

# PCI DSS Web Application Vulnerability Assessment Report

**Target:** <https://wp.98-89-111-197.sslip.io/>

**Assessor:** Tawan Perry – PCI Readiness & Security Consultant

**Tool Used:** OWASP ZAP 2.16.1

**Date:** November 19, 2025

---

## 1. Executive Summary

A web application vulnerability scan was conducted against the WordPress-based site at **wp.98-89-111-197.sslip.io** using OWASP ZAP. The scan identified a total of **16 security alerts**, including several **high-risk vulnerabilities** that impact PCI DSS compliance, data security, and overall site integrity.

These findings represent typical weaknesses found in WordPress environments and should be addressed as part of a PCI DSS 6.x and 11.x compliance program, particularly:




- **Requirement 6.5 (secure coding practices)**
- **Requirement 6.6 (web application protection)**
- **Requirement 11.3 (penetration testing)**

Immediate attention is recommended for vulnerabilities involving session security, missing headers, and potential injection points.

---

## 2. Risk Summary

Risk Level	Number of Alerts	PCI Impact
 <b>High</b>	4	Critical non-compliance / exploitation risk

 <b>Medium</b>	7	Moderate PCI compliance failures
 <b>Low</b>	3	Best-practice improvements
 <b>Informational</b>	2	Minimal risk; recommended for hygiene

---

## 3. High-Risk Findings

These require **immediate remediation**.

---

### 3.1 SQL Injection – SQLite (Time Based)

**Severity:** High

**PCI DSS Mapping:** 6.5.1, 11.3

**Description:**

ZAP detected potential SQL injection behavior via time-based responses. This indicates the application may be vulnerable to malicious queries.

**Impact:**

Attackers could manipulate database queries, extract data, or compromise authentication.

**Recommendations:**

- Enable prepared statements and parameterized queries.
  - Validate and sanitize all user inputs.
  - Implement a WAF rule or plugin (e.g., Wordfence, Sucuri).
  - Conduct a manual penetration test to confirm exploitability.
- 

### 3.2 Absence of Anti-CSRF Tokens

**Severity:** High

**PCI DSS Mapping:** 6.5.9

**Description:**

Forms are missing CSRF protection, allowing attackers to perform unauthorized actions on behalf of users.

**Impact:**

Malicious websites could execute actions using a logged-in administrator session.

**Recommendations:**

- Ensure all forms include CSRF tokens.
  - Enable WordPress security plugins enforcing CSRF protections.
  - Use nonces (`wp_create_nonce`) for all POST requests.
- 

### 3.3 Cookie Without HttpOnly Flag

**Severity:** High

**PCI DSS Mapping:** 6.5, 6.5.10

**Description:**

Cookies can be accessed by client-side JavaScript, increasing risk of session theft through XSS.

**Recommendations:**

- Add `HttpOnly` to all session cookies.
  - Adjust WordPress configuration or use a security plugin that forces secure cookies.
- 

### 3.4 Cookie Without Secure Flag

**Severity:** High

**PCI DSS Mapping:** 4.1, 6.5.10

**Description:**

Cookies are being transmitted over non-secure channels (or ZAP detected no Secure flag).

**Recommendations:**

- Force HTTPS for all cookies.

- Add `define('FORCE_SSL_ADMIN', true);` to wp-config.php.
  - Enable HSTS support.
- 

## 4. Medium-Risk Findings

These are common across small business and WordPress sites.

---

### 4.1 Missing Content Security Policy (CSP)

**Severity:** Medium

**PCI Mapping:** 6.5.7

**Description:**

Site does not define a CSP header.

**Impact:**

Higher risk of XSS attacks and loading unauthorized scripts.

**Recommendations:**

- Add a Content-Security-Policy header restricting sources.
  - Begin with `Content-Security-Policy: default-src 'self';`.
- 

### 4.2 Missing Anti-Clickjacking Header (X-Frame-Options)

**Severity:** Medium

**PCI Mapping:** 6.5.1

**Description:**

Site can be loaded inside iframes by attackers (clickjacking).

**Recommendation:**

Add:

`X-Frame-Options: SAMEORIGIN`

---

### 4.3 Missing Strict-Transport-Security (HSTS)

**Severity:** Medium

**PCI Mapping:** 4.1

**Recommendation:**

Enable HSTS via:

`Strict-Transport-Security: max-age=31536000; includeSubDomains`

---

### 4.4 X-Content-Type-Options Missing

**Severity:** Medium

**PCI Mapping:** 6.5

**Fix:**

Add:

`X-Content-Type-Options: nosniff`

---

### 4.5 Charset Mismatch

**Description:** Charset declared in HTML does not match header.

**Fix:**

Ensure both HTML and server headers specify UTF-8.

---

### 4.6 Cookie Poisoning

**Description:** Cookies are vulnerable to manipulation.

**Fix:**

Digitally sign cookies or store critical session data server-side.

---

### 4.7 Information Disclosure – Suspicious Comments

**Description:** HTML comments may reveal sensitive hints.

**Fix:**

Remove developer comments from source code.

---



## 5. Low-Risk Findings

- Cache-control directive issues
  - Modern web app hints
  - Re-examine caching behavior
- These are low risk but recommended for tightening security.
- 



## 6. Informational Alerts

Not harmful but useful for refining security posture.

- Additional response headers
  - Feature detection info
- 



## 7. Remediation Priority Plan (Recommended Order)

### Immediate (0–48 hours)

1. Secure and HttpOnly cookie flags
2. Anti-CSRF implementation
3. SQLi verification + WAF

4. XSS/Clickjacking header fixes

### Short-Term (1–2 weeks)

1. Add CSP
2. Add HSTS
3. Configure cache and content-type headers
4. Run WPScan + patch plugins/themes

### Ongoing

- Quarterly vulnerability scans
  - Annual penetration tests (PCI 11.3)
  - Monthly plugin & theme updates
- 



## 8. Compliance Conclusion

Based on the findings, the application:

**✗ Does NOT meet PCI DSS requirements for secure web applications.**

Once the high- and medium-severity vulnerabilities are resolved and a follow-up scan shows clean results, the site will be ready for a PCI DSS readiness review.