VULNERABILITY ASSESSMENT REPORT FOR A LIVE WEBSITE

Vulnerability Assessment Report

Target Website: testphp.vulnweb.com

Tool Used: OWASP ZAP, Browser DevTools

Prepared by: POOJA D RAO

Date: 14-02-26

Introduction

This report presents the results of a vulnerability assessment conducted on a demo web application using OWASP ZAP. The purpose of the scan was to identify security weaknesses that could be exploited by attackers. The assessment was performed in a safe test environment.

Scope of Assessment

- Target: http://testphp.vulnweb.com
- Tool used: OWASP ZAP
- Method: Automated vulnerability scan
- Environment: Demo testing website
- No real exploitation performed

Findings Summary Table

VULNERABILITY	RISK LEVEL
SQL Injection	Very dangerous
Time Based SQL Injection	Very dangerous
Cross-Site Scripting (XSS)	High
Absence of Anti-CSRF Tokens	Medium
Missing Security Headers	Medium
Directory Browsing	Medium
Hidden File Found	Low-Medium
Information Leak	Low
HTTP Only Site	Low
Charset Mismatch	Low
User Controllable HTML Attribute	-
Other blue alerts	They are notes, not vulnerabilities

Detailed Findings

1. SQL Injection

Meaning:

The website does not protect its database properly. An attacker could type special input and trick the site into revealing or changing database data.

Risk: Very dangerous

Could leak passwords or private data.

Simple explanation:

The application is vulnerable to SQL Injection, meaning attackers can manipulate database queries and access sensitive information.

2. Time-Based SQL Injection

Meaning:

Same as SQL injection, but stealthy. Attackers can slowly extract data without being noticed.

Risk: Very dangerous

Write:

The system is vulnerable to blind SQL Injection using time delays, allowing attackers to retrieve data secretly.

3. Cross-Site Scripting (XSS)

Meaning:

The website allows malicious scripts to run in a user's browser. It's like letting strangers write code inside your page.

Risk: High

Can steal user sessions or cookies.

Write:

The site reflects unsanitized user input, allowing attackers to inject scripts that may steal user information.

4. Absence of Anti-CSRF Tokens

Meaning:

The website doesn't verify if actions are coming from real users. Attackers can trick a logged-in user into clicking something harmful.

Risk: Medium

Write:

The application lacks CSRF protection, making it possible to perform unauthorized actions through forged requests.

5. Missing Security Headers

These include:

CSP Header Not Set, Anti-clickjacking Header Missing

X-Content-Type-Options Missing

Meaning:

The website lacks protective browser rules. Like a car without seatbelts.

Risk: Medium

Write:

Several HTTP security headers are missing, increasing exposure to clickjacking and content injection attacks.

6. Directory Browsing

Meaning:

Attackers can view server folders like an open file cabinet.

Risk: Medium

Write:

Directory listing is enabled, allowing attackers to view internal server files.

7. Hidden File Found

Meaning:

Sensitive files are accessible publicly.

Risk: Low–Medium

Write:

Hidden or backup files were accessible, which may expose sensitive information.

8. Information Leak

Includes:

Server version exposed,X-Powered-By header visible

Banner information leak

Meaning:

The website reveals what software it uses. Helps attackers target known weaknesses.

Risk: Low

Write:

The server discloses version and technology details that could assist attackers in targeting known vulnerabilities.

9. HTTP Only Site

Meaning:

The website doesn’t enforce HTTPS encryption. Data can be intercepted.

Risk: Low

Write:

The site does not enforce secure HTTPS communication, increasing risk of data interception.

10. Charset Mismatch

Meaning:

Encoding settings are inconsistent. Can sometimes lead to injection issues.

Risk: Low

11. User Controllable HTML Attribute

Meaning:

Users can modify page elements.

Potential XSS risk.

Other blue alerts

These are informational:

Authentication request detected

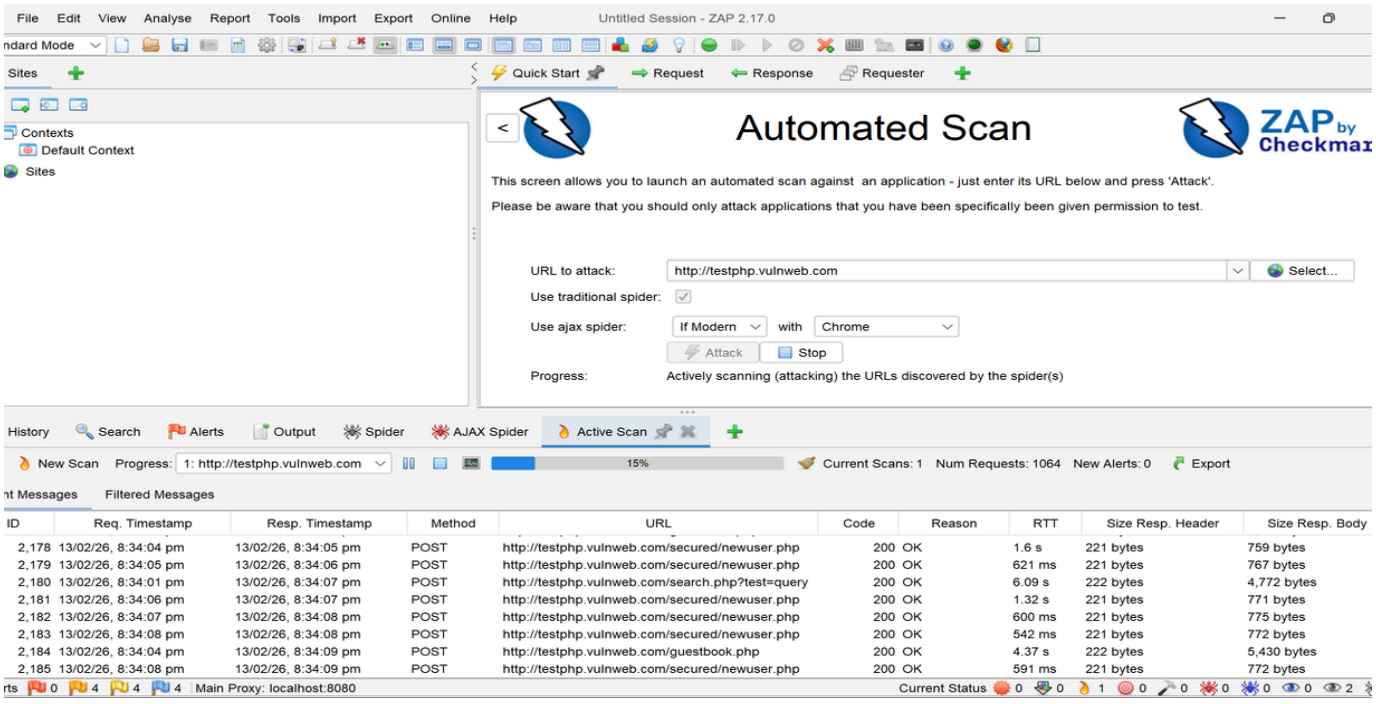
Modern web application

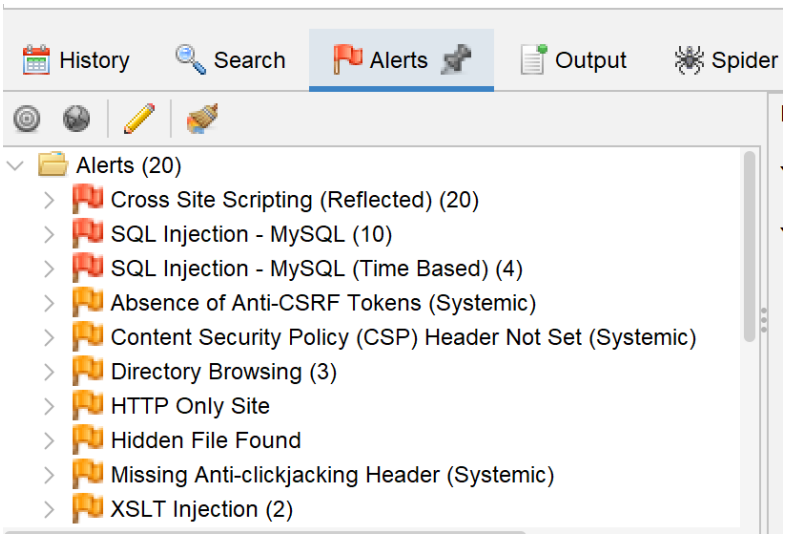
User agent fuzzer

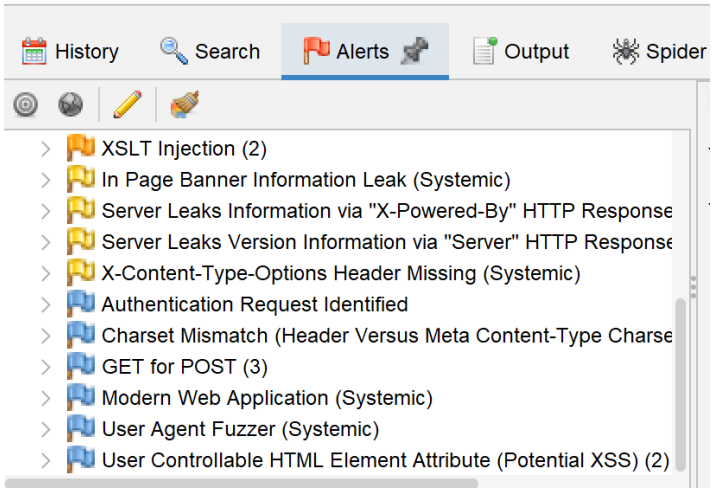
GET for POST

They are notes, not vulnerabilities.

Screenshots Evidence




History
Search
Alerts
Output
Spider
Alerts (20)
Cross Site Scripting (Reflected) (20)
SQL Injection - MySQL (10)
SQL Injection - MySQL (Time Based) (4)
Absence of Anti-CSRF Tokens (Systemic)
Content Security Policy (CSP) Header Not Set (Systemic)
Directory Browsing (3)
HTTP Only Site
Hidden File Found
Missing Anti-clickjacking Header (Systemic)
XSLT Injection (2)


History
Search
Alerts
Output
Spider
XSLT Injection (2)
In Page Banner Information Leak (Systemic)
Server Leaks Information via "X-Powered-By" HTTP Response
Server Leaks Version Information via "Server" HTTP Response
X-Content-Type-Options Header Missing (Systemic)
Authentication Request Identified
Charset Mismatch (Header Versus Meta Content-Type Charse
GET for POST (3)
Modern Web Application (Systemic)
User Agent Fuzzer (Systemic)
User Controllable HTML Element Attribute (Potential XSS) (2)

Remediation Recommendations

To improve the security posture of the application, the following corrective actions are recommended:

1. Prevent SQL Injection

All user inputs should be validated and sanitized before processing. The application should use parameterized queries or prepared statements to prevent attackers from manipulating database commands.

2. Prevent Cross-Site Scripting (XSS)

User-generated content must be properly encoded before being displayed in the browser. Input filtering and output encoding should be implemented to stop malicious scripts from executing.

3. Implement CSRF Protection

Anti-CSRF tokens should be added to sensitive requests to ensure that actions are performed only by authorized users.

4. Add Security Headers

The server should enable standard HTTP security headers such as Content Security Policy, X-Frame-Options, and X-Content-Type-Options to protect users from browser-based attacks.

5. Disable Directory Browsing

Server configuration should prevent public access to internal directories to avoid exposure of sensitive files.

6. Hide System Information

Remove or restrict server version details and technology banners to reduce information disclosure.

7. Enforce HTTPS Encryption

All communication should use HTTPS to protect data from interception during transmission.

8. Conduct Regular Security Testing

Periodic vulnerability assessments and penetration testing should be performed to detect and fix new weaknesses early.

Conclusion

The vulnerability assessment identified several high and medium risk security weaknesses, including SQL Injection and Cross-Site Scripting. These vulnerabilities could expose sensitive data and compromise user safety. Implementing proper input validation, security headers, and regular testing will significantly improve the application’s security. Continuous monitoring and secure development practices are essential to prevent future threats.