haiderm Security
haiderm.com

WordPress Security Assessment Report

INDEPENDENT SECURITY ASSESSMENT

PREPARED BY	haiderm Security
WEBSITE	https://haiderm.com
CLIENT WEBSITE	example-client-site.com
REPORT DATE	8 May 2026

Confidential

01 | DOCUMENT CONTROL

Field	Value
Document Title	WordPress Security Assessment Report
Version	1.0
Date	8 May 2026
Author	HaiderM Security
Status	Client-facing assessment deliverable
Classification	Confidential
Distribution	Prospective client evaluation
Credentials	CISSP, ISSAP, OSCP, MSc Cyber Security, 10+ years of cyber security experience

02 | EXECUTIVE SUMMARY

This sample assessment reviews the public-facing WordPress website example-client-site.com and demonstrates how a formal engagement would present WordPress-specific security risk, business impact, evidence, remediation steps, and retest guidance for a client-facing decision maker.

Coverage includes WordPress core exposure, plugin and theme exposure, authentication security, XML-RPC exposure, REST API exposure, user enumeration, security headers, TLS posture, file and directory exposure, malware indicators, blacklist indicators and safe unauthenticated web application checks.

The overall risk rating is Critical because two component-level issues could plausibly lead to site takeover if confirmed in a live engagement. The report contains 9 findings: 2 Critical, 3 High, 2 Medium, 1 Low, and 1 Informational. The five immediate priorities are the 2 Critical and 3 High findings: patch or remove vulnerable components, review privileged access, restrict exposed WordPress endpoints, strengthen authentication controls, and retest after remediation.

This sanitised sample avoids naming a real vulnerable component or demonstrating exploit payloads. In a live client report, Critical and High findings would include exact component names, installed versions, advisory or CVE references where available, timestamps, scoped request and response evidence, screenshots when useful, and retest proof after remediation.

OVERALL RISK	TOTAL FINDINGS	IMMEDIATE PRIORITIES	RETEST ADVICE
Critical	**9**	**5**	**Yes**
Immediate remediation is required for the two Critical issues.	2 Critical | 3 High | 2 Medium | 1 Low | 1 Info	2 Critical + 3 High findings require action in the first 7 days.	Targeted validation after remediation.

TOP RISK AREA	PRIMARY EXPOSURE	BEST FIRST ACTION	REMEDIATION PACE
Vulnerable components	**Plugin attack surface**	**Patch critical components**	**3 days**
Critical plugin-level exposure is the most urgent remediation area.	Public component exposure and risky workflows increase targeted attack likelihood.	Update or remove vulnerable plugins before lower-risk hardening work.	Critical and High actions should start immediately, with retesting after fixes.

03 | RISK RATING

Severity	Count	Meaning
Critical	2	Immediate, high-impact compromise risk with realistic paths to site takeover, code execution, administrative compromise or sensitive data exposure.
High	3	Significant risk with realistic exploitation paths and strong potential for account compromise, service abuse or major control failure.

Medium	2	Important weakness that increases exposure or contributes meaningfully to attack success, but is less immediately damaging on its own.
Low	1	Limited direct impact, often relating to information leakage or smaller hardening gaps.
Informational	1	Governance or defence-in-depth observation without direct exploitability in the current context.

04 | SCOPE

In scope	Out of scope
• Public WordPress frontend and login page • Public WordPress endpoints • Plugin and theme exposure checks • Version disclosure checks • Security headers and TLS • User enumeration • XML-RPC and REST API exposure • File and directory exposure • Malware and blacklist indicators • Non-destructive login attack surface review	• Denial of service • Social engineering or phishing • Live credential stuffing or password spraying against real accounts • Physical security • Destructive exploitation • Production data modification • Admin panel testing unless credentials are provided • Source code review unless separately agreed

05 | POSITIVE OBSERVATIONS

- HTTPS is enabled
- WordPress core appears up to date.
- No obvious malware indicators were observed.
- No public database error leakage was identified.
- Admin routes do not appear intentionally indexed.
- No sensitive payment data was identified during unauthenticated testing.
- Public content did not expose obvious customer records.
- Contact forms did not show obvious reflected handling issues during safe testing.

06 | FINDINGS SUMMARY

ID	Finding	Severity	Affected area	Priority
C-01	Vulnerable Plugin Allows Unauthenticated Remote Code Execution	Critical	Publicly exposed vulnerable plugin functionality	Immediate
C-02	Vulnerable Plugin Allows Privilege Escalation to Administrator	Critical	User registration or account-management plugin workflow	Immediate
H-01	XML-RPC Enabled and Exposed to Abuse	High	xmlrpc.php	Immediate
H-02	Missing Login Rate Limiting and Weak Brute Force Protection	High	wp-login.php	Immediate
H-03	SQL Injection Risk in Exposed Plugin or Theme Functionality	High	Public plugin or theme request handling	Immediate
M-01	REST API Allows User Enumeration	Medium	/wp-json/wp/v2/users	High
M-02	Missing or Weak Security Headers	Medium	HTTP response headers	High
L-01	WordPress Version Disclosure	Low	HTML source or asset query strings	Medium
I-01	No Web Application Firewall Detected	Informational	Edge protection	Planned

07 | RETEST STATUS

ID	Severity	Retest status	Closure evidence expected
C-01	Critical	Pending remediation	Patched component version, removed exposed code path and clean external validation.
C-02	Critical	Pending remediation	Updated workflow, reviewed admin users and confirmed no role escalation path.
H-01	High	Pending remediation	XML-RPC disabled or restricted with blocked unauthorised external access.
H-02	High	Pending remediation	Rate limiting, MFA and lockout/challenge behaviour confirmed.
H-03	High	Pending remediation	Patched database-facing component and safe validation of input handling.
M-01	Medium	Pending remediation	REST user exposure reduced or accepted with compensating login controls.
M-02	Medium	Pending remediation	Security-header baseline present without breaking site functionality.
L-01	Low	Pending remediation	Avoidable version disclosure reduced where practical.
I-01	Informational	Client decision	WAF/CDN control accepted, deferred or implemented with logging enabled.

08 | FINDINGS

C-01 | Vulnerable Plugin Allows Unauthenticated Remote Code Execution	CRITICAL
AFFECTED COMPONENT	Publicly exposed vulnerable plugin functionality
BUSINESS IMPACT	Possible full site compromise and malicious server-side code execution
PRIORITY	Immediate
SCORE	9.8
DESCRIPTION	In this sample scenario, a publicly exposed WordPress plugin is treated as a Critical issue because the affected functionality could permit unauthenticated remote code execution if the vulnerable version is confirmed. This is one of the highest-impact WordPress risk classes because successful exploitation may allow arbitrary server-side code execution, malware deployment, content tampering and persistent administrative access.
TECHNICAL EVIDENCE	`Sample evidence would include the exact plugin name, installed version, exposed file path, vendor advisory or CVE reference, timestamped request/response evidence and confirmation that the affected functionality is reachable without authentication. This sample report intentionally uses sanitised evidence and does not demonstrate exploit payloads.`

REMEDIATION	Treat this as an emergency change. Place the site into a controlled maintenance window, take a verified full backup, and record the current WordPress core, theme and plugin versions before making changes. Identify the affected plugin from wp-admin, wp-content/plugins, hosting file manager or WP-CLI. Compare the installed version with the vendor changelog and current security advisories. Update to the fixed release immediately, or remove the plugin completely if no fixed release is available. If removal is required, deactivate the plugin first, confirm the site still performs its required business functions, then delete the plugin directory rather than leaving inactive vulnerable code on disk. Replace it with a maintained alternative only after checking its update history and support status. Harden the exposed path after patching. Block direct access to unnecessary plugin endpoints, restrict administrative functions to authenticated users, and add WAF rules for suspicious requests targeting the affected component. Assume compromise may have been attempted. Review administrator users, recent file changes, unknown PHP files, scheduled tasks, access logs and security plugin alerts. Rotate WordPress administrator passwords, hosting credentials, SFTP/SSH credentials and database credentials if compromise is suspected. Verify the fix by confirming the vulnerable version is no longer present, the public metadata no longer exposes the vulnerable release, and the previously reachable risky functionality is no longer accessible without the intended controls.
RETEST GUIDANCE	Confirm the vulnerable component version is no longer present, validate that the exposed functionality has been hardened, and re-check public metadata leakage after remediation.
REFERENCES	WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/ WPScan Vulnerability Database - https://wpscan.com/vulnerability-database/ OWASP Web Security Testing Guide - https://owasp.org/www-project-web-security-testing-guide/

C-02 | Vulnerable Plugin Allows Privilege Escalation to Administrator	**CRITICAL**
AFFECTED COMPONENT	User registration or account-management plugin workflow
BUSINESS IMPACT	Unauthorised administrator-level access and full WordPress takeover
PRIORITY	Immediate
SCORE	9.4
DESCRIPTION	In this sample scenario, an exposed account-management or registration workflow is treated as Critical because a vulnerable implementation could allow a lower-privileged user to obtain administrator permissions. In WordPress, administrator access usually means full control over content, plugins, themes, user accounts and potentially server-side code execution through plugin or theme changes.
TECHNICAL EVIDENCE	`Sample evidence would include the affected plugin or workflow name, installed version, role-assignment configuration, relevant advisory references and safe validation showing whether untrusted users can influence role or capability assignment. No live privilege escalation against a real account is performed in this sample.`

REMEDIATION	Apply the vendor security update immediately or remove the affected account-management plugin if a fixed version is unavailable. Do this before accepting new user registrations or role changes through the affected workflow. Temporarily disable public registration, membership sign-up, profile editing and role-management features until the patch is confirmed. Where registration is business-critical, require manual approval and limit new accounts to the lowest possible role. Audit all WordPress users after patching. Look for recently created administrators, unexpected role changes, unfamiliar email addresses, dormant accounts that became active, and accounts created around suspicious log entries. Remove unknown accounts and demote any user that does not require elevated access. Enforce MFA for administrators and editors, require strong password resets for privileged users, and limit admin dashboard access by IP or identity-aware access where possible. Review plugin settings for role mapping, default role assignment, invitation links, social login, REST/API actions and AJAX handlers. Confirm users cannot influence their assigned role through request parameters, profile fields or registration metadata. Retest by creating a low-privileged test account and confirming it cannot assign, request, inherit or escalate to administrator permissions through registration, profile updates, API calls or plugin-specific workflows.
RETEST GUIDANCE	Verify that privilege boundaries are correctly enforced, administrator roles cannot be assigned through public workflows, and all related components are fully patched.
REFERENCES	WordPress Roles and Capabilities - https://wordpress.org/documentation/article/roles-and-capabilities/ OWASP ASVS - https://owasp.org/www-project-application-security-verification-standard/ WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/

H-01 | XML-RPC Enabled and Exposed to Abuse	**HIGH**
AFFECTED COMPONENT	xmlrpc.php
BUSINESS IMPACT	Automated password attack and service abuse surface
PRIORITY	Immediate
SCORE	8.1
DESCRIPTION	The target exposes WordPress XML-RPC functionality to unauthenticated users. XML-RPC exposure is not automatically a vulnerability where a legitimate integration requires it; the risk becomes High when the endpoint is openly reachable and can support bulk login attempts, system.multicall abuse or automated attack workflows, especially alongside weak login-rate limiting or limited account protection.
TECHNICAL EVIDENCE	`POST /xmlrpc.php HTTP/1.1 Host: example-client-site.com Content-Type: text/xml HTTP/1.1 200 OK XML-RPC server accepts POST requests only. In a live engagement, severity would be confirmed by checking whether XML-RPC is required, whether system.multicall is available, whether login attempts are rate-limited, and whether administrator accounts are protected with MFA.`

REMEDIATION	Disable XML-RPC completely if the site does not rely on Jetpack, remote publishing, mobile app posting or another confirmed integration. This can be done through a security plugin, web server rule, WAF rule or hosting control panel. If XML-RPC must remain enabled, restrict it rather than leaving it open to the internet. Allow only trusted integration IP ranges where possible, block system.multicall abuse, and rate-limit requests to /xmlrpc.php separately from normal page traffic. Add authentication protections around the risk it creates. Enforce MFA for administrators, use strong unique passwords, remove unused accounts, and disable password reuse across hosting, email and WordPress accounts. Monitor for repeated POST requests to /xmlrpc.php, high request volumes from single IPs, multicall patterns and failed login bursts. Feed these events into the WAF or security plugin so abusive sources are blocked automatically. Verify the fix by requesting /xmlrpc.php from an external network. The preferred result is a blocked response such as 403/404, or a tightly limited response if a business integration still needs access.
RETEST GUIDANCE	Confirm the endpoint is disabled or appropriately restricted and that unauthorised requests no longer receive functional responses.
REFERENCES	WordPress XML-RPC API - https://developer.wordpress.org/apis/xml-rpc/ WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/ OWASP WSTG - https://owasp.org/www-project-web-security-testing-guide/

H-02 | Missing Login Rate Limiting and Weak Brute Force Protection	**HIGH**
AFFECTED COMPONENT	wp-login.php
BUSINESS IMPACT	Higher likelihood of admin credential attack success
PRIORITY	Immediate
SCORE	7.6
DESCRIPTION	The default WordPress login interface appears directly reachable without visible evidence of strong rate limiting, bot management or equivalent anti-automation control.

TECHNICAL EVIDENCE	`$ curl -I https://example-client-site.com/wp-login.php`
	`HTTP/1.1 200 OK`
	`Cache-Control: no-store`
	
	`No visible anti-automation challenge, lockout signal or rate-limit response was observed during safe unauthenticated review. A live engagement would validate this carefully with an agreed low-volume test plan and without credential stuffing.`

REMEDIATION	Install or configure a login protection control that rate-limits failed authentication attempts on wp-login.php and any custom login endpoint. Set lockout thresholds that slow automated guessing without blocking normal users. Enable MFA for all administrator and editor accounts. Prioritise authenticator-app or hardware-key MFA over email-only codes where possible, and require backup codes to be stored securely. Reduce the number of high-value targets. Remove unused administrator accounts, demote users who do not need admin rights, disable shared accounts, and require each privileged user to have a named individual account. Improve password policy and recovery controls. Require long unique passwords, reset weak or reused passwords, protect password-reset forms from enumeration and ensure admin email accounts are also protected with MFA. Add layered blocking. Use a WAF, CDN security rule or hosting firewall to challenge repeated login attempts, block obvious automation, and alert on login attempts from unexpected countries or hosting-provider IP ranges. Verify by performing controlled failed-login testing: repeated incorrect attempts should trigger throttling, lockout or challenge behaviour, and security logs should clearly show the events.
RETEST GUIDANCE	Confirm anti-automation controls trigger appropriately and that administrator accounts are MFA-protected.

REFERENCES	OWASP Top 10 A07 Identification and Authentication Failures - https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/ OWASP ASVS - https://owasp.org/www-project-application-security-verification-standard/ WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/

H-03 | SQL Injection Risk in Exposed Plugin or Theme Functionality	HIGH
AFFECTED COMPONENT	Public plugin or theme request handling
BUSINESS IMPACT	Potential database compromise, content manipulation and sensitive data exposure
PRIORITY	Immediate
SCORE	8.6
DESCRIPTION	The assessment identified exposed plugin or theme functionality that should be treated as a high-priority SQL injection risk until the affected code path is patched or safely ruled out. In WordPress environments, unsafe database-facing request handling can allow attackers to extract data, alter content or support administrator compromise. This sample rates the issue High because the evidence is framed as safe, non-destructive validation rather than confirmed live exploitation.
TECHNICAL EVIDENCE	`Public parameter handling patterns and component exposure indicate that database-facing functionality requires immediate secure code and vendor advisory validation. Safe validation should focus on parameterised query handling, unexpected database errors, time-delay behaviour in a controlled environment and affected component version checks. No destructive exploit traffic was performed in this sample report.`
REMEDIATION	Patch or remove the affected plugin, theme or custom code path immediately. If the vulnerable functionality is custom-built, route the fix through code review and deploy only after testing on a staging copy of the site. Replace unsafe database handling with prepared statements and strict input validation. In WordPress code, use $wpdb->prepare() for SQL containing user-controlled values, validate expected data types, and reject unexpected operators, sort fields, filters or identifiers. Apply allow-lists for parameters such as IDs, order fields, category names and query filters. Avoid passing request values directly into SQL fragments, table names, column names, ORDER BY clauses or LIKE patterns without validation and escaping. Reduce exposure while the permanent fix is prepared. Disable the affected feature, restrict it to authenticated users, add WAF rules for SQL injection patterns, and monitor requests containing unusual quotes, comments, UNION, sleep/time-delay patterns or encoded payloads. Check whether data may have been accessed or altered. Review database users, recent administrator activity, suspicious content changes, unknown posts/pages, injected options and unexpected WordPress admin accounts. Verify the fix with safe retesting: confirm malicious-looking input is rejected or parameterised, database errors are not exposed, and the application returns controlled responses without changing query structure.
RETEST GUIDANCE	Re-check the affected workflow after patching, confirm that the vulnerable component is updated or removed, and validate that user-controlled input cannot change SQL query structure.
REFERENCES	WordPress wpdb::prepare() - https://developer.wordpress.org/reference/classes/wpdb/prepare/ OWASP SQL Injection - https://owasp.org/www-community/attacks/SQL_Injection PortSwigger SQL Injection Academy - https://portswigger.net/web-security/sql-injection

M-01 | REST API Allows User Enumeration	MEDIUM
AFFECTED COMPONENT	/wp-json/wp/v2/users
BUSINESS IMPACT	Improves attacker targeting by revealing likely account identifiers
PRIORITY	High
SCORE	5.3

DESCRIPTION	The public REST API appears to expose user-related data that may reveal author slugs or account identifiers. This is common in WordPress and is not always a standalone vulnerability; the risk becomes more meaningful when exposed identifiers map to login names or support targeted password attacks against administrator or editor accounts.

TECHNICAL EVIDENCE	`GET /wp-json/wp/v2/users`
	`HTTP/1.1 200 OK`
	`[{"id":1,"slug":"siteeditor"}]`

REMEDIATION	Review whether public user data is needed through the REST API. If not required, restrict unauthenticated access to user endpoints such as /wp-json/wp/v2/users using a security plugin, custom permission callback or web server/WAF rule. Reduce author identifier leakage. Avoid exposing predictable usernames in author archives, REST responses, page source, feeds and public profile links. Use display names that differ from login names. If public authors must remain visible, make enumeration less useful by enforcing MFA, strong passwords and login rate limiting. Usernames should not be treated as secrets, but they should not make credential attacks easier. Check plugins and themes that add custom REST routes. Ensure each route has an explicit permission_callback and does not return unnecessary user email addresses, roles, internal IDs or account metadata. Verify by requesting the user REST endpoints while logged out. The response should either be blocked, return only intentionally public author information, or omit fields that would assist account targeting.
RETEST GUIDANCE	Re-check public API responses for user objects and author enumeration paths.
REFERENCES	WordPress REST API Handbook - https://developer.wordpress.org/rest-api/ WordPress REST API Users Reference - https://developer.wordpress.org/rest-api/reference/users/ WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/

M-02 | Missing or Weak Security Headers	**MEDIUM**
AFFECTED COMPONENT	HTTP response headers
BUSINESS IMPACT	Higher exposure to clickjacking and content abuse
PRIORITY	High
SCORE	6.1
DESCRIPTION	The response set lacks a stronger baseline of modern security headers such as Content-Security-Policy, frame protections, Referrer-Policy and Permissions-Policy.

TECHNICAL EVIDENCE	`$ curl -I https://example-client-site.com/`
	`HTTP/1.1 200 OK`
	`Content-Security-Policy: not present`
	`X-Frame-Options: not present`
	`Referrer-Policy: not present`

REMEDIATION	Define a security-header baseline for the site and apply it at the web server, CDN or security plugin layer so it covers WordPress pages, static assets and error responses consistently. Add or tune Content-Security-Policy in report-only mode first. Inventory required script, style, image, font, frame and connection sources, then move to enforcement once legitimate site functionality is confirmed. Set clickjacking protection with frame-ancestors in CSP, or X-Frame-Options if legacy support is needed. Only allow trusted framing partners when the business has a clear requirement. Add Referrer-Policy, Permissions-Policy, X-Content-Type-Options and HSTS where appropriate. For HSTS, confirm HTTPS works correctly across the main domain and subdomains before enabling long max-age or includeSubDomains. Test the headers on the homepage, login page, admin redirects and representative content pages. Confirm the changes do not break checkout, forms, embedded media, analytics or third-party integrations. Maintain the header configuration as part of deployment documentation so future plugin, CDN or hosting changes do not silently remove the controls.
RETEST GUIDANCE	Validate the new header baseline and ensure the policy set does not break required front-end functionality.
REFERENCES	OWASP Secure Headers Project - https://owasp.org/www-project-secure-headers/ Mozilla HTTP Observatory - https://developer.mozilla.org/en-US/observatory MDN Content-Security-Policy - https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

L-01 | WordPress Version Disclosure	**LOW**
AFFECTED COMPONENT	HTML source or asset query strings
BUSINESS IMPACT	Supports environment profiling
PRIORITY	Medium
SCORE	3.1
DESCRIPTION	Version information appears inferable from source output or asset version parameters. This can help attackers narrow exploit research more quickly.
TECHNICAL EVIDENCE	`Observed HTML and static asset references included version-style query parameters.`
REMEDIATION	Keep WordPress core, plugins and themes fully patched; this is more important than hiding the version. Establish a regular update cadence with backups, staging checks and rollback steps. Reduce avoidable version leakage from generator tags, readme files, asset query strings and exposed metadata where doing so does not break cache-busting or site functionality. Review theme and plugin output for hard-coded version comments or public changelog references. Remove unnecessary disclosure from custom theme templates and build artefacts. Pair this with inventory management. Maintain a list of installed component versions internally so defenders can patch quickly without needing public disclosure to identify exposure. Verify by checking page source, public metadata files and common scanner-visible locations. Remaining version information should either be intentionally public or operationally necessary.
RETEST GUIDANCE	Review HTML and asset paths again after hardening changes.
REFERENCES	WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/ OWASP WSTG Information Gathering - https://owasp.org/www-project-web-security-testing-guide/

I-01 | No Web Application Firewall Detected	INFO
AFFECTED COMPONENT	Edge protection
BUSINESS IMPACT	Reduced resilience against automated WordPress attack traffic and commodity exploitation attempts
PRIORITY	Planned
SCORE	0.0
DESCRIPTION	No clear public signal of a dedicated web application firewall, security-enabled CDN or equivalent edge protection was observed. This is not a confirmed vulnerability and should not be treated as mandatory by itself; it is included as a defence-in-depth improvement for WordPress sites exposed to automated scanning, login attacks and plugin-targeted exploitation.
TECHNICAL EVIDENCE	`Public response headers and edge behaviour did not clearly indicate a WAF challenge, managed security CDN or application-layer filtering control. Internal confirmation with the client or hosting provider would be required before treating this as definitive.`
REMEDIATION	Select an edge protection option appropriate for the site, such as a managed WordPress security firewall, hosting WAF or CDN security service. Enable managed WordPress rules, bot protections, login endpoint controls and virtual patching for known plugin vulnerabilities. Start in monitoring or low-friction mode if the site has checkout, forms or membership features, then move to stricter blocking once false positives are reviewed. Configure alerting for blocked exploit attempts, repeated login failures and suspicious requests to wp-admin, xmlrpc.php and plugin paths. Review the WAF policy after major plugin, theme or hosting changes so legitimate business workflows continue to work.
RETEST GUIDANCE	Confirm that the selected control is active on the production domain, that common WordPress attack patterns are logged or blocked, and that normal site functions such as forms, login and checkout remain usable.
REFERENCES	OWASP ModSecurity Core Rule Set - https://owasp.org/www-project-modsecurity-core-rule-set/ WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening/ OWASP WSTG - https://owasp.org/www-project-web-security-testing-guide/

09 | COVERAGE CHECKLIST

Control area	Status	Observation
WordPress core version	Observed	Core appears current in the sample, but version signals should still be tracked so patch status can be verified after every WordPress release.
Plugin exposure	Priority	Critical component exposure is the leading remediation area; plugin names, versions and public metadata should be mapped to current advisories before closure.
Theme exposure	Improvement	Theme paths and asset versions can support fingerprinting; unused themes should be removed and the active theme should be kept under update control.
Admin login security	Priority	Login protection should include rate limiting, MFA, strong password policy and alerting for repeated failed attempts against privileged accounts.
MFA recommendation	Recommended	MFA should be mandatory for administrators and editors, with recovery codes stored securely and shared accounts removed where possible.
XML-RPC	Priority	Public XML-RPC access should be disabled unless a confirmed integration requires it; if retained, restrict source IPs and monitor POST volume.

REST API	Improvement	Unauthenticated user exposure should be reduced where feasible, especially where author slugs map to login names or support targeted password attacks.
Security headers	Improvement	A baseline should include CSP planning, frame protection, Referrer-Policy, Permissions-Policy, X-Content-Type-Options and HSTS where suitable.
TLS	Good	HTTPS is enabled; confirm certificate chain quality, protocol support, redirect behaviour and HSTS readiness during retesting.
Directory listing	Checked	No directory-listing finding is retained, but uploads, cache and temporary paths should still be spot-checked after deployments.
Backups	Checked	No exposed-backup finding is retained, but backup and export files should be stored outside the web root and covered by routine exposure checks.
Debug files	No issue	No public debug leakage was confirmed; keep WP_DEBUG disabled in production and ensure logs are not written to public paths.
Malware indicators	No issue	No obvious public compromise indicators were observed; this does not replace authenticated file-integrity review or server-side malware scanning.
Blacklist indicators	No issue	No immediate blacklist warning behaviour was seen; monitor search-console, safe-browsing and email reputation signals after remediation.
WAF or CDN	Opportunity	No clear WAF presence was detected; a managed WAF or security-enabled CDN would add virtual patching, bot filtering and better attack visibility.
Logging and monitoring	Review needed	Client-side confirmation is needed for access logs, security alerts, administrator activity logs and a process for reviewing high-risk events.
Contact forms	No issue	No obvious reflected handling issue was observed during safe testing; forms should still be protected with spam controls, validation and alerting.

10 | REMEDIATION ROADMAP

EMERGENCY CONTAINMENT
0 TO 12 HOURS

- Patch or remove critical vulnerable components.
- Review administrator accounts and recent role changes.
- Restrict XML-RPC if it is not required.
- Enable login rate limiting and MFA for privileged users.
- Preserve logs and evidence before major cleanup.

CONTROL HARDENING
12 TO 48 HOURS

- Validate and patch SQL injection exposure
- Reduce REST API user enumeration where feasible
- Deploy a security-header baseline
- Decide on WAF/CDN protection and enable logging
- Reduce avoidable version disclosure

VERIFICATION
48 TO 72 HOURS

- Retest all Critical and High findings
- Document accepted residual risks
- Create plugin ownership and update policy
- Improve monitoring and alert review
- Schedule periodic WordPress security reassessment

11 | WHAT HAIDERM SECURITY WOULD DO NEXT

After delivering this assessment, the next stage would focus on safe validation, practical remediation support, and closure evidence. The goal is not just to identify issues, but to help the client reduce risk quickly and prove the fixes worked.

Step	Outcome
Validate affected components	Confirm exact plugin or theme names, versions, advisories and real exposure within the agreed scope.
Support remediation	Help prioritise patching, removal, configuration changes and compensating controls without disrupting normal site operations.
Retest fixes	Re-check remediated findings and provide clear fixed, partially fixed or still exposed status.

Step	Outcome
Close the loop	Provide a concise closure note that the client can keep for governance, insurer, stakeholder or supplier assurance needs.

12 | TOOLS AND TECHNIQUES

Assessment area	Tools/scripts
Manual request validation	Burp Suite, OWASP ZAP, browser DevTools
WordPress component exposure	WPScan, Wappalyzer, custom WordPress metadata script
Known-vulnerability checks	Nuclei with curated safe templates, WPScan vulnerability data
HTTP header and policy review	Mozilla HTTP Observatory, ProjectDiscovery httpx, curl header script
TLS and certificate review	SSL Labs Server Test, testssl.sh
Passive reconnaissance	OWASP Amass, certificate transparency review, httpx reachability checks
Retest and closure evidence	Burp Repeater, curl replay script, screenshot comparison, response-diff script
Authenticated configuration review	WP-CLI, WordPress Site Health, plugin inventory export script

13 | OFFICIAL HARDENING GUIDANCE REFERENCES

- WordPress Hardening Guide - https://developer.wordpress.org/advanced-administration/security/hardening
- OWASP Web Security Testing Guide (stable) - https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing
- OWASP Secure Headers Project - https://owasp.org/www-project-secure-headers/
- NCSC Application Development Guidance - https://www.ncsc.gov.uk/guidance/application-development-guidance-introduction

14 | ABOUT HAIDER QURESHI

Haider Qureshi is an application security professional based in London, with more than 10 years of cyber security experience across banking, telecoms, technology, and consulting. His work focuses on secure SDLC, threat modelling, secure design reviews, penetration testing, cloud security, and practical remediation support.

Haider holds CISSP, ISSAP, and OSCP certifications, and an MSc in Cyber Security from Royal Holloway, University of London. His approach combines attacker-aware technical testing with clear business-risk reporting, so clients understand what matters, why it matters, and how to fix it safely.

15 | WORK WITH HAIDER

If you need a WordPress security assessment, remediation support, or retesting after fixes, you can contact Haider directly using the links below. Engagements can be scoped around public WordPress exposure, plugin and theme risk, authentication security, REST API/XML-RPC exposure, security headers, malware indicators, and clear remediation reporting.

Contact: https://haiderm.com/contact/

WordPress Security Assessment Service: https://haiderm.com/wordpress-security-assessment-service/