Vulnerability Assessment Report

Date: October 17, 2025 (Asia/Dhaka)

Target: testphp.vulnweb.com

Prepared for: Masud Rana (user)

Executive Summary

This document summarizes vulnerabilities discovered during interactive testing of the demo web application **testphp.vulnweb.com** on October 17, 2025. The site is intentionally vulnerable (Acunetix demo). Findings below include confirmed and probable issues: persistent (stored) XSS, reflected XSS, DOM-based XSS, potential SQL injection, open redirect, CSRF absence, informational header leakage, missing security headers, and outdated PHP. All testing was non-destructive and limited to information provided by the user via HTTP responses.

Summary of Findings

Vulnerability	Location / Parameter	Evidence (excerpt)	Severity	Recommended Remediation
Stored Cross-Site Scripting (XSS)	`/guestbook.php` (POST name/message)	`<script>alert("stored")</script>` stored and executed when viewing guestbook	**High**	Escape output (htmlspecialchars), sanitize input, use CSP, validate input length
Reflected Cross-Site Scripting (XSS)	`/search.php` (searchFor GET/POST)	`<h2 id='pageName'>searched for: postvalue</h2>` reflected unescaped	**High**	Context-aware output encoding, CSP, input validation
DOM-based XSS	`/AJAX/index.php` (XML responses used in innerHTML)	`cd.innerHTML = inner;` with unescaped XML node values	**High**	Escape/sanitize XML-derived data before inserting into DOM; use textContent or templating; validate/parse XML safely
Potential SQL Injection	`/search.php` and `/userinfo.php` (searchFor, uname, pass)	Demo site notes and unsanitized inputs; typical payloads accepted	**High**	Use prepared statements/parameterized queries; input validation; least privilege DB user
Open Redirect	`/login.php?redirect=...`	`redirect=https://evil.example` parameter accepted	**High**	Validate allowlist of redirect targets or use relative paths only; canonicalize and validate
CSRF Protection Missing	Forms across site (login, guestbook add)	Forms have no CSRF tokens (e.g., login form posts to userinfo.php)	**Medium**	Implement CSRF tokens, SameSite cookies, and verify origin/header
Information Disclosure (Headers)	HTTP response headers	`Server: nginx/1.19.0; X-Powered-By: PHP/5.6.40-...`	**Medium**	Hide server and X-Powered-By headers; upgrade PHP and apply patches
Missing Security Response Headers	All pages	No CSP, X-Frame-Options, X-Content-Type-Options, HSTS observed	**Medium**	Add CSP, X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security where applicable

Detailed Findings

1. Stored Cross-Site Scripting (XSS)

Location/Parameter: /guestbook.php (POST name/message)
Evidence (excerpt):
`alert("stored") stored and executed when viewing guestbook`
Severity: High
Recommended Remediation: Escape output (htmlspecialchars), sanitize input, use CSP, validate input length

Reflected Cross-Site Scripting (XSS)

Location/Parameter: /search.php (searchFor GET/POST)
Evidence (excerpt):
`searched for: postvalue reflected unescaped`
Severity: High
Recommended Remediation: Context-aware output encoding, CSP, input validation

2. DOM-based XSSLocation/Parameter: /AJAX/index.php (XML responses used in innerHTML) Evidence (excerpt):

`cd.innerHTML = inner; with unescaped XML node values`

Severity: High

Recommended Remediation: Escape/sanitize XML-derived data before inserting into DOM; use textContent or templating; validate/parse XML safely

3. Potential SQL Injection

Location/Parameter: /search.php and /userinfo.php (searchFor, uname, pass)

Evidence (excerpt):

`Demo site notes and unsanitized inputs; typical payloads accepted`

Severity: High

Recommended Remediation: Use prepared statements/parameterized queries; input validation; least privilege DB user

4. Open Redirect

Location/Parameter: /login.php?redirect=...

Evidence (excerpt):

`redirect=https://evil.example parameter accepted`

Severity: High

Recommended Remediation: Validate allowlist of redirect targets or use relative paths only; canonicalize and validate

5. CSRF Protection Missing

Location/Parameter: Forms across site (login, guestbook add)

Evidence (excerpt):

`Forms have no CSRF tokens (e.g., login form posts to userinfo.php)`

Severity: Medium

Recommended Remediation: Implement CSRF tokens, SameSite cookies, and verify origin/header

6. Information Disclosure (Headers)

Location/Parameter: HTTP response headers

Evidence (excerpt):

`Server: nginx/1.19.0; X-Powered-By: PHP/5.6.40-...`

Severity: Medium

Recommended Remediation: Hide server and X-Powered-By headers; upgrade PHP and apply patches

7. Missing Security Response Headers

Location/Parameter: All pages

Evidence (excerpt):

`No CSP, X-Frame-Options, X-Content-Type-Options, HSTS observed`

Severity: Medium

Recommended Remediation: Add CSP, X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security where applicable

Proof-of-Concept (PoC) Notes

- Stored XSS (guestbook): POST `name=attacker&message;=alert(1)` then view /guestbook.php to observe execution. - Reflected XSS (search): POST/GET `searchFor=alert(1)` to /search.php and observe reflected output. - DOM XSS (AJAX): Intercept XML responses (artists.php, titles.php) and inject script-bearing node values; page uses innerHTML. - SQLi: Test `uname=' OR '1'='1` on login form or use sqlmap for automated testing (authorized testing only). Notes: Only test on authorized targets. This report is informational and intended for authorized security

Remediation Checklist (Suggested Priorities)

1. Patch and upgrade PHP to a supported version; harden server configuration.
2. Add global security headers (CSP, HSTS, X-Frame-Options, X-Content-Type-Options).

3. Fix XSS: apply context-aware output encoding; sanitize input; use CSP as defense-in-depth.
4. Fix SQLi: parameterize all DB queries; use ORM or prepared statements.
5. Implement CSRF protections for all state-changing forms.
6. Remove unnecessary disclosure headers and debug output in production.
7. Review AJAX/XML handling: disable external entity resolution, sanitize XML content, avoid innerHTML.
8. Perform a full authenticated security test after fixes.

Appendix: Captured Headers & Sample Requests

Sample observed response headers (excerpt): Server: nginx/1.19.0 X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1 Content-Type: text/html; charset=UTF-8

Sample requests used during testing (non-destructive samples): curl -s -X POST 'http://testphp.vulnweb.com/guestbook.php' -d 'name=attacker&message;=alert(1)' curl -s -X POST 'http://testphp.vulnweb.com/search.php?test=query' -d 'searchFor=alert(1)' curl -s 'http://testphp.vulnweb.com/AJAX/index.php' # review JS for DOM XSS curl -s 'http://testphp.vulnweb.com/login.php?redirect=https://evil.example'

End of report

Prepared by: Masud Rana