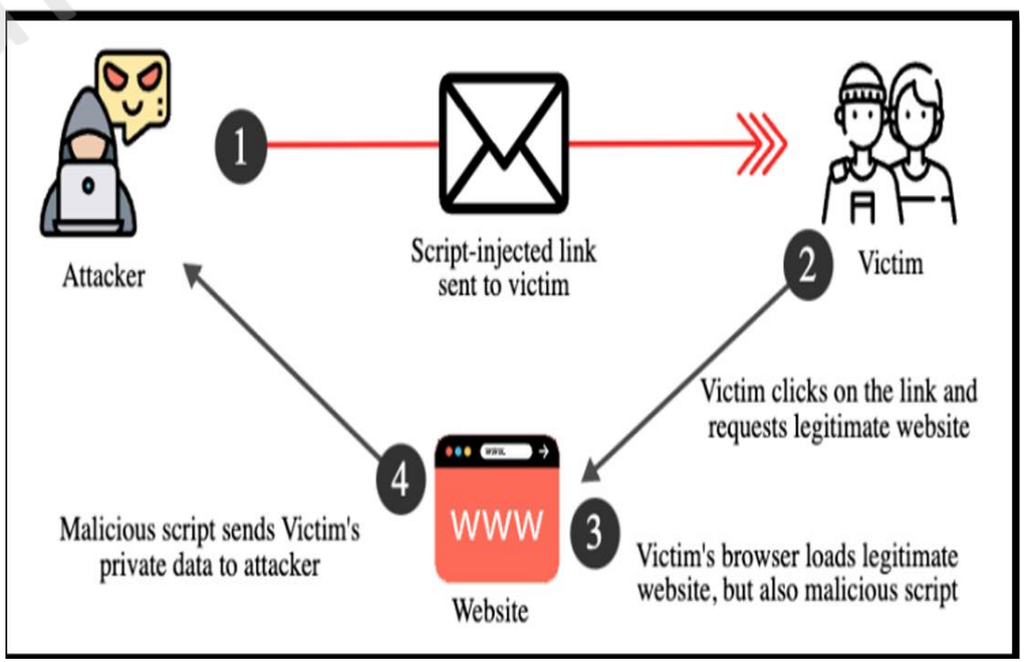
Common Web Application Attack

- **Cross-Site Scripting (XSS)** is a **type of injection attack where malicious scripts (usually JavaScript or HTML) are injected into vulnerable web pages**.
- When the victim loads the infected page, the script runs in their browser.

Attackers use XSS to (**Impact**):

- **Steal session cookies (login | passwords)**
- **Capture login credentials**
- **Redirect users to phishing sites**
- **Spread malicious worms**

- Prevention:
  - **Output encoding**
  - **Input sanitization**
  - **Content Security Policy (CSP)**



Common Web Application Attack

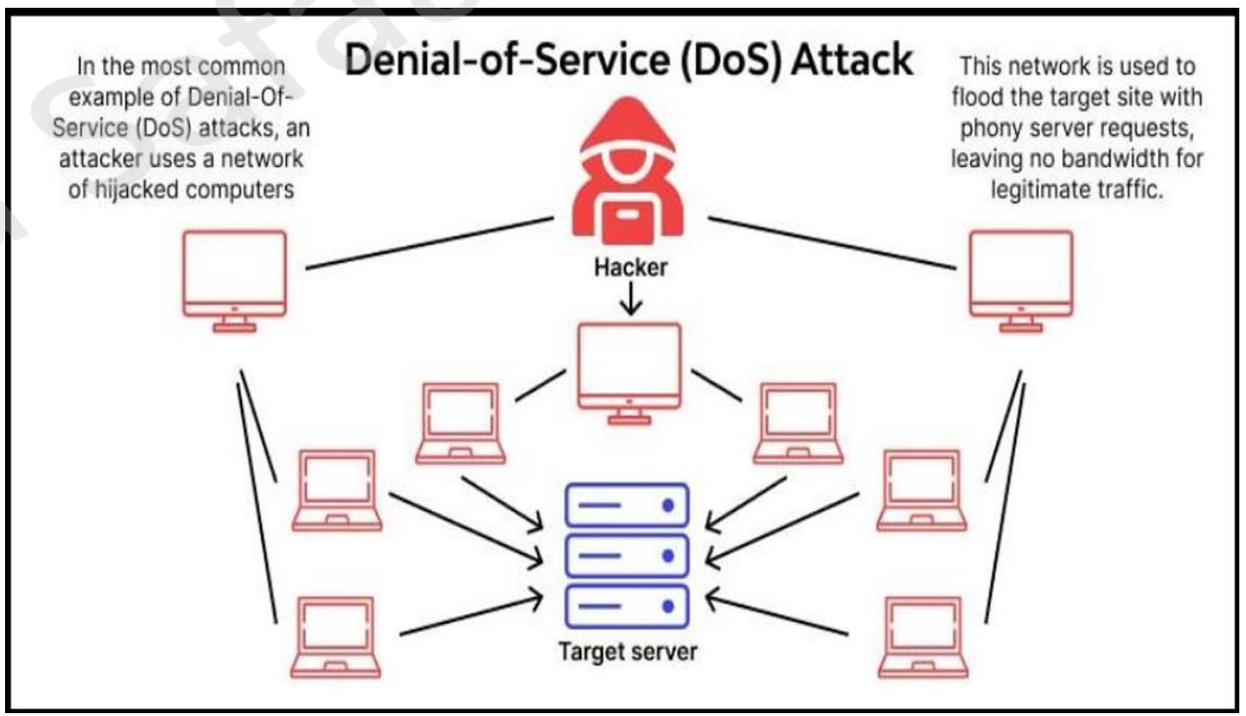
- **A buffer overflow** occurs when an application attempts to writes more data into a buffer than it can hold.
- **Writing outside the space assigned to buffer allows an attacker to overwrite the content of adjacent memory blocks**, causing (**Impact**):
  - **Data corruption**
  - **Application crashes**
  - **Execution of malicious code**

Prevention:

- **Boundary verification**
- **Secure programming languages**
- **Memory protection mechanisms**

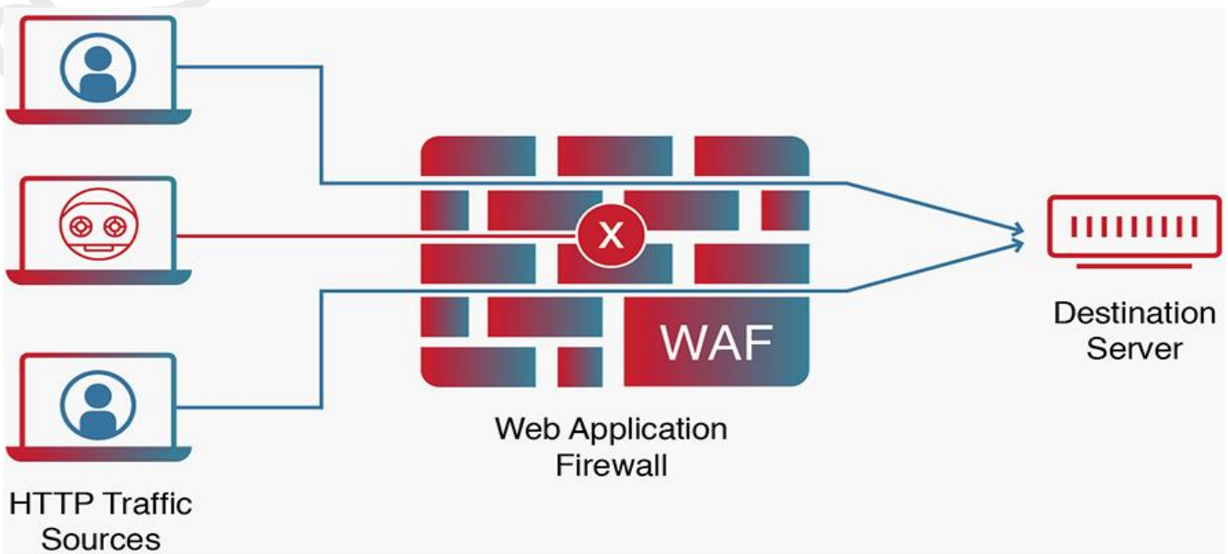
Common Web Application Attack

- A **Denial-of-Service (DoS) attack prevents legitimate users from accessing a systems, service, or other network resources due to the actions of a malicious cyber threat actor**.
- It is **achieved** by **flooding** the **target with excessive traffic** until it becomes **unavailable or crashes**.
- Affected services may include:
  - **Websites**
  - **Email systems**
  - **Banking platforms**
  - **Online services**
- **DoS attacks can result** in:
  - **Financial losses**
  - **Service downtime**
  - **Reputational damage**



Web Application Firewalls (WAF)

- A **Web Application Firewall** (**WAF**) is a security system that monitors and filters internet traffic between web applications and the internet.
- **It is used to monitor HTTP traffic, blocks malicious requests, and protects against common injection attacks such as SQL Injection, Cross-Site Scripting (XSS), and Application-layer attacks.**
- In the **Open Systems Interconnection** (**OSI**) model, a **WAF** operates at **Layer 7** (**Application Layer**)
- However, a WAF is not a complete security solution. It is usually deployed as part of a **broader security architecture** to defend a network, computer, or application that includes:
  - **Network firewalls**
  - **Intrusion detection systems**
  - **Secure coding practices**
  - **Security testing**



References

- Kohnfelder, L. (2021). *Designing secure software: A guide for developers*. O'Reilly Media.

- McGraw, G. (2006). *Software security: Building security in*. Addison-Wesley Professional.

- OWASP Foundation. (2021). *OWASP Top 10: The ten most critical web application security risks*. https://owasp.org/www-project-top-ten/