



# Vulnerability Assessment Report

Target website: testphp.vulnweb.com

Assessment type: Read only/passive

Prepared By: Harish Saud

Role: Cybersecurity Intern

Date: January 20, 2026

# 1. Executive Summary

This Vulnerability Assessment was conducted to evaluate the security posture of the publicly accessible website **testphp.vulnweb.com** using **passive and non-intrusive testing techniques**. The objective of the assessment was to identify common configuration weaknesses, assess their potential business impact, and provide clear remediation guidance in a manner suitable for non-technical stakeholders.

The assessment was performed strictly within a **read-only scope**, focusing only on public-facing components of the application. No exploitation, authentication bypass, brute-force attempts, or denial-of-service activities were conducted during the engagement. The analysis relied on passive traffic inspection, configuration review, and service exposure analysis using industry-recognized tools.

During the assessment, a total of **seven security findings** were identified. These findings primarily relate to **missing security headers, absence of protective controls, and information disclosure through server configuration**. No **high-risk or critical vulnerabilities** were observed as part of this assessment; however, several **medium-risk issues** were identified that could increase the application's exposure to common web-based attacks if left unaddressed.

The most notable issues include the absence of a **Content Security Policy (CSP)**, missing **anti-clickjacking protections**, lack of **anti-CSRF mechanisms**, and disclosure of server software and version information. While these issues do not represent an immediate compromise of the system, they weaken the overall security posture and may assist attackers in executing client-side attacks or conducting targeted reconnaissance.

From a business perspective, these weaknesses may lead to **reduced user trust, increased attack surface, and higher risk of browser-based attacks**, such as clickjacking or malicious script execution. The identified issues are largely **configuration-level weaknesses** and can be mitigated through well-established security best practices with minimal operational impact.

It is strongly recommended that the organization prioritize the implementation of missing security headers, reduce unnecessary information disclosure, and periodically review server configurations to align with modern web security standards. Addressing these findings will significantly enhance the website's resilience against common threats and demonstrate a proactive approach to cybersecurity risk management.

## 2. Scope of Assessment

The scope of this assessment was limited to the **publicly accessible components** of the website **testphp.vulnweb.com**. The assessment was conducted under a **read-only scope**, ensuring no impact on system availability, integrity, or confidentiality.

### In Scope

- Public-facing web pages
- HTTP response headers and configurations
- Client-side behavior observable via passive analysis
- Service exposure and basic network visibility

### Out of Scope

- Authentication and authorization testing
- Login bypass or credential-based attacks
- Exploitation of vulnerabilities
- Brute-force attacks, fuzzing, or DoS testing
- Access to administrative or restricted resources

### 3. Methodology & Tools Used

#### Methodology

The assessment followed a structured approach:

1. Target identification and scope definition
2. Passive vulnerability detection
3. Security header and configuration analysis
4. Service exposure analysis
5. Risk classification and reporting

#### Tools Used

- **OWASP ZAP (Passive Scan)** – OWASP Zed Attack Proxy (ZAP) is an open-source web application security tool used to identify common security issues by passively analyzing HTTP and HTTPS traffic between a browser and a web application. It helps detect misconfigurations such as missing security headers, insecure cookies, and information disclosure without actively attacking the target, making it suitable for ethical, read-only security assessments.
- **Nmap** – Nmap (Network Mapper) is an open-source network scanning tool used to identify publicly exposed services and open ports on a target system. In this assessment, Nmap was used in a non-intrusive manner to detect accessible services and server version information, helping evaluate basic network exposure without attempting exploitation or disrupting the target system.
- **Browser Developer Tools** – Browser Developer Tools are built-in features of modern web browsers that allow security analysts to inspect client-side behavior of web applications. In this assessment, they were used to review HTTP response headers, cookies, and security configurations directly from the browser, helping identify missing security controls and misconfigurations without interacting with or altering the application.

## 4. Risk Rating Criteria

Risk Level	Description
High	Issues that may directly lead to data compromise or system abuse
Medium	Security weaknesses that increase attack surface
Low	Information disclosure or best-practice gaps
Informational	Observations with no direct security impact

## 5. Summary of Findings

Vulnerability	Risk Level
Absence of Anti-CSRF Tokens	Medium
Content Security Policy (CSP) Not Set	Medium
Missing Anti-Clickjacking Header	Medium
Server Version Disclosure	Low
X-Content-Type-Options Header Missing	Low
X-Powered-By Header Disclosure	Low
Charset Mismatch	Low

## 6. Detailed Findings

### Finding 1: Absence of Anti-CSRF Tokens

**Risk Level:** Medium

**Source:** OWASP ZAP

#### Description:

The application does not implement Anti-Cross-Site Request Forgery (CSRF) tokens to protect against unauthorized requests initiated from external websites. Without CSRF protection, an attacker may be able to trick authenticated users into performing unintended actions, which could impact application integrity and user trust.

#### Business Impact:

The absence of Anti-CSRF tokens can allow attackers to manipulate user actions without their knowledge, potentially leading to unauthorized changes, misuse of application features, and loss of user confidence. This may result in reputational damage, customer dissatisfaction, and increased security risk for the business.

#### Evidence:

The screenshot shows the OWASP ZAP interface with the 'Alerts' tab selected. A single alert is highlighted: 'Absence of Anti-CSRF Tokens'. The alert details are as follows:

- URL:** http://testphp.vulnweb.com/
- Risk:** Medium
- Confidence:** Low
- Parameter:** (empty)
- Attack:** (empty)
- Evidence:** <form action="search.php?test=query" method="post">
- CWE ID:** 352
- WASC ID:** 9
- Source:** Passive (10202 - Absence of Anti-CSRF Tokens)
- Input Vector:** (empty)
- Description:** No Anti-CSRF tokens were found in a HTML submission form.  
A cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim. The underlying cause is application functionality using
- Other Info:** No known Anti-CSRF token [anticsrf, CSRFToken, \_\_RequestVerificationToken, csrfmiddlewaretoken, authenticity\_token, OWASP\_CSRFTOKEN, anoncsrf, csrf\_token, \_\_csrfSecret, \_\_csrf\_magic, CSRF, \_\_token, \_\_csrf\_token, \_\_csrfToken] was found in the following HTML form: [Form 1: "goButton" "searchFor" ].
- Solution:** Phase: Architecture and Design  
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
- Reference:** [https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)  
<https://cwe.mitre.org/data/definitions/352.html>
- Alert Tags:**

Key	Value
OWASP_2021_A01	<a href="https://owasp.org/Top10/A01_2021-Broken_Access_Control/">https://owasp.org/Top10/A01_2021-Broken_Access_Control/</a>
POLICY_QA_STD	
POLICY_PENTEST	
SYSTEMIC	
WSTG-v42-SESS-05	<a href="https://www.zaproxy.org/docs/desktop/addons/common-library/">https://www.zaproxy.org/docs/desktop/addons/common-library/</a> <a href="https://owasp.org/www-project-web-security-testing-guide/v42/">https://owasp.org/www-project-web-security-testing-guide/v42/</a>

#### Recommendation:

Implement Anti-CSRF protection by generating unique, unpredictable tokens for all state-changing requests. Ensure these tokens are securely associated with user sessions and validated on the server side. Apply CSRF protection consistently across all forms and sensitive endpoints. Regularly review and test CSRF controls to ensure they remain effective against evolving threats.

## Finding 2: Content Security Policy (CSP) Header Not Set

**Risk Level:** Medium

**Source:** OWASP ZAP

### Description:

The website does not define a Content Security Policy (CSP) header to control which external resources the browser is allowed to load. Without CSP, the browser has no restrictions on executing scripts or loading content from untrusted sources. This increases the likelihood of client-side attacks such as malicious script injection and data manipulation.

### Business Impact:

The absence of a Content Security Policy (CSP) header increases the risk of malicious scripts being executed in users' browsers. This can lead to data theft, session compromise, and reduced trust in the website. For the business, such attacks may result in reputational damage, potential compliance issues, and loss of customer confidence.

### Evidence:

Content Security Policy (CSP) Header Not Set

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

Risk: Medium

Confidence: High

Parameter:

Attack:

Evidence:

CWE ID: 693

WASC ID: 15

Source: Passive (10038 - Content Security Policy (CSP) Header Not Set)

Alert Reference: 10038-1

Input Vector:

Description:

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content.

Other Info:

Solution:

Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

Reference:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CSP  
https://cheatsheetseries.owasp.org/cheatsheets/Content\_Security\_Policy\_Cheat\_Sheet.html  
https://www.w3.org/TR/CSP/

Key	Value
CWE-693	https://cwe.mitre.org/data/definitions/693.html
OWASP_2021_A05	https://owasp.org/Top10/A05_2021-Security_Misconfiguration/
OWASP_2017_A06	https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration/
POLICY_QA_STD	
POLICY_PENTEST	

### Recommendation:

Implement a strong Content Security Policy (CSP) header to restrict the execution of scripts, styles, and other resources to trusted sources only. Define allowed domains explicitly and block inline and unauthorized scripts wherever possible. Test the CSP configuration in report-only mode before full enforcement to avoid breaking functionality. Regularly review and update the policy as the application evolves.

## Finding 3: Missing Anti-Clickjacking Header

**Risk Level:** Medium

**Source:** OWASP ZAP

### Description:

The website does not include an anti-clickjacking protection header, such as **X-Frame-Options** or the equivalent **frame-ancestors** directive in Content Security Policy. Without this protection, the application can be embedded within malicious iframes, potentially tricking users into interacting with hidden or disguised content.

### Business Impact:

The absence of anti-clickjacking protection can allow attackers to deceive users into performing unintended actions on the website. This may lead to unauthorized transactions, misuse of application features, and erosion of user trust. For the business, such incidents can result in reputational damage and increased security risk.

### Evidence:

The screenshot shows the OWASP ZAP tool's interface. On the left, there is a tree view under 'Alerts (7)' with several items expanded, including 'Missing Anti-Clickjacking Header (Systemic)'. The main pane displays detailed information about this specific finding:

- Missing Anti-Clickjacking Header**
- URL:** http://testphp.vulnweb.com/
- Risk:** Medium
- Confidence:** Medium
- Parameter:** x-frame-options
- Attack:** Clickjacking
- Evidence:** X-Powered-By, Server, X-Content-Type-Options, Charset Mismatch
- CWE ID:** 1021
- WASC ID:** 15
- Source:** Passive (10020 - Anti-clickjacking Header)
- Alert Reference:** 10020-1
- Input Vector:** None
- Description:** The response does not protect against 'Clickjacking' attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.
- Other Info:** None
- Solution:** Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use
- Reference:** <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/X-Frame-Options>

**Alert Tags:**

Key	Value
OWASP_2021_A05	<a href="https://owasp.org/Top10/A05_2021-Security_Misconfiguration/">https://owasp.org/Top10/A05_2021-Security_Misconfiguration/</a>
POLICY_QA_STD	
POLICY_PENTEST	
CWE-1021	<a href="https://cwe.mitre.org/data/definitions/1021.html">https://cwe.mitre.org/data/definitions/1021.html</a>
SYSTEMIC	<a href="https://www.zaproxy.org/docs/desktop/addons/common-library...">https://www.zaproxy.org/docs/desktop/addons/common-library...</a>

### Recommendation:

Configure the **X-Frame-Options** header to restrict how the website can be embedded in iframes. Alternatively, implement the **frame-ancestors** directive within a Content Security Policy for more flexible control. Apply this protection consistently across all pages to prevent clickjacking attacks. Regularly test and review header configurations to ensure continued effectiveness.

## Finding 4: Server Version Disclosure

**Risk Level:** Low

**Source:** Nmap & OWASP ZAP

### Description:

The server discloses its software type and version (nginx 1.19.0). The web server discloses detailed version information through HTTP response headers, such as the **Server** or **X-Powered-By** fields. Exposing this information allows attackers to identify the underlying server technology and version, which may help them target known vulnerabilities associated with that software.

### Business Impact:

The disclosure of server software type and version information makes it easier for attackers to identify known vulnerabilities specific to the disclosed technology. This increases the risk of targeted attacks and reduces the overall security posture of the website. For the business, such exposure can lead to higher chances of exploitation and potential service disruption.

### Evidence:

The screenshot shows the OWASP ZAP application window. On the left, there's a sidebar with icons for Alerts, Issues, and Tools. Below that, under 'Alerts (7)', several items are listed: Absence of Anti-CSRF Tokens (Systemic), Content Security Policy (CSP) Header Not Set (Systemic), Missing Anti-clickjacking Header (Systemic), Server Leaks Information via 'X-Powered-By' HTTP Response Header Field(s) (Systemic), Server Leaks Version Information via "Server" HTTP Response Header Field (Systemic), X-Content-Type-Options Header Missing (Systemic), and Charset Mismatch (Header Versus Meta Content-Type Charset) (Systemic). The main panel displays a detailed alert for 'Server Leaks Version Information via "Server" HTTP Response Header Field'. It includes fields for URL (http://testphp.vulnweb.com), Risk (Low), Confidence (High), Parameter, Attack, Evidence (nginx/1.19.0), CWE ID (497), WASC ID (13), Source (Passive (10036 - HTTP Server Response Header)), Alert Reference (10036-2), 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, Solution (Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.), Reference (links to https://httpd.apache.org/docs/current/mod/core.html#servertokens, https://learn.microsoft.com/en-us/previous-versions/msp-n-p/ff648552(v=pandp.10), and https://www.troyhunt.com/shhh-dont-let-your-response-headers/), and Alert Tags. The 'Alert Tags' table has two rows: OWASP\_2021\_A05 and OWASP\_2017\_A06, each with a 'Key' and 'Value' column.

### Recommendation:

Configure the web server to suppress or remove detailed version information from HTTP response headers. Ensure that headers such as **Server** and **X-Powered-By** reveal minimal or no technology details. Regularly update and patch the server software to reduce exposure to known vulnerabilities. Review server configuration periodically to maintain secure information disclosure practices.

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

**Risk Level:** Low

**Source:** OWASP ZAP

### Description:

The website does not include the **X-Content-Type-Options** security header in its HTTP responses. Without this header, browsers may attempt to interpret files as a different content type than intended, which can increase the risk of certain client-side attacks such as MIME-type sniffing.

### Business Impact:

The absence of the X-Content-Type-Options header increases the risk of malicious content being incorrectly interpreted by users' browsers. This can potentially lead to script execution, data exposure, and a reduced level of trust in the website. For the business, such risks may result in security incidents and reputational damage.

### Evidence:

The screenshot shows the OWASP ZAP interface with the following details:

- Alerts (7)**: A list of findings, with the last one selected.
- X-Content-Type-Options Header Missing**: The selected alert.
- URL:** <http://testphp.vulnweb.com/>
- Risk:** Low
- Confidence:** Medium
- Parameter:** x-content-type-options
- Attack:** None
- Evidence:** None
- CWE ID:** 693
- WASC ID:** 15
- Source:** Passive (10021 - X-Content-Type-Options Header Missing)
- Input Vector:** None
- Description:** The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type. Current (early 2014) and legacy versions of Firefox will use the declared content type.
- Other Info:** This issue still applies to error type pages (401, 403, 500, etc.) as those pages are often still affected by injection issues, in which case there is still concern for browsers sniffing pages away from their actual content type. At "High" threshold this scan rule will not alert on client or server error responses.
- Solution:** Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages. If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all.
- Reference:**
  - [https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941\(v=vs.85\)](https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85))
  - [https://owasp.org/www-community/Security\\_Headers](https://owasp.org/www-community/Security_Headers)
- Alert Tags:**

Key	Value
CWE-693	<a href="https://cwe.mitre.org/data/definitions/693.html">https://cwe.mitre.org/data/definitions/693.html</a>
OWASP_2021_A05	<a href="https://owasp.org/Top10/A05_2021-Security_Misconfiguration/">https://owasp.org/Top10/A05_2021-Security_Misconfiguration/</a>
OWASP_2017_A06	<a href="https://owasp.org/www-project-top-ten/2017/A6_2017-Securit/">https://owasp.org/www-project-top-ten/2017/A6_2017-Securit/</a>
POLICY_QA_STD	
POLICY_PENTEST	

### Recommendation:

Configure the web server to include the **X-Content-Type-Options: nosniff** header in all HTTP responses. This prevents browsers from performing MIME-type sniffing and enforces proper content handling. Apply the header consistently across all application pages and resources. Regularly verify security headers as part of routine security reviews.

## Finding 6: X-Powered-By Header Disclosure

**Risk Level:** Low

**Source:** OWASP ZAP

### Description:

The website exposes the **X-Powered-By** HTTP response header, which reveals information about the underlying technology used by the application. Disclosing such implementation details provides unnecessary insight into the technology stack, which attackers can leverage to identify potential weaknesses or known vulnerabilities.

### Business Impact:

The disclosure of technology details through the X-Powered-By header increases the likelihood of targeted attacks against known vulnerabilities in the underlying platform. This exposure weakens the overall security posture of the website and may lead to increased risk of exploitation. For the business, it can result in security incidents and loss of user confidence.

### Evidence:

The screenshot shows the OWASP ZAP interface with the title bar "Alcibi". On the left, there's a toolbar with icons for alerts, filters, and other functions. Below the toolbar, a sidebar lists "Alerts (7)" with several items expanded, including "Absence of Anti-CSRF Tokens (Systemic)", "Content Security Policy (CSP) Header Not Set (Systemic)", "Missing Anti-clickjacking Header (Systemic)", and "Server Leaks Information via \"X-Powered-By\" HTTP Response Header Field(s) (Systemic)". The main panel is titled "Server Leaks Information via \"X-Powered-By\" HTTP Response Header Field(s)". It displays the following details:

- URL:** http://testphp.vulnweb.com/
- Risk:** Low
- Confidence:** Medium
- Parameter:**
- Attack:**
- 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:**
- Solution:** Ensure that your web server, application server, load balancer, etc. is configured to suppress "X-Powered-By" headers.
- Reference:**
  - [https://owasp.org/www-project-web-security-testing-guide/v42/4-Web\\_Application\\_Security\\_Testing/01-Information\\_Gathering/08-Fingerprint\\_Web\\_Application\\_Framework](https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/08-Fingerprint_Web_Application_Framework)
  - <https://www.troyhunt.com/shhh-dont-let-your-response-headers/>
- Alert Tags:**

Key	Value
OWASP_2021_A01	<a href="https://owasp.org/Top10/A01_2021-Broken_Access_Control/">https://owasp.org/Top10/A01_2021-Broken_Access_Control/</a>
WSTG-v42-INFO-08	<a href="https://owasp.org/www-project-web-security-testing-guide/v42/">https://owasp.org/www-project-web-security-testing-guide/v42...</a>
POLICY_QA_STD	
POLICY_PENTEST	
SYSTEMIC	<a href="https://www.zaproxy.org/docs/desktop/addons/common-library...">https://www.zaproxy.org/docs/desktop/addons/common-library...</a>

### Recommendation:

Disable or remove the **X-Powered-By** header from server responses to limit exposure of internal technology details. Ensure only necessary information is included in HTTP headers. Keep all underlying technologies updated and securely configured. Periodically review response headers to prevent unintended information disclosure.

## Finding 7: Charset Mismatch

**Risk Level:** Low

**Source:** OWASP ZAP

### Description:

A charset mismatch was identified between the HTTP response header and the HTML meta Content-Type declaration. When character encoding is inconsistently defined, browsers may incorrectly interpret page content, which can lead to unexpected behavior and potential security issues such as improper content rendering or injection risks.

### Business Impact:

A charset mismatch can cause browsers to misinterpret page content, leading to display issues and inconsistent user experience. In some cases, it may increase the risk of client-side security issues, such as improper handling of user input. For the business, this can reduce application reliability and user trust.

### Evidence:

The screenshot shows the OWASP ZAP interface with the 'Alerts' tab selected. A single alert is highlighted: 'Charset Mismatch (Header Versus Meta Content-Type Charset)'. The alert details are as follows:

- URL:** http://testphp.vulnweb.com/
- Risk:** Informational
- Confidence:** Low
- Parameter:** (empty)
- Attack:** (empty)
- Evidence:** (empty)
- CWE ID:** 436
- WASC ID:** 15
- Source:** Passive (90011 - Charset Mismatch)
- Alert Reference:** 90011-1
- Input Vector:** (empty)
- Description:** This check identifies responses where the HTTP Content-Type header declares a charset different from the charset defined by the body of the HTML or XML. When there's a charset mismatch between the HTTP header and content body Web browsers can be forced into an undesirable content-sniffing mode to determine the content's correct character set.
- Other Info:** There was a charset mismatch between the HTTP Header and the META content-type encoding declarations: [UTF-8] and [iso-8859-2] do not match.
- Solution:** Force UTF-8 for all text content in both the HTTP header and meta tags in HTML or encoding declarations in XML.
- Reference:** [https://code.google.com/archive/p/browsersec/wikis/Part2.wiki#Character\\_set\\_handling\\_and\\_detection](https://code.google.com/archive/p/browsersec/wikis/Part2.wiki#Character_set_handling_and_detection)
- Alert Tags:**

Key	Value
CWE-436	<a href="https://cwe.mitre.org/data/definitions/436.html">https://cwe.mitre.org/data/definitions/436.html</a>
POLICY_QA_STD	
POLICY_PENTEST	
SYSTEMIC	<a href="https://www.zaproxy.org/docs/desktop/addons/common-library...">https://www.zaproxy.org/docs/desktop/addons/common-library...</a>

### Recommendation:

Ensure that the character encoding defined in the HTTP response header matches the encoding specified in the HTML meta Content-Type tag. Standardize the application to use a single, secure charset such as UTF-8. Apply consistent encoding settings across all pages and resources. Regularly validate response headers and page content for encoding consistency.

## 7. Network Exposure Analysis (Nmap)

### Open Ports Identified:

- Port 80 (HTTP)

### Service Detected:

- nginx 1.19.0

**Risk Level:** Low

### Evidence:

```
ubuntu@ip-172-31-22-213:~$ nmap -sV -Pn --reason testphp.vulnweb.com
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-16 05:36 UTC
Nmap scan report for testphp.vulnweb.com (44.228.249.3)
Host is up, received user-set (0.055s latency).
rDNS record for 44.228.249.3: ec2-44-228-249-3.us-west-2.compute.amazonaws.com
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON  VERSION
80/tcp    open  http    syn-ack nginx 1.19.0

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.07 seconds
```

## 8. Overall Recommendations

Overall, it is recommended that the website improve its security posture by implementing all missing HTTP security headers to enhance protection against common client-side attacks. Unnecessary disclosure of server and technology information should be minimized to reduce the risk of targeted attacks. Security configurations should be applied consistently across all public-facing pages and reviewed regularly to ensure they remain effective. The server and underlying technologies should be kept up to date with the latest security patches and best practices. Additionally, periodic vulnerability assessments and security reviews should be conducted to proactively identify and address emerging risks, helping maintain system reliability, protect user data, and preserve business trust.

## 9. Conclusion

This vulnerability assessment was conducted to evaluate the security posture of the target website using passive and non-intrusive techniques. The assessment identified several security misconfigurations, primarily related to missing HTTP security headers and information disclosure through server responses. While no critical vulnerabilities requiring immediate exploitation were observed, these issues may increase the risk of client-side attacks and targeted threats if left unaddressed.

Implementing the recommended security controls will help strengthen the website's overall security, reduce exposure to common web-based risks, and improve resilience against potential attacks. Regular security reviews and vulnerability assessments are advised to ensure continued protection as the application evolves. This assessment demonstrates the importance of proactive security measures in maintaining user trust and supporting long-term business objectives.

## 10. Appendix (Evidence)

- OWASP ZAP Alerts Summary – During the passive security assessment conducted using OWASP ZAP, multiple security misconfigurations were identified related to missing or improperly configured HTTP security headers and information disclosure. The findings primarily indicate weaknesses in browser-side security controls, which may increase the risk of client-side attacks such as clickjacking, cross-site request forgery, and malicious script execution. No active exploitation was performed, and all alerts were generated through passive analysis of server responses. Overall, the identified issues suggest opportunities to strengthen the website's security posture through proper header configuration and secure server settings.

The screenshot shows the OWASP ZAP interface with the 'Alerts' tab selected. The sidebar on the left shows a folder icon followed by 'Alerts (7)'. The main pane lists seven systemic alerts:

- > Absence of Anti-CSRF Tokens (Systemic)
- > Content Security Policy (CSP) Header Not Set (Systemic)
- > Missing Anti-clickjacking Header (Systemic)
- > Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s) (Systemic)
- > Server Leaks Version Information via "Server" HTTP Response Header Field (Systemic)
- > X-Content-Type-Options Header Missing (Systemic)
- > Charset Mismatch (Header Versus Meta Content-Type Charset) (Systemic)

- Individual ZAP Findings – The OWASP ZAP passive scan identified several individual security findings related to HTTP response headers and application configuration. Each finding represents a specific area where security controls are either missing or not optimally configured. While none of the issues indicate immediate exploitation, they collectively increase the website's exposure to client-side attacks and information disclosure. The findings have been assessed individually based on potential impact and likelihood, with clear recommendations provided to help improve the overall security posture of the application.
- Request:

The screenshot shows the OWASP ZAP Request/Response tool with the 'Request' tab selected. The 'Header' section shows the following fields:

```
Header: Text
GET http://testphp.vulnweb.com/ HTTP/1.1
host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=0, i
```

- Response:

HTTP/1.1 200 OK  
Server: nginx/1.19.0  
Date: Fri, 16 Jan 2026 04:43:50 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: 4958

- Zap Sites Tree:

File Edit View Analyse Report Tools Import Export Online Help  
Standard Mode < Quick Start Request Response Requester +  
Header: Text Body: Text |

Sites +  
Contexts Default Context  
Sites http://testphp.vulnweb.com  
  GET:/  
  AJAX  
    GET:artists.php  
    GET:cart.php  
    GET:categories.php  
    GET:disclaimer.php  
    GET:favicon.ico  
    GET:guestbook.php  
  images  
    GET:index.php  
    GET:login.php  
    GET:style.css  
    GET:userinfo.php

HTTP/1.1 200 OK  
Server: nginx/1.19.0  
Date: Fri, 16 Jan 2026 04:43:50 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: 4958