

TASK 6 – Cyber Security

Web Application Security Audit Report

1 Target Website

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

Type: Intentionally vulnerable test web application (legal lab)

Purpose: Educational security testing and vulnerability assessment

2 Tools Used

- **OWASP ZAP** – Automated vulnerability scanning
 - **Manual Testing (Browser-based)** – Input validation & logic testing
 - **Kali Linux** – Testing environment
-

3 Vulnerabilities Found

Vulnerability	Risk Level	Impact
SQL Injection	High	Database data exposure, authentication bypass
Cross-Site Scripting (XSS)	Medium	Session hijacking, malicious script execution
Missing Security Headers	Medium	Increased attack surface
Insecure Input Validation	High	Injection attacks
No HTTPS Enforcement	Low	Data interception risk

4 Exploitation Overview

SQL Injection (High Risk)

Some input fields (such as search and login parameters) did not properly sanitize user input. By inserting basic SQL payloads like ' OR '1'='1, the application responded abnormally, indicating possible SQL injection.

Impact:

Attackers can read, modify, or delete database records and bypass login authentication.

Cross-Site Scripting – XSS (Medium Risk)

User input fields reflected JavaScript payloads like <script>alert(1)</script> without proper encoding.

Impact:

An attacker can inject malicious scripts that run in a victim's browser, leading to session hijacking or phishing attacks.

Missing Security Headers (Medium Risk)

Important HTTP headers such as:

- Content-Security-Policy
- X-Frame-Options
- X-Content-Type-Options

were missing.

Impact:

Makes the application more vulnerable to clickjacking and content injection attacks.

Insecure Input Validation (High Risk)

Inputs were not properly validated or filtered.

Impact:

Allows multiple injection attacks including SQLi and XSS.

No HTTPS Usage (Low Risk)

The application runs over HTTP instead of HTTPS.

Impact:

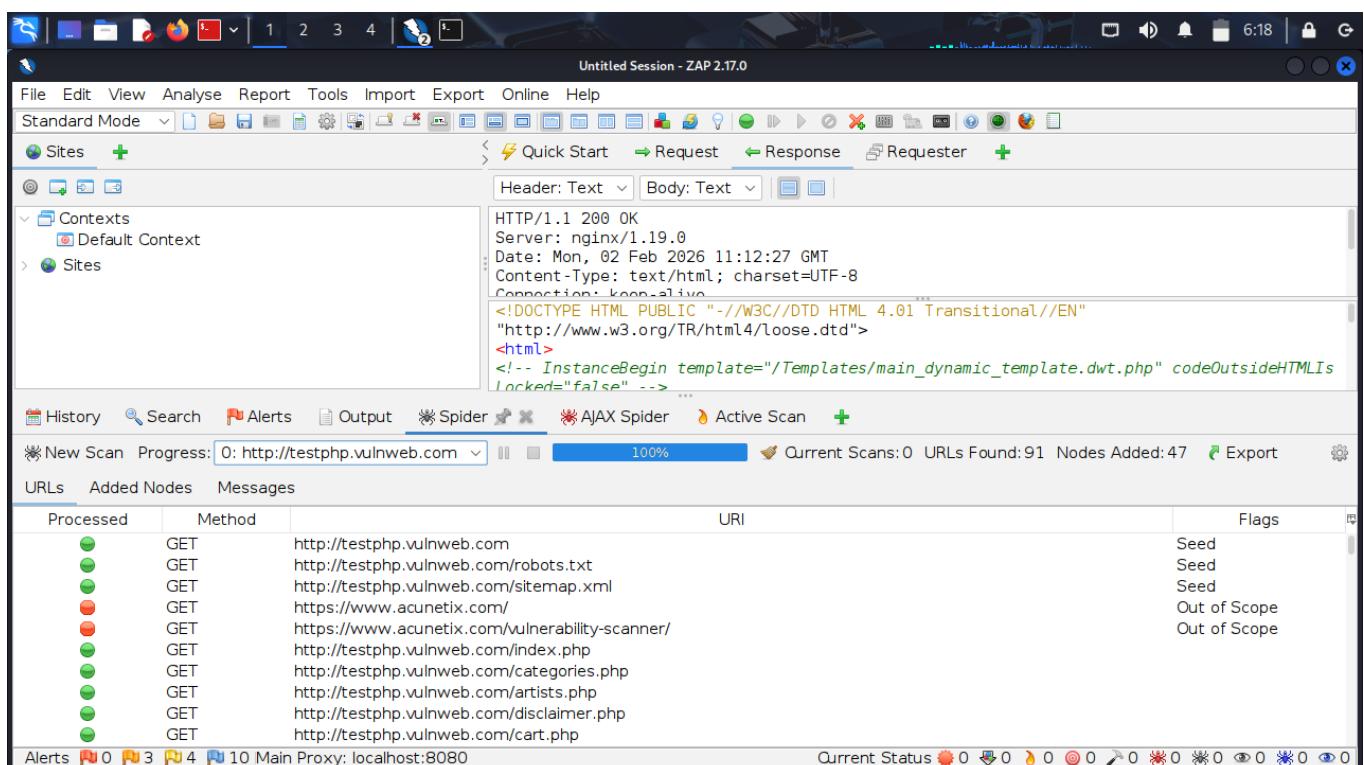
Data sent between client and server can be intercepted using man-in-the-middle attacks.

5 Security Recommendations

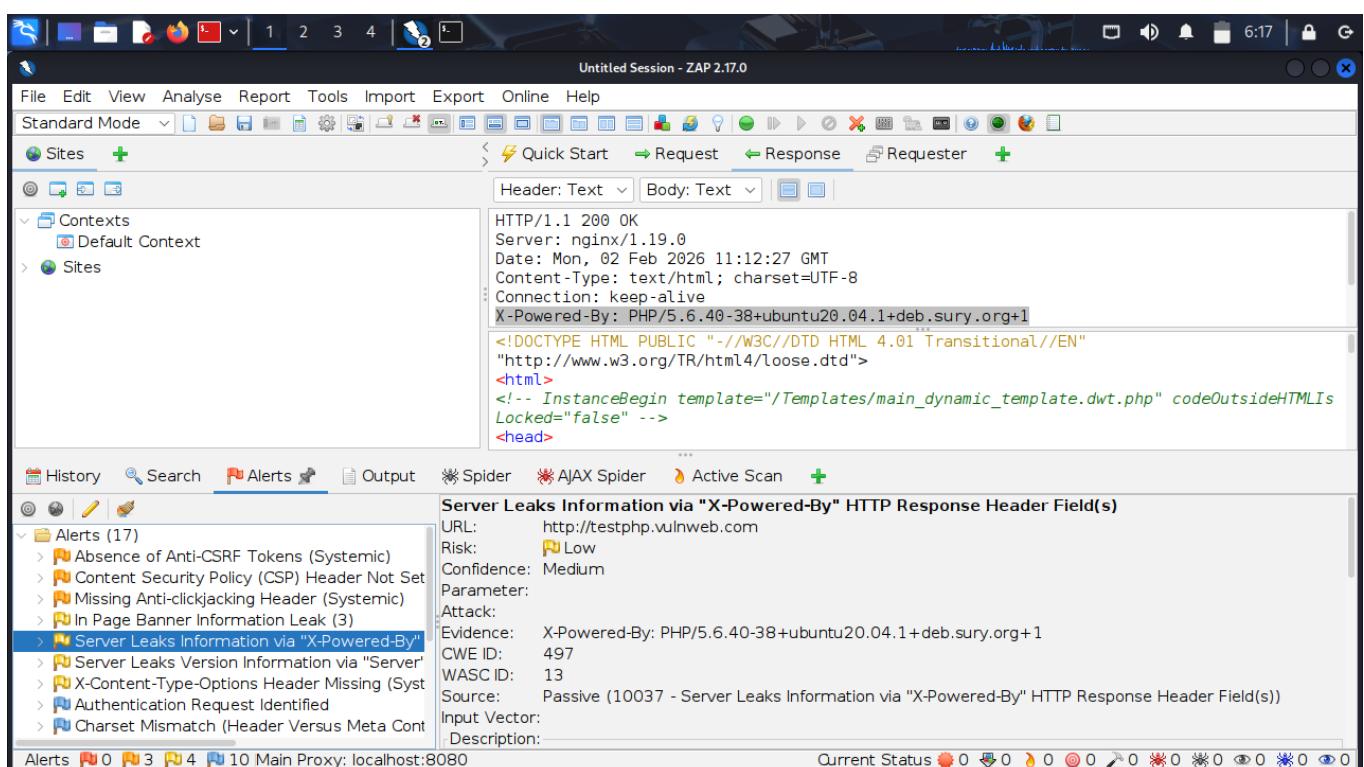
- ✓ Implement **server-side input validation and sanitization**
- ✓ Use **prepared statements / parameterized queries** to prevent SQL Injection
- ✓ Encode output to prevent **XSS attacks**
- ✓ Enforce **HTTPS** using SSL/TLS certificates
- ✓ Add important **security headers**:
 - Content-Security-Policy
 - X-Frame-Options
 - Strict-Transport-Security
 - ✓ Implement secure authentication and session handling
 - ✓ Regularly perform security testing and audits

What I Learned From This Task

From this task, I learned how to conduct a practical web application security audit using both automated and manual techniques. I understood how common vulnerabilities like SQL Injection and XSS are identified, analyzed, and categorized based on risk level. This task improved my understanding of real-world web security flaws, exploitation methods, and how to document findings professionally. It also enhanced my skills in using OWASP ZAP and applying cybersecurity best practices for secure web development.



The screenshot shows the OWASP ZAP interface in Standard Mode. The top navigation bar includes File, Edit, View, Analyse, Report, Tools, Import, Export, Online, and Help. Below the menu is a toolbar with various icons for session management, requests, responses, and tools. The main window has tabs for Sites, Contexts, and Requests. The Requests tab is active, displaying a list of captured HTTP requests with columns for Processed, Method, URI, and Flags (e.g., Seed, Out of Scope). The Headers and Body panes show the raw HTTP response from the target site, which includes an X-Powered-By header. The bottom status bar indicates a progress of 100% and current scan statistics.



This screenshot shows the OWASP ZAP interface with the Alerts tab selected. The left sidebar lists various findings under the 'Alerts' section, including 'Absence of Anti-CSRF Tokens (Systemic)', 'Content Security Policy (CSP) Header Not Set', 'Missing Anti-clickjacking Header (Systemic)', 'In Page Banner Information Leak (3)', 'Server Leaks Information via "X-Powered-By"', 'Server Leaks Version Information via "Server"', 'X-Content-Type-Options Header Missing (Syst...', 'Authentication Request Identified', and 'Charset Mismatch (Header Versus Meta Cont...'. The right pane provides detailed information about the 'Server Leaks Information via "X-Powered-By"' alert, including the URL (http://testphp.vulnweb.com), Risk (Low), Confidence (Medium), and Evidence (X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1). It also lists CWE ID (497), WASC ID (13), Source (Passive (10037 - Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s))), Input Vector, and Description. The bottom status bar shows the same progress and statistics as the previous screenshot.