WEB SECURITY FUNDAMENTALS

SQL Injection, XSS, CSRF & Password Security

TechRacine Solutions Academy

https://techracine.com

https://academy.techracine.com

Study Guide

TechRacine

Table of Contents

TechRacine

Introduction to Web Security

Web Security refers to protecting websites, APIs, and web applications from attacks and unauthorized access.

Modern applications handle sensitive data such as passwords, financial records, and personal information.

Improper security practices can lead to data breaches, hacking, and system compromise.

Developers must understand common vulnerabilities and secure coding practices.

Importance of Web Security

Web security protects customer data and business systems.

Security vulnerabilities may cause financial and reputational damage.

Secure applications improve customer trust and regulatory compliance.

Security should be considered during all stages of software development.

Common Web Security Threats

Modern web applications face several common security threats.

1. SQL Injection

2. Cross Site Scripting (XSS)

3. Cross Site Request Forgery (CSRF)

4. Broken Authentication

5. Session Hijacking

6. Password Attacks

7. API Security Issues

TechRacine

Authentication and Authorization

Authentication verifies user identity.

Authorization controls what users can access.

Applications should implement strong login validation and role-based access control.

SQL Injection Basics

SQL Injection is one of the most dangerous web vulnerabilities.

Attackers inject malicious SQL queries into application inputs.

Improper query handling may expose or modify database data.

SQL Injection attacks commonly target login forms and search inputs.

```
Unsafe Query Example:

SELECT * FROM users
WHERE username = 'admin'
AND password = '123';
```

Types of SQL Injection

Different types of SQL Injection attacks exist.

1. Authentication Bypass

2. UNION-Based Injection

3. Blind SQL Injection

4. Error-Based Injection

5. Time-Based SQL Injection

TechRacine

SQL Injection Prevention

Applications should never directly concatenate user input into SQL queries.

Input validation and parameterized queries improve security.

Least privilege database access should also be implemented.

Prepared Statements and Parameterized Queries

Prepared statements separate SQL logic from user input.

Parameterized queries are one of the best protections against SQL Injection.

```
PHP PDO Example:

$stmt = $pdo->prepare(
"SELECT * FROM users
WHERE email = ?"
);

$stmt->execute([$email]);
```

Cross Site Scripting (XSS)

Cross Site Scripting allows attackers to inject malicious JavaScript into webpages.

XSS attacks can steal cookies, session tokens, and sensitive user information.

Applications that display unsanitized user input are vulnerable.

TechRacine

Types of XSS Attacks

1. Stored XSS

2. Reflected XSS

3. DOM-Based XSS

Stored XSS is especially dangerous because malicious scripts are permanently stored in databases.

XSS Prevention Techniques

Escape user-generated content before displaying it.

Validate and sanitize all inputs.

Use modern frontend frameworks carefully.

Implement Content Security Policy (CSP).

```
Example:

htmlspecialchars($userInput);
```

Cross Site Request Forgery (CSRF)

CSRF attacks trick authenticated users into performing unwanted actions.

Attackers exploit user sessions to execute requests without permission.

Banking and account management systems are common targets.

TechRacine

CSRF Prevention

Applications should use CSRF tokens in forms and requests.

Sensitive actions should require re-authentication when necessary.

SameSite cookie settings improve protection.

```
<input type="hidden"
name="_token"
value="csrf_token_here">
```

Password Security Basics

Passwords should never be stored as plain text.

Weak password storage can expose user accounts during breaches.

Applications should enforce strong password policies.

Password Hashing

Password hashing converts passwords into irreversible hashed values.

Hashing improves security because original passwords cannot easily be recovered.

Modern applications should use secure hashing algorithms.

Using bcrypt and Modern Hashing

bcrypt is one of the most commonly used password hashing algorithms.

Modern frameworks provide built-in password hashing support.

Hashing should always include salting.

```
PHP Example:

$passwordHash =
password_hash(
$password,
PASSWORD_BCRYPT
);
```

TechRacine

Session and Cookie Security

Sessions maintain authenticated user state.

Cookies should use HttpOnly and Secure flags.

Session hijacking protection is important for secure applications.

HTTPS and SSL

HTTPS encrypts communication between browsers and servers.

SSL/TLS certificates improve data protection.

Modern websites should always use HTTPS.

Security Headers

Security headers improve browser-level protection.

Examples include Content-Security-Policy, X-Frame-Options, and Strict-Transport-Security.

Input Validation and Sanitization

All user input should be validated before processing.

Validation checks correctness.

Sanitization removes unsafe characters and content.

TechRacine

Logging and Monitoring

Applications should log security events and suspicious activities.

Monitoring helps detect attacks and abnormal behavior.

Security logs improve incident analysis.

Security Best Practices

Use parameterized queries.

Escape user-generated content.

Use strong password hashing.

Enable HTTPS everywhere.

Validate all user inputs.

Keep frameworks and libraries updated.

Use role-based access control.

Summary

Web security is a critical part of modern software development.

Understanding SQL Injection, XSS, CSRF, and password security helps developers build secure applications.

Secure coding practices protect users, business data, and systems.

Developers should prioritize security throughout the software development lifecycle.

TechRacine