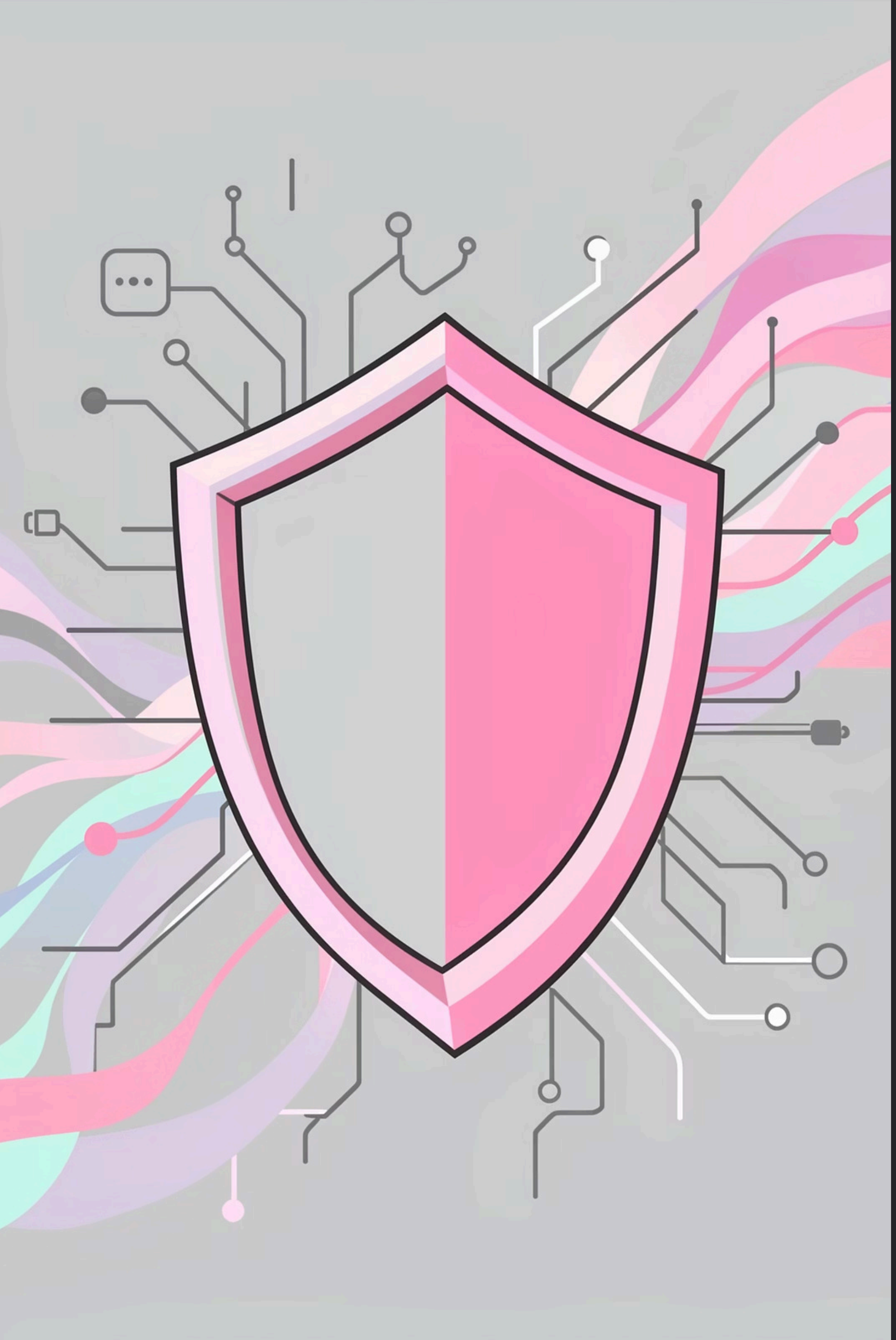


VULNERABILITY ASSESSMENT REPORT Task 1 – Web Application Security Assessment

Assessment Details

Intern Name: Ankit

Internship Program: Cyber Security Internship – Future Interns

Assessment Date: 14 February 2026

Testing Environment: Kali Linux

Quick Overview

This report documents a comprehensive vulnerability assessment of a test web application using industry-standard security tools.

1. Executive Summary

This report presents the findings of a vulnerability assessment conducted on the publicly accessible test web application **http://testphp.vulnweb.com**

The objective of this assessment was to identify common security weaknesses using industry-standard tools including Nmap and OWASP ZAP, without performing any intrusive or exploitative actions.

The assessment identified multiple security misconfigurations primarily related to missing HTTP security headers and information disclosure. While no critical vulnerabilities were discovered, several medium and low-risk issues were observed that could potentially expose the application to security threats if deployed in a production environment.

Overall Risk Overview:

2	4	1
Medium Risk Issues	Low Risk Issues	Informational Issues

This assessment was performed strictly in passive mode for educational purposes as part of the internship program.

1. Scope of Assessment

Target Application

http://testphp.vulnweb.com

Application Type

Web Application Testing Type: Passive Vulnerability Assessment

Testing Method

Service Enumeration and HTTP Response Analysis

Authorization

Publicly accessible intentionally vulnerable test environment

The assessment was limited to passive testing techniques only. No brute force attempts, exploitation attempts, or intrusive scanning methods were performed.

1. Methodology

The vulnerability assessment was conducted using a structured multi-phase approach to ensure systematic identification of potential security weaknesses.

01

Reconnaissance and Service Enumeration

Network-level enumeration was performed using Nmap to identify open ports and running services on the target host.

Command used:

```
nmap -sV testphp.vulnweb.com
```

The scan identified port 80 (HTTP) as open and detected the web server as nginx version 1.19.0.

02

Web Application Exploration

The application was manually browsed using a web browser to generate HTTP traffic. This allowed passive monitoring of server responses without performing any intrusive actions.

03

Passive Vulnerability Scanning

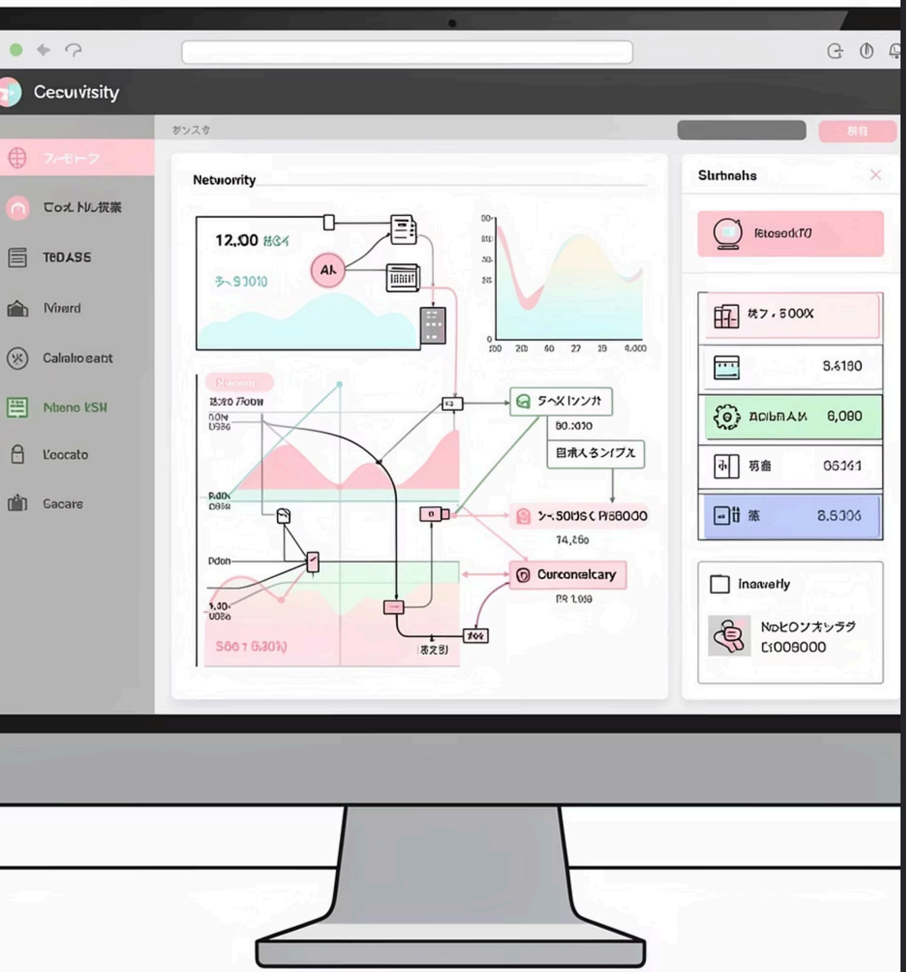
OWASP ZAP was configured as a local proxy (127.0.0.1:8080) to intercept and analyze HTTP requests and responses.

The passive scan module analyzed response headers and application behavior to detect security misconfigurations and information disclosure issues.

04

Header Analysis

HTTP response headers were examined to identify missing security controls such as Content Security Policy (CSP), X-Frame-Options, and X-Content-Type-Options.



5. Tools Used

The following tools were used during the assessment:

Nmap

Purpose: Network scanning and service version detection

OWASP ZAP

Purpose: Passive vulnerability scanning and HTTP response analysis

Kali Linux

Purpose: Testing environment for conducting security assessment

Web Browser

Purpose: Manual exploration and traffic generation

1. Findings

The following security misconfigurations were identified during the passive vulnerability assessment.

🔎 Finding 1: Content Security Policy (CSP) Header Not Set

Risk Level	**Medium**
Confidence	High
CWE Reference	CWE-693

Description: The application does not implement a Content Security Policy (CSP) header. CSP is a security mechanism that helps prevent Cross-Site Scripting (XSS) and other client-side injection attacks by defining approved sources of content.

Impact: Without CSP, malicious scripts may execute in the user's browser if injected, increasing the risk of client-side exploitation and session compromise.

Evidence: OWASP ZAP detected the absence of the Content-Security-Policy header in the HTTP response.

Recommendation: Implement a strict Content-Security-Policy header defining trusted sources for scripts, styles, images, and other resources.

🔎 Finding 2: Missing Anti-Clickjacking Header

Risk Level	**Medium**
Confidence	Medium
CWE Reference	CWE-1021

Description: The application does not include the X-Frame-Options header or an equivalent clickjacking protection mechanism. This header is used to prevent a webpage from being embedded within an iframe on a malicious site.

Impact: Without clickjacking protection, attackers may trick users into clicking hidden elements by embedding the application inside malicious frames, potentially leading to unauthorized actions.

Evidence: OWASP ZAP reported absence of the X-Frame-Options header in the HTTP response.

Recommendation: Configure the web server to include one of the following headers:

`X-Frame-Options: DENY` or `X-Frame-Options: SAMEORIGIN`

🔎 Finding 3: Server Version Disclosure via "Server" Header

Risk Level	**Low**
Confidence	High
CWE Reference	CWE-497

Description: The HTTP response header discloses the web server version information as nginx/1.19.0. Exposing server version details provides attackers with information that may be used to identify known vulnerabilities specific to that version.

Impact: Although version disclosure alone does not directly compromise the system, it aids attackers during the reconnaissance phase by reducing uncertainty about the underlying infrastructure.

Evidence: The HTTP response header contained the value:

```
Server: nginx/1.19.0
```

Recommendation: Disable server version disclosure in the web server configuration to minimize information exposure.

🔎 Finding 4: Information Disclosure via "X-Powered-By" Header

Risk Level	**Low**
Confidence	High
CWE Reference	CWE-200

Description: The HTTP response includes the X-Powered-By header, which reveals backend technology information (such as PHP version). This information may assist attackers in identifying technology-specific vulnerabilities.

Impact: Technology disclosure increases the attack surface by enabling attackers to search for known exploits related to the disclosed version.

Evidence: OWASP ZAP identified presence of the X-Powered-By header in HTTP responses.

Recommendation: Disable the X-Powered-By header in server configuration. For PHP, this can be done by setting:

```
expose_php = Off
```

in the php.ini configuration file.

🔎 Finding 5: X-Content-Type-Options Header Missing

Risk Level	Low
Confidence	Medium
CWE Reference	CWE-16

Description: The application does not include the X-Content-Type-Options header. This header prevents browsers from MIME-sniffing a response away from the declared content type.

Impact: Without this protection, attackers may exploit MIME-sniffing behavior to execute malicious scripts under certain conditions.

Evidence: OWASP ZAP detected absence of the X-Content-Type-Options header in the HTTP response.

Recommendation: Add the following header in server configuration:

```
X-Content-Type-Options: nosniff
```

🔎 Finding 6: Absence of Anti-CSRF Tokens

Risk Level	Low
Confidence	Medium
CWE Reference	CWE-352

Description: The application forms do not appear to implement anti-CSRF (Cross-Site Request Forgery) tokens. CSRF tokens are used to verify that state-changing requests originate from authenticated users.

Impact: Without CSRF protection, attackers may craft malicious requests that cause unintended actions on behalf of authenticated users.

Evidence: OWASP ZAP reported absence of anti-CSRF tokens in application forms during passive analysis.

Recommendation: Implement server-side validation of unique anti-CSRF tokens for all state-changing operations such as login, form submission, and account modifications.

🔎 Finding 7: Charset Mismatch

Risk Level	Informational
Confidence	Low

Description: A mismatch was observed between the charset defined in the HTTP response header and the charset specified in the HTML meta tag.

Impact: Although this does not directly create a critical vulnerability, inconsistent charset definitions may lead to improper content interpretation and unexpected behavior in certain environments.

Evidence: OWASP ZAP reported a charset mismatch between HTTP header and HTML response.

Recommendation: Ensure consistent charset configuration in both server headers and HTML meta tags.

Risk Classification Summary

The vulnerabilities identified during the assessment are categorized as follows:

- **Medium Severity Issues:** 2
- **Low Severity Issues:** 4
- **Informational Issues:** 1

Severity Level	Number of Findings
Medium	2
Low	4
Informational	1

Conclusion

The vulnerability assessment identified multiple security misconfigurations primarily related to missing HTTP security headers and information disclosure.

While no critical vulnerabilities were identified during passive testing, the presence of medium-risk issues indicates that the application lacks proper security hardening controls.

If deployed in a production environment, these weaknesses could increase exposure to client-side attacks and reconnaissance-based exploitation.

Implementing the recommended security headers and configuration changes would significantly improve the overall security posture of the application.

Disclaimer

This assessment was conducted strictly for educational purposes as part of the Cyber Security Internship program at Future Interns.

The target application is an intentionally vulnerable public test environment. No intrusive exploitation or unauthorized access attempts were performed.