

Task 7

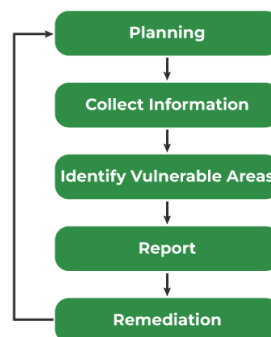
APPLICATION VULNERABILITY TESTING

JASHMI KS

Vulnerability testing, is the process of identifying, evaluating, and mitigating security weaknesses in a system, network, application, or infrastructure. This approach helps organizations detect potential vulnerabilities that could be exploited by attackers, allowing them to take preventive measures before any security breach occurs.

Vulnerability Testing Process

There are five simple steps in the vulnerability testing process:



Application Vulnerability Testing is the process of identifying security weaknesses in web applications that attackers can exploit. This testing helps in understanding how insecure inputs, poor validation, or misconfigurations can lead to data breaches, unauthorized access, or system compromise.

Burp Suite Community Edition is a widely used web application security testing tool. It allows security testers to intercept, inspect, and modify HTTP/HTTPS requests between the browser and the web application. An alternative open-source tool for similar testing is **OWASP ZAP**.

OWASP vulnerabilities:

1. **A01: Broken Access Control:** Failures to enforce appropriate restrictions on what authenticated users can access or do. Server-Side Request Forgery (SSRF) is now included in this category.
2. **A02: Security Misconfiguration:** This includes insecure default settings, incomplete configurations, exposed cloud storage, and unpatched systems, which are common and widespread issues.
3. **A03: Software Supply Chain Failures:** An expansion of the previous "Vulnerable and Outdated Components" to cover the entire software ecosystem, including dependencies, build systems, and distribution infrastructure.
4. **A04: Cryptographic Failures:** Weaknesses related to data encryption, such as using weak algorithms or failing to encrypt sensitive data in transit and at rest.
5. **A05: Injection:** Flaws that allow untrusted data to be sent to an interpreter as a command or query, such as SQL, NoSQL, and command injection, as well as Cross-Site Scripting (XSS).

6. **A06: Insecure Design:** Covers security flaws caused by a lack of security controls or design errors in the architecture and design phase of software development.
7. **A07: Authentication Failures:** Issues related to user identification, authentication, and session management, such as weak passwords or missing multi-factor authentication (MFA).
8. **A08: Software or Data Integrity Failures:** Failures to verify the integrity of software, code, or data artifacts, which can allow attackers to use tampered code or data.
9. **A09: Security Logging & Alerting Failures:** Inadequate logging and real-time active alerting, which can significantly delay incident detection and response.
10. **A10: Mishandling of Exceptional Conditions:** A new category for 2025 that focuses on improper error and exception handling, which can expose sensitive information or aid attackers in triggering specific application behaviors.

Setting Up a Vulnerable Application (DVWA / OWASP Juice Shop)

To safely learn and practice web application security testing, intentionally vulnerable applications are used in a **controlled local environment**.

DVWA (Damn Vulnerable Web Application)

DVWA is a PHP and MySQL-based web application designed to demonstrate common web vulnerabilities at different difficulty levels (low, medium, high). It allows learners to understand how vulnerabilities occur and how security controls affect exploitation.

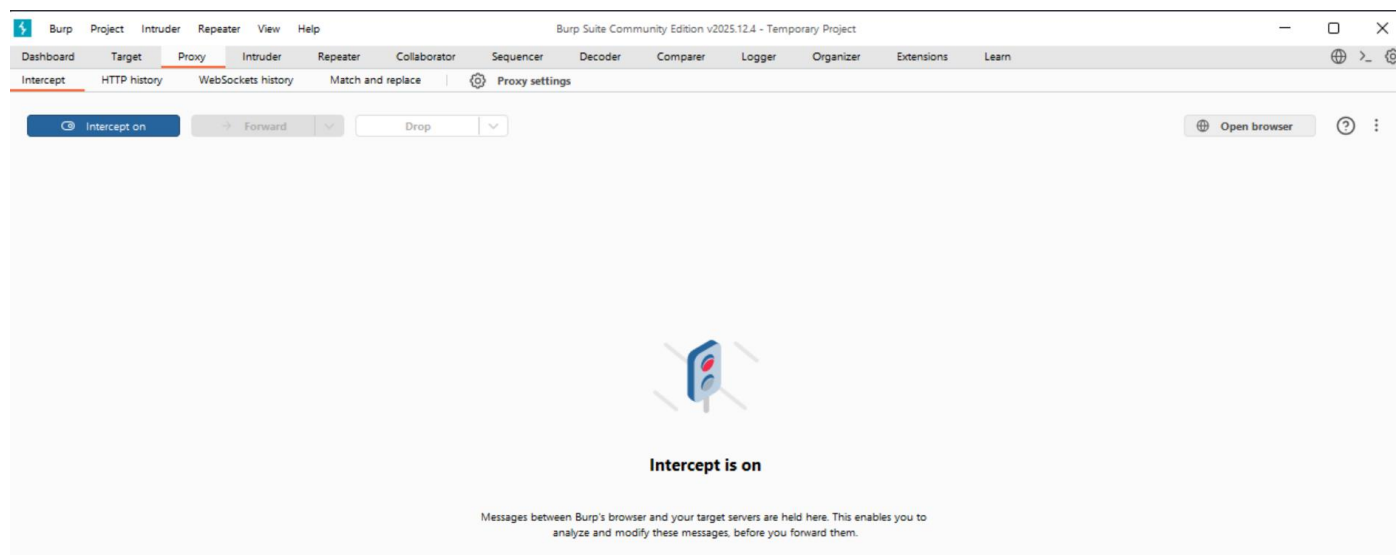
OWASP Juice Shop

OWASP Juice Shop is a modern JavaScript-based application that contains realistic security flaws commonly found in real-world applications. It is widely used for learning advanced web security testing. These applications are typically set up using:

- **XAMPP / WAMP** for DVWA
- **Node.js or Docker** for Juice Shop

Using vulnerable applications ensures that testing does not harm real systems and allows repeated experimentation.

Intercepting Requests Using Burp Suite



Burp Suite is a widely used web application security testing tool that acts as an **intercepting proxy** between the user's browser and the web server.

Once the browser is configured to route traffic through Burp:

- All HTTP and HTTPS requests are captured.
- Request headers, parameters, cookies, and payloads become visible.
- Testers can modify requests before sending them to the server.

This interception helps in understanding how data flows between client and server and enables manual testing of vulnerabilities by tampering with request parameters that are not normally editable through the browser interface.

Testing for SQL Injection

SQL Injection (SQLi) is a vulnerability that occurs when user input is directly embedded into SQL queries without proper validation or sanitization.

How SQL Injection Works

Web applications often interact with databases to authenticate users or retrieve data. If user input is not securely handled, attackers can insert malicious SQL code into input fields.

For example:

- Login forms
- Search boxes
- URL parameters

Using Burp Suite, testers modify request parameters and inject SQL payloads to check whether the application executes unintended database commands.

Impact of SQL Injection

- Bypassing authentication
- Accessing sensitive database records
- Modifying or deleting data
- Complete database compromise

SQL Injection remains one of the most dangerous vulnerabilities due to its severe impact.

Testing for Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) is a client-side vulnerability where attackers inject malicious JavaScript into web pages viewed by other users.

Types of XSS

- **Stored XSS:** Malicious script is permanently stored in the database.
- **Reflected XSS:** Script is reflected in the server's response.
- **DOM-based XSS:** Script execution occurs in the browser through JavaScript manipulation.

Testers enter script payloads into input fields or URL parameters to see whether the application executes the code in the browser.

Impact of XSS

- Stealing session cookies
- Defacing web pages
- Redirecting users to malicious sites
- Performing actions on behalf of users

XSS attacks mainly target users and exploit trust in the web application.

Observing Application Responses

After submitting test payloads, the application's responses are carefully analyzed to identify abnormal behavior.

Testers look for:

- Database error messages
- Unexpected output changes
- Script execution in the browser
- HTTP response code variations
- Disclosure of internal information

These responses help confirm whether the application is vulnerable and reveal how the backend processes user input.

Documenting Identified Vulnerabilities

Proper documentation is a critical part of security testing. Each vulnerability is recorded with clear and structured details.

Documentation typically includes:

- Vulnerability name and category
- Affected endpoint or parameter
- Steps to reproduce the issue
- Observed behavior and impact

Well-documented findings help developers understand the risk and prioritize fixes.

Suggesting Mitigations

Once vulnerabilities are identified, appropriate mitigation strategies are recommended to strengthen application security.

SQL Injection Mitigations

- Use prepared statements and parameterized queries
- Apply strict input validation
- Implement least-privilege database access

XSS Mitigations

- Sanitize and validate all user inputs
- Encode output before rendering
- Use Content Security Policy (CSP)

General Security Measures

- Secure configuration of servers
- Regular security testing and updates
- Use of web application firewalls
- Secure session and authentication mechanisms

Mitigations reduce attack surface and protect applications from real-world exploitation.