VULNERABILITY ASSESSMENT REPORT

Test Website Security Analysis

Target: http://testphp.vulnweb.com

Prepared By:
Bornini Paul

Future Interns
Cyber Security Internship

Date: February 2026

Objective

The objective of this vulnerability assessment is to evaluate the security posture of the target web application and identify potential security weaknesses in a safe and ethical manner.

This assessment aims to:

- Identify publicly exposed services and open ports
- Analyze HTTP response headers for missing or misconfigured security controls
- Detect information disclosure issues related to server and technology details
- Identify insecure cookie configurations and session management weaknesses
- Detect absence of protection mechanisms such as CSRF tokens
- Highlight improper cache control mechanisms
- Classify identified vulnerabilities based on their risk level (Low / Medium)
- Provide clear and practical remediation recommendations

The overall goal is to enhance the security awareness of common web vulnerabilities and demonstrate the process of conducting a professional vulnerability assessment.

Executive Summary

This report presents a vulnerability assessment conducted on the test website http://testphp.vulnweb.com. The objective of this assessment was to identify common security weaknesses through safe and non-intrusive methods.

The analysis was performed using passive techniques such as network scanning, browser-based inspection, and automated vulnerability scanning tools. No active exploitation or harmful actions were carried out during this assessment.

Several security issues were identified, including missing security headers, information disclosure, insecure cookie configurations, absence of anti-CSRF protection, and exposed network services. These vulnerabilities may increase the risk of unauthorized access, data exposure, and web-based attacks.

Each identified issue has been categorized based on its severity level and is accompanied by clear and practical remediation steps. Addressing these vulnerabilities will significantly improve the overall security posture of the website.

Scope of Assessment

The scope of this vulnerability assessment was limited to the publicly accessible components of the target website.

The following activities were **included in the assessment**:
• Analysis of publicly available web pages
• Identification of exposed network services
• Examination of HTTP response headers
• Detection of visible server and technology information
• Identification of insecure cookie configurations
• Basic vulnerability identification using passive scanning tools

The assessment strictly avoided any intrusive or potentially harmful activities.

The following actions were **explicitly excluded from the scope**:
• Exploitation of vulnerabilities
• Brute-force attacks
• Denial-of-Service (DoS) testing
• Authentication bypass attempts
• Any activity that could disrupt normal operation of the website

Methodology

The vulnerability assessment was conducted using a structured and systematic approach to identify potential security weaknesses in the target website.
The methodology followed during the assessment included the following steps:

- **Reconnaissance:**
Initial information gathering was performed to understand the structure of the website and identify publicly accessible components.
- **Network Scanning:**
Basic network scanning was conducted using Nmap to identify open ports and exposed services.
- **Header Analysis:**
Browser Developer Tools were used to examine HTTP response headers and identify missing or misconfigured security headers.
- **Passive Vulnerability Scanning:**
OWASP ZAP was used in passive mode to detect common vulnerabilities without performing any intrusive or harmful actions.
- **Analysis and Classification:**
All identified issues were analyzed and categorized based on their severity level and potential impact.

This methodology ensured that the assessment remained safe, ethical, and non-disruptive while still providing meaningful security insights.

Tools Used

The following tools were used to perform the vulnerability assessment:

- **Nmap**
Nmap was used to perform basic network scanning and identify open ports and exposed services on the target website.

- **OWASP ZAP (Passive Scan)**
OWASP ZAP was used in passive mode to detect common web application vulnerabilities such as missing security headers, insecure cookie configurations, and other misconfigurations without performing any intrusive actions.

- **Browser Developer Tools**
Browser Developer Tools were used to inspect HTTP response headers, analyze requests and responses, and identify information disclosure issues such as exposed server and technology details.

Findings Summary

S.No	Vulnerability	Risk Level	Tool Used
1	Information Disclosure (Server and PHP Version)	Medium	DevTools
2	Missing Security Headers	Medium	DevTools + OWASP ZAP
3	Insecure Cookie Configuration	Medium	OWASP ZAP
4	Missing Anti-CSRF Protection	Low	OWASP ZAP
5	Cache Control Issues	Low	OWASP ZAP
6	Open Port Exposure	Low	Nmap

Detailed Findings

1. Information Disclosure (Server & PHP Version)

Description:
The web server reveals sensitive information such as the server type and the PHP version through HTTP response headers. This information can be used by attackers to identify known vulnerabilities associated with specific versions of software.

Risk Level:
Medium

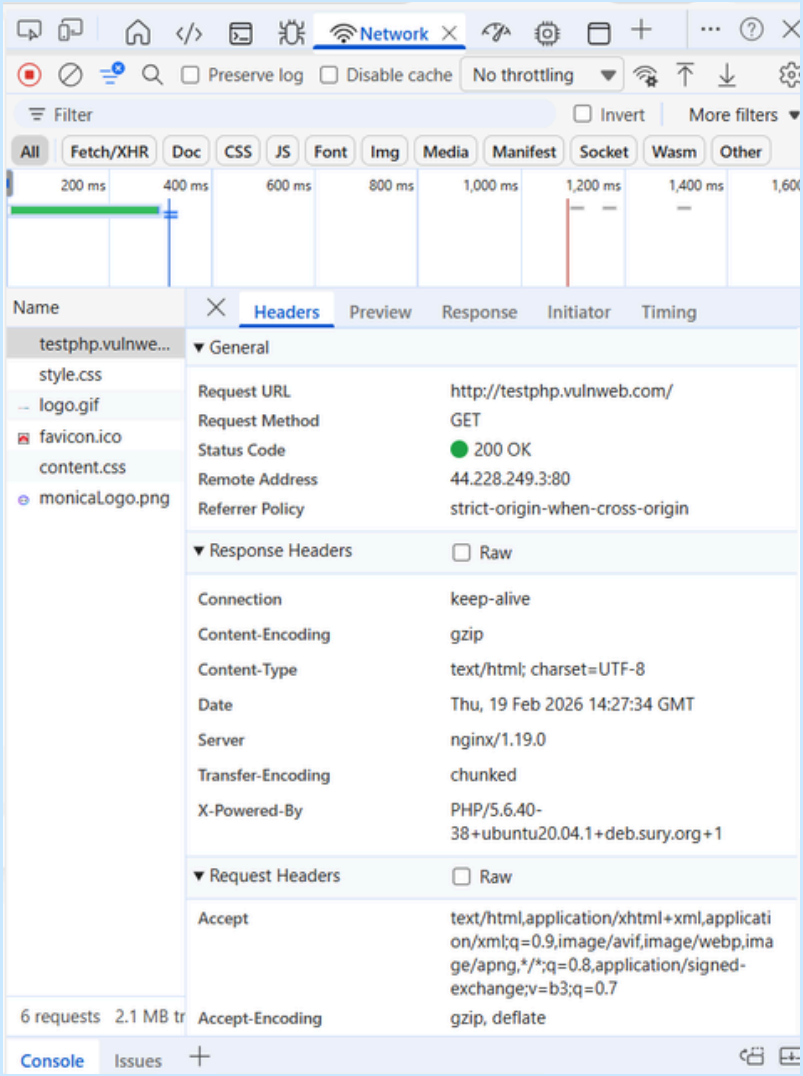
Impact:
Exposure of server and technology details can help attackers plan targeted attacks by exploiting known vulnerabilities in outdated or specific versions of software.

Affected Headers:

- Server → Reveals web server version (nginx/1.19.0)
- X-Powered-By → Reveals backend technology (PHP/5.6.40)

1.Information Disclosure (Continued)

Evidence:



Remediation:

It is recommended to disable or hide server version details and remove the "X-Powered-By" header from responses. This can be done through proper server configuration settings.

2. Missing Security Headers

Description:
The website does not implement several important HTTP security headers that are essential for protecting against common web-based attacks. These headers help enforce security policies in the browser.

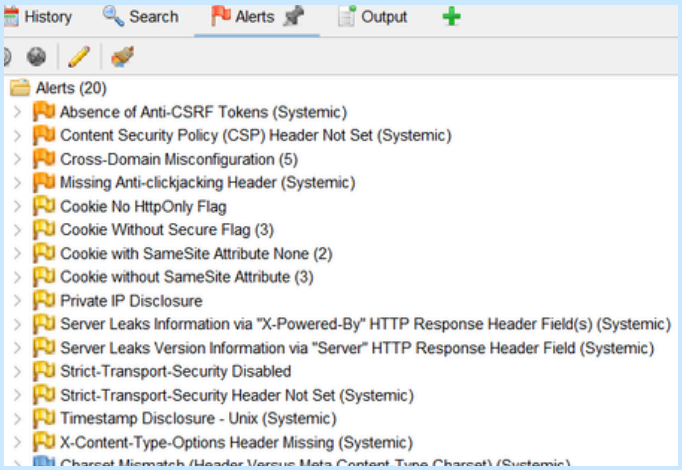
Risk Level:
Medium

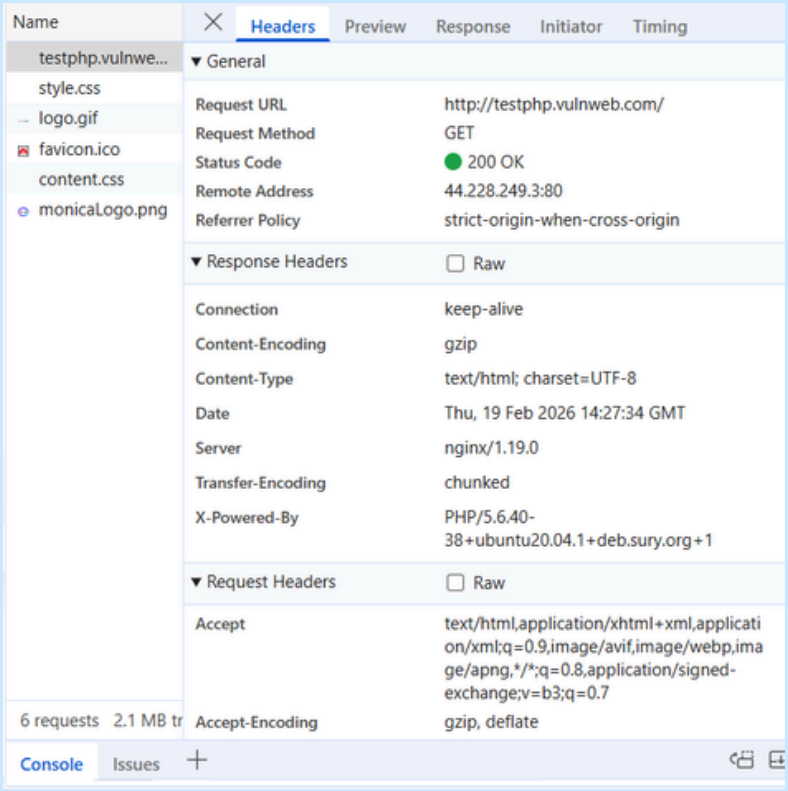
Impact:
The absence of security headers increases the risk of attacks such as Cross-Site Scripting (XSS), clickjacking, and data injection attacks. It weakens the overall security posture of the application.

Affected Headers:
• Content-Security-Policy (CSP) – Not set
• X-Frame-Options – Not set
• Strict-Transport-Security (HSTS) – Not set
• X-Content-Type-Options – Not set

2. Missing Security Headers (Continued)

Evidence:





Remediation:

It is recommended to implement appropriate HTTP security headers on the server. These include setting a Content Security Policy, enabling X-Frame-Options to prevent clickjacking, enforcing HTTPS using HSTS, and adding X-Content-Type-Options to prevent MIME-type sniffing.

3. Insecure Cookie Configuration

Description:
The application uses cookies that are not properly secured with recommended security attributes. Specifically, cookies are missing the "HttpOnly" and "Secure" flags, which are important for protecting sensitive session data.

Risk Level:
Medium

Impact:
Cookies without the HttpOnly flag can be accessed through client-side scripts, increasing the risk of session hijacking through Cross-Site Scripting (XSS) attacks. Additionally, cookies without the Secure flag can be transmitted over unencrypted connections, making them vulnerable to interception.

Affected Configuration:
• HttpOnly flag – Not set
• Secure flag – Not set

3. Insecure Cookie Configuration (Continued)

Evidence:

Remediation:

It is recommended to configure cookies with the "HttpOnly" and "Secure" flags. This ensures that cookies are only transmitted over secure connections and are not accessible via client-side scripts.

4. Missing Anti-CSRF Protection

Description:
The application does not implement Anti-Cross Site Request Forgery (CSRF) protection mechanisms. CSRF attacks trick authenticated users into performing unintended actions on a web application without their consent.

Risk Level:
Low

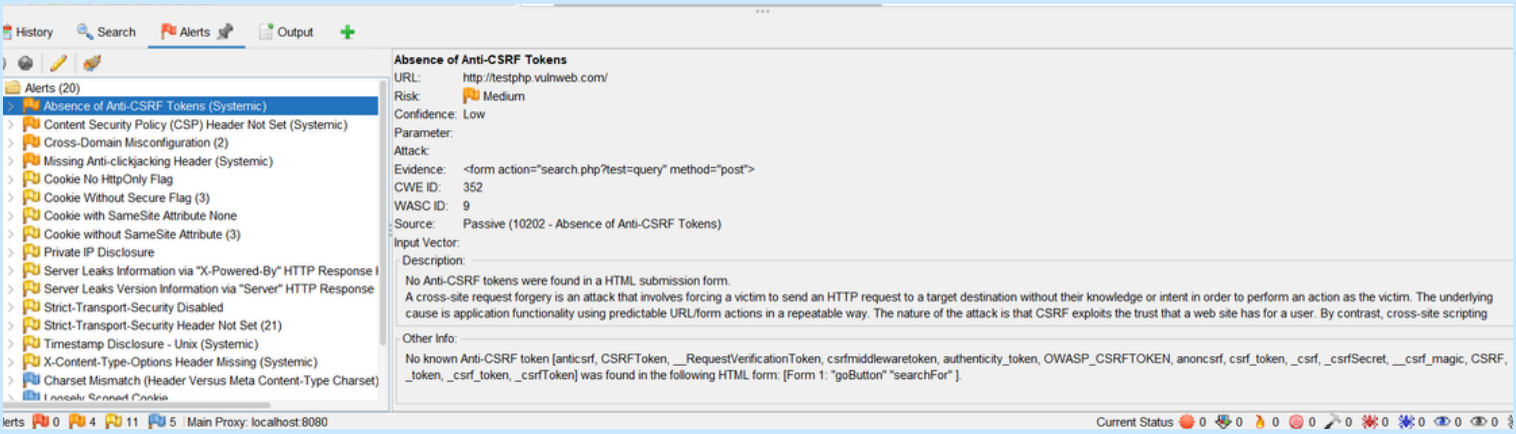
Impact:
An attacker could exploit this vulnerability to perform unauthorized actions such as form submissions or account changes on behalf of an authenticated user. This can lead to unauthorized transactions or data modification.

Affected Area:

- Forms and state-changing requests lack CSRF tokens
- No validation mechanism to verify legitimate user requests

4. Missing Anti-CSRF Protection (Continued)

Evidence:



Remediation:

It is recommended to implement CSRF protection by including unique and unpredictable tokens in all forms and validating them on the server side. Framework-based CSRF protection mechanisms should be enabled wherever possible.

5. Cache Control Issues

Description:
The application does not implement proper cache control mechanisms through HTTP headers. As a result, sensitive data may be stored in the browser cache or intermediary proxy servers, which can be accessed even after the user session has ended.

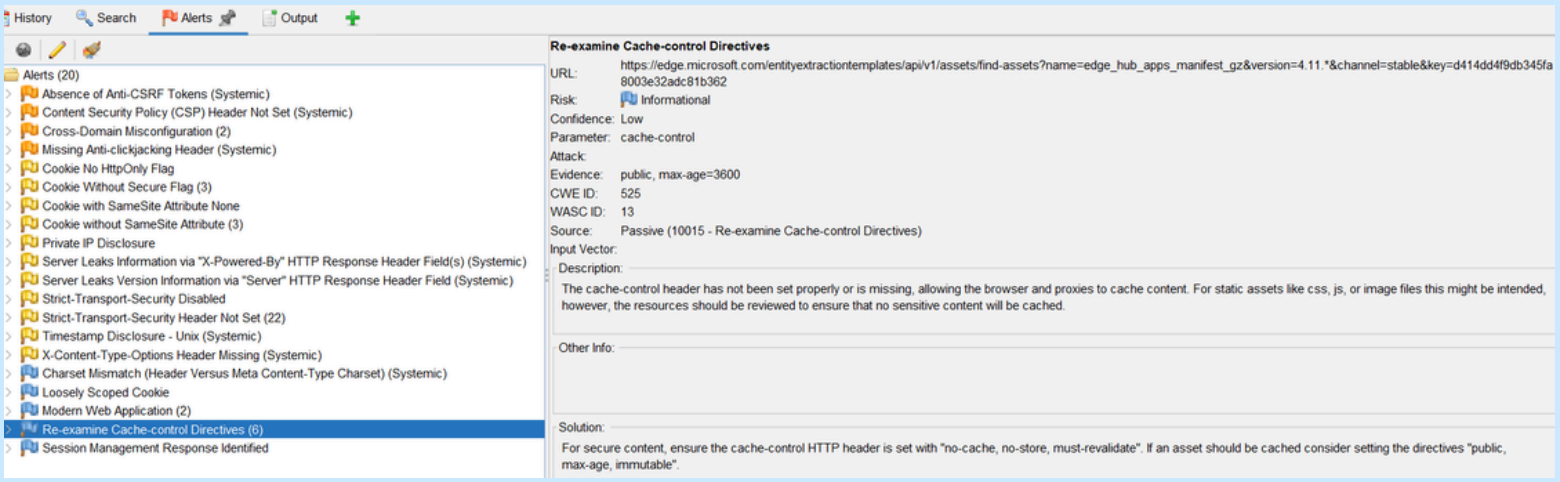
Risk Level:
Low

Impact:
Improper caching can lead to exposure of sensitive information such as session data, user credentials, or previously accessed pages. This increases the risk of unauthorized access, especially on shared or public systems.

Affected Configuration:
- Cache-Control header not properly configured
- Pragma header missing or not set correctly
- Sensitive responses may be cached by browsers or proxies

5. Cache Control Issues (Continued)

Evidence:



Remediation:

It is recommended to properly configure cache-related HTTP headers. Sensitive pages should include directives such as **Cache-Control: no-store, no-cache, must-revalidate** and **Pragma: no-cache** to prevent storage of sensitive data in browser or proxy caches.

6. Open Port Exposure

Description:
The network scan revealed that port 80 (HTTP) is open on the target server. Open ports indicate accessible services that may be targeted by attackers if not properly secured or monitored.

Risk Level:
Low

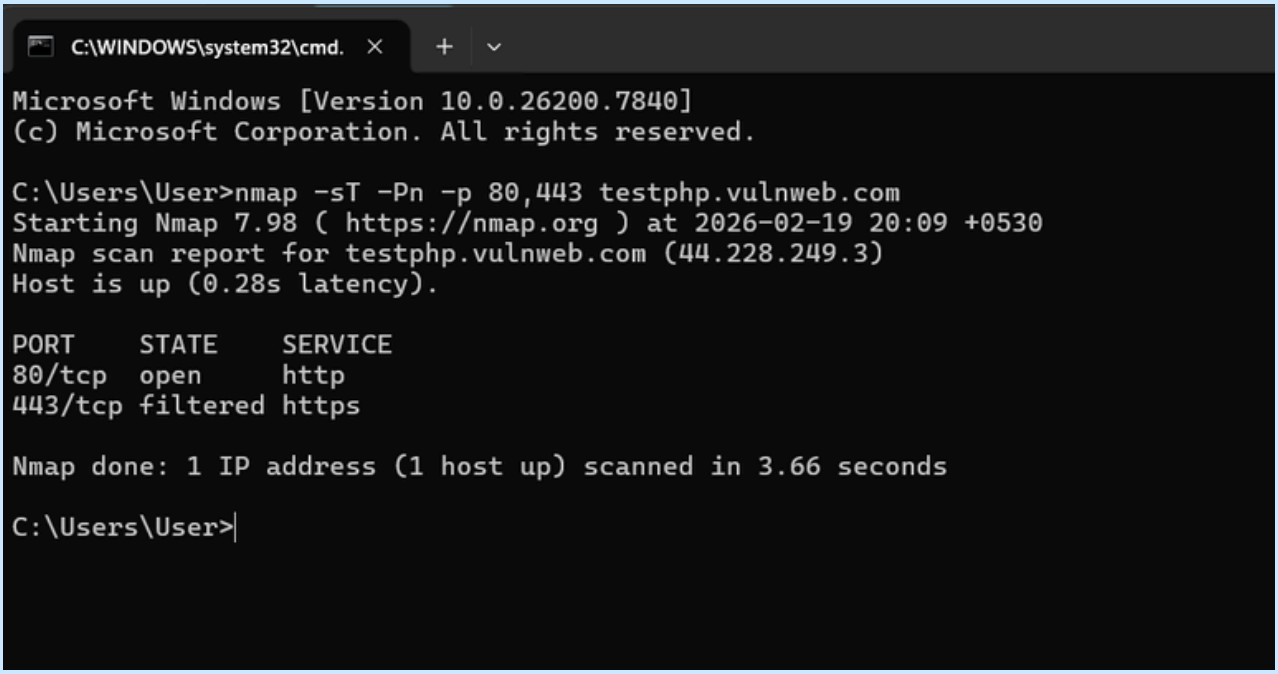
Impact:
An open port exposes a service running on the server, which could be exploited if vulnerabilities exist within that service. Attackers may use this information to perform further reconnaissance or launch targeted attacks.

Affected Area:

- Port 80 (HTTP service) is open
- Service accessible over the public network

6. Open Port Exposure (Continued)

Evidence:



Remediation:

It is recommended to restrict access to only necessary ports and services. Unused ports should be closed, and firewalls should be configured to limit access. Additionally, ensure that services running on open ports are regularly updated and securely configured.

Conclusion

This vulnerability assessment identified several security weaknesses in the target website http://testphp.vulnweb.com using non-intrusive and passive testing methods. The findings primarily include issues related to information disclosure, missing security headers, insecure cookie configurations, absence of CSRF protection, improper cache control, and open port exposure.

Most of the identified vulnerabilities fall under low to medium risk categories; however, they still pose potential security threats if left unaddressed. These weaknesses could allow attackers to gather sensitive information, exploit session data, or perform web-based attacks such as Cross-Site Scripting (XSS) and request forgery.

Overall, while the application serves as a test environment, the assessment highlights the importance of implementing proper security controls. Addressing the identified issues will significantly improve the security posture and resilience of the web application.

Recommendations

Based on the findings of this assessment, the following security improvements are strongly recommended:

1. Implement Security Headers

Configure essential HTTP security headers including:

- **Content-Security-Policy (CSP)**
- **X-Frame-Options**
- **Strict-Transport-Security (HSTS)**
- **X-Content-Type-Options**

These headers help protect against XSS, clickjacking, and other browser-based attacks.

2. Secure Cookie Configuration

Ensure all cookies use:

- **HttpOnly flag**
- **Secure flag**
- **Appropriate SameSite attribute**

This prevents client-side script access and reduces session hijacking risks.

3. Hide Server & Technology Information

Remove or disable unnecessary headers such as:

- **Server**
- **X-Powered-By**

This reduces information disclosure and limits reconnaissance opportunities.

4. Implement CSRF Protection

Enable anti-CSRF tokens for all state-changing requests to prevent unauthorized actions.

5. Improve Cache Control Policies

Configure proper cache directives (no-store, no-cache, must-revalidate) for sensitive content.

6. Restrict Open Ports & Services

Close unused ports and restrict access through firewall rules.

Limitations of Assessment

This vulnerability assessment was conducted using passive and non-intrusive techniques. As a result, certain limitations apply to the scope and depth of the findings.

- The assessment did not include active exploitation of vulnerabilities, so some issues may not have been fully verified.
- Only publicly accessible components of the website were tested; internal systems and authenticated areas were not assessed.
- Automated tools may produce false positives or may not detect all existing vulnerabilities.
- The assessment was performed within a limited time frame and may not reflect newly emerging vulnerabilities.
- Advanced security testing techniques such as penetration testing, fuzzing, or source code analysis were not performed.

Disclaimer

This vulnerability assessment report has been prepared for educational and internship purposes only. The analysis was conducted using passive and non-intrusive techniques on a publicly accessible test website.

No active exploitation, denial-of-service attacks, or unauthorized access attempts were performed during this assessment. All testing activities were carried out within ethical and permitted boundaries.

The findings presented in this report are based on the tools and methods used during the assessment and may not represent all possible vulnerabilities present in the application. Some findings may be subject to false positives or may require further validation.

This report should not be used for malicious purposes. The objective of this assessment is to demonstrate security analysis skills and promote awareness of common web application vulnerabilities.

This assessment was conducted on a deliberately vulnerable test application.

References

- OWASP Foundation. OWASP Top 10 Web Application Security Risks https://owasp.org/www-project-top-ten/
- OWASP ZAP Tool Documentation https://www.zaproxy.org/
- Nmap Official Documentation https://nmap.org/docs.html
- Acunetix Test Website http://testphp.vulnweb.com
- Mozilla Developer Network (MDN) – HTTP Security Headers https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers