

FUTURE_CS_01

Web Application Security Testing

Introduction

- ❖ **Tester Name:** Shruti More
- ❖ **Intern Domain:** Cyber Security Intern
- ❖ **Tools Used:** OWASP ZAP 2.16.1
- ❖ **Operating System:** Windows
- ❖ **Target:** <http://testphp.vulnweb.com>

1. Executive Summary

A penetration test was conducted on <http://testphp.vulnweb.com> using OWASP ZAP to simulate a real-world web application security assessment. The goal was to identify vulnerabilities, evaluate their impact, and recommend mitigation measures.

The assessment revealed **High (SQL Injection)**, **Medium (XSLT Injection)**, **Low (Server Information Disclosure)**, and **Informational (Modern Web Application)** issues. These represent weaknesses that attackers could exploit to compromise confidentiality, integrity, and availability of the system.

2. Methodology

The assessment followed a systematic approach using OWASP ZAP:

- i. **Reconnaissance & Spidering** – Enumerated endpoints using OWASP ZAP Spider.
- ii. **Active Vulnerability Scanning** – Automated scanning of parameters and inputs for injection flaws and misconfigurations.
- iii. **Manual Validation** – Verified results and took screenshots of selected vulnerabilities.
- iv. **Reporting** – Documented findings, severity levels, OWASP Top 10 mapping and mitigation strategies.

Tool Used: OWASP ZAP v2.16.1

Target: <http://testphp.vulnweb.com>

3. Findings

3.1 SQL Injection (High)

- **Description:** Input fields failed to sanitize user input, allowing SQL Injection attacks.
- **Impact:** Attackers could extract or modify sensitive database contents.
- **Severity:** High
- **Mitigation:** Use prepared statements, parameterized queries with strict input validation and ORM frameworks.

- **Evidence: Screenshot of SQL Injection**

Screenshot of ZAP 2.16.1 showing a successful SQL injection exploit.

Header Text:

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Sun, 07 Sep 2025 12:27:30 GMT
Content-Type: image/jpeg
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 11465
```

Body Image:

Alerts (18):

- Cross Site Scripting (Reflected) (18)
- Path Traversal (2)
- SQL Injection
- SQL Injection - MySQL (9)
- SQL Injection - SQLite (Time Based) (3)
- Absence of Anti-CSRF Tokens (40)
- Content Security Policy (CSP) Header Not Set (48)
- Missing Anti-clickjacking Header (44)
- XSLT Injection (2)
- Server Leaks Information via "X-Powered-By" HTTP Response Header (1)
- Server Leaks Version Information via "Server" HTTP Response Header (1)
- X-Content-Type-Options Header Missing (68)
- Authentication Request Identified

Details for SQL Injection - SQLite (Time Based):

- URL:** http://testphp.vulnweb.com/showimage.php?file=/pictures/6.jpg&size=160
- Risk:** High
- Confidence:** Medium
- Parameter:** size
- Attack:** case randomblob(10000) when not null then 1 else 1 end
- Evidence:** SQL injection may be possible.
- CWE ID:** 89
- WASC ID:** 19
- Source:** Active (40024 - SQL Injection - SQLite (Time Based))
- Input Vector:** URL Query String
- Description:** SQL injection may be possible.

Other Info:

Screenshot of ZAP 2.16.1 showing a failed SQL injection attempt due to a database connection error.

Header Text:

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Sun, 07 Sep 2025 12:23:04 GMT
Content-Type: text/html; charset=UTF-8
Connection: keep-alive
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 170
```

Body Text:

```
Warning: mysql_connect(): Connection refused in /hj/var/www/database_connect.php on line 2
Website is out of order. Please visit back later. Thank you for understanding.
```

Alerts (14):

- Cross Site Scripting (Reflected) (18)
- Path Traversal (2)
- SQL Injection
- SQL Injection - MySQL (9)
- Absence of Anti-CSRF Tokens (40)
- Content Security Policy (CSP) Header Not Set (48)
- Missing Anti-clickjacking Header (44)
- Server Leaks Information via "X-Powered-By" HTTP Response Header (1)
- Server Leaks Version Information via "Server" HTTP Response Header (1)
- X-Content-Type-Options Header Missing (68)
- Authentication Request Identified
- Charset Mismatch (Header Versus Meta Content-Type Charset) (1)
- Modern Web Application (9)

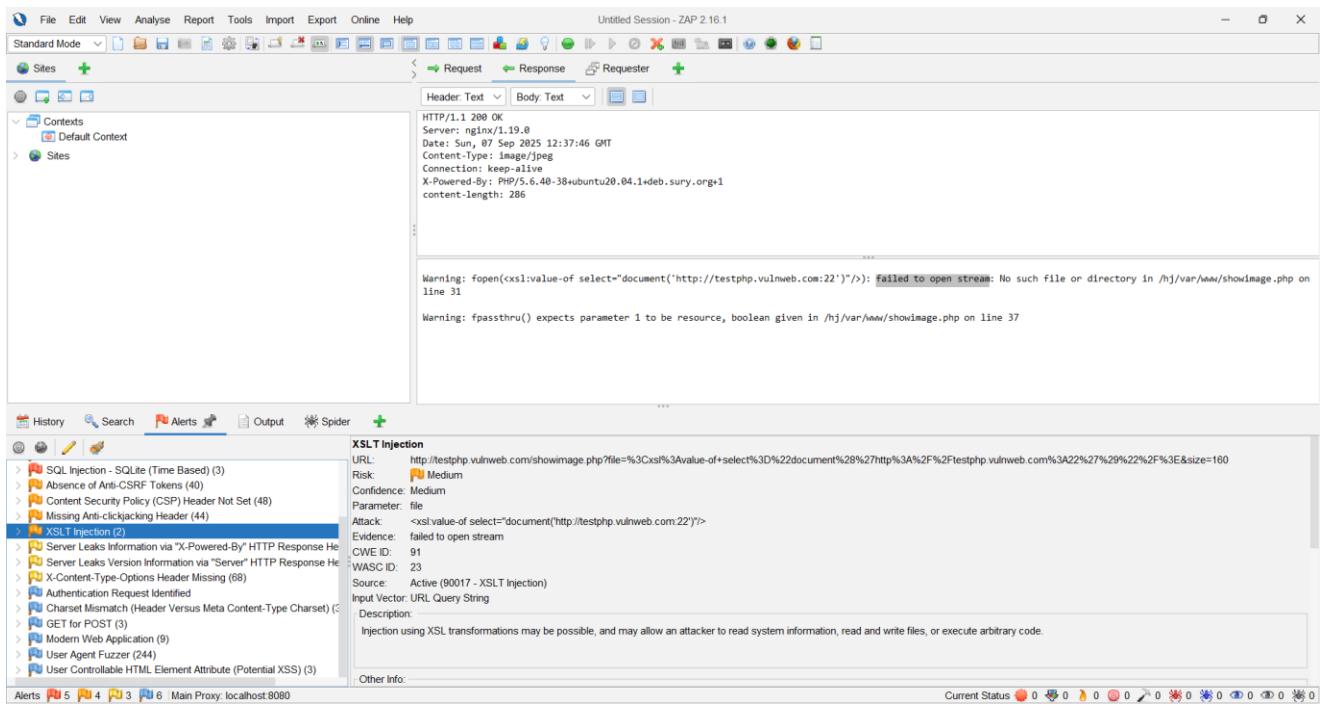
Details for SQL Injection:

- URL:** http://testphp.vulnweb.com/secured/newuser.php
- Risk:** High
- Confidence:** Medium
- Parameter:** uname
- Attack:** ZAP OR '1'='1' --
- Evidence:** SQL injection may be possible.
- CWE ID:** 89
- WASC ID:** 19
- Source:** Active (40018 - SQL Injection)
- Input Vector:** Form Query
- Description:** SQL injection may be possible.

Other Info:

3.2 XSLT Injection (Medium) ●

- **Description:** XSLT Injection vulnerability was detected, allowing manipulation of XML transformations.
- **Impact:** Could lead to server-side data exposure or execution of unauthorized transformations.
- **Severity:** Medium
- **Mitigation:** Disable external entity processing, sanitize XML input, and use safe XSLT parsers.
- **Evidence:** *Screenshot of XSLT Injection*



3.3 Modern Web Application (Informational) ●

- **Description:** Application identifies itself as a "Modern Web Application." This indicates framework usage but does not directly present a vulnerability.
- **Impact:** Possible misconfigurations in modern frameworks could create indirect risks.
- **Severity:** Informational
- **Mitigation:** Regularly review and harden framework configurations, apply updates.

- **Evidence:** Screenshot of Modern Web Application

The screenshot displays the ZAP 2.16.1 interface during a scan of a modern web application. The top portion of the interface shows the Request and Response panes. The Request pane contains an XSS payload: `<script>alert('A cookie was set by JavaScript.');`. The Response pane shows the resulting HTML output, which includes a table with links to 'artists.php', 'categories.php', and 'titles.php'. The bottom portion of the interface is the 'Modern Web Application' summary pane, which lists several findings:

- SQL Injection - SQLite (Time Based) (3)
- Absence of Anti-CSRF Tokens (40)
- Content Security Policy (CSP) Header Not Set (48)
- Missing Anti-clickjacking Header (44)
- XSLT Injection (2)
- Server Leaks Information via "X-Powered-By" HTTP Response Header (1)
- Server Leaks Version Information via "Server" HTTP Response Header (1)
- X-Content-Type-Options Header Missing (68)
- Authentication Request Identified (1)
- Charset Mismatch (Header Versus Meta Content-Type Charset) (1)
- GET for POST (3)
- Modern Web Application (9)
- User Agent Fuzzer (244)
- User Controllable HTML Element Attribute (Potential XSS) (3)

The summary pane also provides details about the application being scanned, such as the URL (`http://testphp.vulnweb.com/AJAX/index.php`), risk level (Informational), and a note that it appears to be a modern web application.

3.4 Server Information Leak (Low) 🌟

- **Description:** The application server discloses version details via response headers (Server and X-Powered-By).
- **Impact:** Attackers can use version information to craft targeted exploits.
- **Severity:** Informational
- **Mitigation:** Disable or obfuscate Server and X-Powered-By headers in the configuration.
- **Evidence:** Screenshot of Server Information Leak (Server & X-powered By)

The screenshot shows the ZAP interface with the following details:

- Sites:** Contexts (Default Context, Sites).
- Request/Response:** Headers:

```
HTTP/1.1 404 Not Found
Server: nginx/1.19.0
Date: Sun, 07 Sep 2025 12:15:07 GMT
Content-Type: text/html
Content-Length: 555
Connection: keep-alive
```
- Body:**

```
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<br><center>nginx/1.19.0</center>
</body>
</html>
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
<!-- a padding to disable MSIE and Chrome friendly error page -->
```
- Alerts:** SQL Injection - SQLite (Time Based) (3), Absence of Anti-CSRF Tokens (40), Content Security Policy (CSP) Header Not Set (48), Missing Anti-clickjacking Header (44), XSLT Injection (2), Server Leaks Version Information via "X-Powered-By" HTTP Response Header (1), Server Leaks Version Information via "Server" HTTP Response Header (1), X-Content-Type-Options Header Missing (68), Authentication Request Identified, Charset Mismatch (Header Versus Meta Content-Type Charset) (1), GET for POST (3), Modern Web Application (9), User Agent Fuzzer (244), User Controllable HTML Element Attribute (Potential XSS) (3).
- Server Leaks Version Information via "Server" HTTP Response Header Field:**
 - URL: <http://testphp.vulnweb.com/sitemap.xml>
 - Risk: Low
 - Confidence: High
 - Parameter:
 - Attack:
 - Evidence: nginx/1.19.0
 - CWE ID: 497
 - WASC ID: 13
 - Source: Passive (10036 - HTTP Server Response Header)
 - Input Vector:
 - Description: The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.
 - Other Info:

The screenshot shows the ZAP interface with the following details:

- Header:** Request, Response, Requester
- Header Text:** HTTP/1.1 200 OK, Server: Apache/2.4.19-0, Date: Sun, 07 Sep 2025 12:15:08 GMT, Content-Type: text/html; charset=UTF-8, Connection: keep-alive, X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1, content-length: 6115
- Response Body:** HTML code including doctype, head, meta, and body sections.
- Alerts:** A list of security issues found:
 - SQL Injection - SQLite (Time Based) (3)
 - Absence of Anti-CSRF Tokens (40)
 - Content Security Policy (CSP) Header Not Set (48)
 - Missing Anti-clickjacking Header (44)
 - XSS Injection (2)
 - Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (1)
 - Server Leaks Version Information via "Server" HTTP Response Header Field(s) (1)
 - X-Content-Type-Options Header Missing (68)
 - Authentication Request Identified (1)
 - Charset Mismatch (Header Versus Meta Content-Type Charset) (1)
 - GET for POST (3)
 - Modern Web Application (9)
 - User Agent Fuzzer (244)
 - User Controllable HTML Element Attribute (Potential XSS) (3)
- Details:** A detailed view of the "Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)" alert.
 - URL: http://testphp.vulnweb.com/categories.php
 - Risk: Low
 - Confidence: Medium
 - Parameter:
 - Evidence: X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
 - CWE ID: 497
 - WASC ID: 13
 - Source: Passive (10037 - Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s))
 - Input Vector:
 - Description: The web/application server is leaking information via one or more "X-Powered-By" HTTP response headers. Access to such information may facilitate attackers identifying other frameworks/components your web application is reliant upon and the vulnerabilities such components may be subject to.
 - Other Info:
- Bottom Navigation:** Alerts (5), Scan (4), Plugins (3), Output (6), Main Proxy: localhost:8080, Current Status (with various icons).

4. OWASP Top 10 Mapping

Vulnerability	OWASP Top 10 Category	Severity (ZAP Flag)
Cross-Site Scripting (Reflected)	A3 – Injection	● High
Path Traversal	A1 – Broken Access Control / A5 Misconfig.	● High
SQL Injection (incl. MySQL & SQLite Time Based)	A1 – Injection	● High
Absence of Anti-CSRF Tokens	A8 – Software and Data Integrity Failures	● Medium
Content Security Policy (CSP) Header Not Set	A5 – Security Misconfiguration	● Medium
Missing Anti-clickjacking Header	A5 – Security Misconfiguration	● Medium
XSLT Injection	A4 – Insecure Design / A1 Injection (related)	● Medium
Server Leaks Info via X-Powered-By Header	A5 – Security Misconfiguration	● Low
Server Leaks Version Info via Server Header	A5 – Security Misconfiguration	● Low
X-Content-Type-Options Header Missing	A5 – Security Misconfiguration	● Low
Authentication Request Identified	A7 – Identification & Authentication Failures	● Informational
Charset Mismatch (Header vs Meta)	A5 – Security Misconfiguration	● Informational
GET for POST	A5 – Security Misconfiguration / Design Issue	● Informational

Modern Web Application (Framework Disclosure)	A9 – Security Logging & Monitoring Failures	● Informational
User Agent Fuzzer	A4 – Insecure Design (Potential Abuse)	● Informational
User Controllable HTML Element Attribute (Potential XSS)	A3 – Injection	● Informational

5. Recommendations

- Implement strict input validation and parameterized queries for all database operations.
- Sanitize and validate XML/XSLT inputs; disable unsafe parsing features.
- Remove or obfuscate sensitive server headers (Server, X-Powered-By).
- Enforce missing HTTP security headers (CSP, X-Frame-Options, X-Content-Type-Options).
- Apply framework updates and follow secure configuration practices.
- Perform regular penetration testing and code reviews.

6. Conclusion

The security assessment revealed vulnerabilities across all severity levels, including high, medium, low, and informational findings. The most critical issue identified was SQL Injection, which poses a direct threat to the confidentiality and integrity of the database. Medium-severity issues such as XSLT Injection and missing security headers increase the risk of exploitation and weaken the overall security posture. Low-severity issues like server information disclosure expose unnecessary details that could aid attackers, while informational findings such as framework disclosure highlight potential misconfigurations that should be monitored. By addressing these vulnerabilities with the recommended mitigations, the application's resilience against cyberattacks will be significantly improved, ensuring better protection of sensitive data and system reliability.