SECURITY ASSESSMENT REPORT

WordPress Intranet — Static Code Analysis

Prepared for: [Client] · Confidential

Analyst: Daniel Oa · danieloa.com

Date: February 2026 · Classification: CONFIDENTIAL

1. Executive Summary

A deep static code analysis of a WordPress intranet application revealed 20 vulnerabilities across the codebase, server configuration, and third-party integrations. Five findings were rated Critical, including hardcoded API credentials for external HR and payment systems, a misconfigured JWT authentication plugin, exposed database credentials, and a hosting encryption key stored in a web-accessible location. Despite these issues, the application demonstrates several strong security controls including disabled file editing, blocked XML-RPC, enforced HTTPS, and enterprise Azure AD integration.

CRITICAL	HIGH	MEDIUM	LOW
5	6	6	3

2. Scope & Methodology

Target	PHP/WordPress intranet application (static backup analysis)
Testing Type	Static code analysis — full source code review
Methods	Manual code review · Configuration analysis · Plugin audit · Log inspection
Framework	OWASP Testing Guide v4.2 · OWASP Top 10
Period	February 2026
Analyst	Daniel Oa — danieloa.com

3. Findings Summary

ID	Title	Severity	CVSS
F-01	Hardcoded external API credentials in source code	Critical	9.8
F-02	Hardcoded Basic Auth token for third-party payment service	Critical	9.5
F-03	JWT plugin active — JWT_AUTH_SECRET_KEY undefined	Critical	9.3
F-04	Live database credentials exposed in backup file	Critical	9.1
F-05	Hosting encryption key stored in web-accessible path	Critical	8.9
F-06	Stored XSS — unescaped third-party API output	High	8.5
F-07	Path traversal — bypassable sanitization in file download handler	High	8.2
F-08	Information disclosure before redirect — private file URLs exposed	High	7.8
F-09	CSRF logout bypass + open redirect in theme functions	High	7.5
F-10	SSL certificate verification disabled on all external API calls	High	7.3
F-11	Unauthenticated access to draft/private content via REST API	High	7.0
F-12	Dangling .htaccess exception for non-existent directory	Medium	6.5
F-13	Azure tenant ID hardcoded in plugin source	Medium	6.2
F-14	PHP error log accessible inside public_html	Medium	5.9
F-15	Unlimited cURL timeout — potential resource exhaustion	Medium	5.5
F-16	53+ metadata files (.DS_Store) deployed to production	Medium	5.2

F-17	Hardcoded PHP session cookie in external API headers	Medium	4.8
F-18	Three conflicting security plugins running simultaneously	Low	3.9
F-19	Access control partially reliant on client-side JavaScript	Low	3.5
F-20	WordPress readme.html exposes version number	Low	3.2

4. Detailed Findings

F-01 — Hardcoded External API Credentials — CRITICAL · CVSS 9.8

Description	Plaintext credentials for an external HR system API are hardcoded directly in two PHP source files. The credentials include email and password used to authenticate against a third-party HR platform.
Impact	Full unauthorized access to the HR system. An attacker with source code access can extract all employee records including names, contacts, positions, and sensitive HR data.
Remediation	1. Remove credentials from source code immediately. 2. Store in environment variables or wp-config.php constants. 3. Rotate the compromised credentials on the third-party system. 4. Audit access logs on the HR platform for unauthorized access.

F-02 — Hardcoded Basic Auth Token — Payment Service — CRITICAL · CVSS 9.5

Description	An HTTP Basic Authentication token is hardcoded as a base64 string in a PHP file. The token provides authenticated access to a third-party payment/billing platform.
Impact	Full compromise of the payment service account. Credentials can be extracted by any person with access to source code, version history, or server files.
Remediation	1. Revoke the token on the payment platform immediately. 2. Generate a new credential and store it as a server environment variable. 3. Never commit credentials to version control — add a pre-commit hook.

F-03 — JWT Plugin Active — Secret Key Undefined — CRITICAL · CVSS 9.3

Description	The JWT authentication plugin is installed and active, but the required JWT_AUTH_SECRET_KEY constant is never defined in wp-config.php.
Impact	All JWT-protected REST API endpoints either fail with errors or fall back to no authentication, potentially exposing protected data to unauthenticated requests.
Remediation	1. Add to wp-config.php: define('JWT_AUTH_SECRET_KEY', ''); 2. Generate a secure value with: openssl rand -base64 48 3. Test all REST API endpoints after the fix.

F-04 — Database Credentials Exposed in Backup — CRITICAL · CVSS 9.1

Description	The database name, username, and password are stored in plaintext in wp-config.php within a backup that was accessible. Old commented-out authentication keys were also present.
Impact	An attacker who accesses this backup file obtains full database access. The old commented keys increase the attack surface for session forgery.
Remediation	1. Delete lines containing old commented-out authentication keys. 2. Rotate the database password immediately. 3. Move wp-config.php one level above the web root. 4. Restrict file permissions: chmod 600 wp-config.php

F-05 — Hosting Encryption Key in Web-Accessible Path — CRITICAL · CVSS 8.9

Description	The hosting provider's security module encryption key is stored in a PHP file inside the public web directory, making it potentially accessible via direct URL request.
Impact	Compromise of all data protected by this encryption key, potentially including security logs, encrypted backups, and security module data.
Remediation	1. Contact the hosting provider to rotate this key. 2. Verify the file is not directly accessible via HTTP. 3. Add .htaccess protection: Require all denied

F-06 — Stored XSS — Unescaped Third-Party API Output — HIGH · CVSS 8.5

Description	All employee fields fetched from the external HR API are rendered directly into HTML without any escaping. Fields include names, email addresses, phone numbers, and role descriptions.
Impact	Stored XSS via a compromised third-party API. If the HR system is breached or returns malicious data, all users viewing the employee directory execute attacker-controlled JavaScript.
Remediation	1. Wrap every output with htmlspecialchars(): echo htmlspecialchars($value, ENT_QUOTES, 'UTF-8'); 2. Or use WordPress's esc_html() function for all echoed values. 3. Apply a Content Security Policy header to limit script execution.

F-07 — Path Traversal — File Download Handler — HIGH · CVSS 8.2

Description	The file download handler uses str_replace('..', '', $file) as its only sanitization. This is trivially bypassed with payloads like ....// which collapse after the replacement.
Impact	An authenticated user can read arbitrary files from the server filesystem, including wp-config.php, /etc/passwd, SSL private keys, and any file readable by the web process.
Remediation	1. Replace sanitization with proper path validation: $file = realpath($basedir . '/' . ltrim($_GET['file'], '/')); if (strpos($file, $basedir) !== 0) die('Access denied'); 2. Use WordPress's download_url() for file delivery.

F-08 — Information Disclosure Before Redirect — HIGH · CVSS 7.8

Description	When an unauthenticated user requests a private file, the script prints the full URL of the file upload directory before issuing the redirect, disclosing internal server paths.
Impact	Private file paths and directory structure are exposed to unauthenticated requests, enabling direct hotlinking and download of files intended to be protected.
Remediation	1. Remove the echo statement on the affected line. 2. Call wp_redirect('/') and exit immediately without printing any URL. 3. Never output server-side paths to unauthenticated users.

F-09 — CSRF Logout Bypass + Open Redirect — HIGH · CVSS 7.5

Description	A theme function hooks into WordPress's nonce verification and explicitly skips the nonce check for logout when a specific GET parameter is present, then redirects to a user-supplied URL.
Impact	(1) CSRF Logout: any page the user visits can silently terminate their session. (2) Open Redirect: users can be sent to arbitrary external URLs after logout.
Remediation	1. Remove this function entirely — use WordPress's standard nonce-protected logout flow. 2. If custom redirect is required, validate against an allowlist of internal URLs. 3. Never bypass WordPress nonce verification.

F-10 — SSL Verification Disabled on External API Calls

HIGH · CVSS 7.3

Description	CURLOPT_SSL_VERIFYPEER is set to false on all outbound cURL requests to the HR API. This completely disables TLS certificate chain validation and hostname verification.
Impact	On any compromised network path, an attacker can intercept all API traffic via MITM, stealing the Bearer token, capturing full HR data responses, and injecting malicious responses.
Remediation	1. Remove CURLOPT_SSL_VERIFYPEER => false from all cURL calls. 2. If the server CA bundle is outdated: update-ca-certificates 3. Set CURLOPT_CAINFO to a trusted CA bundle path if needed.

F-11 — Unauthenticated REST API Access to Private Content

HIGH · CVSS 7.0

Description	A disabled custom plugin contains a REST endpoint with 'permission_callback' => '__return_true' (no auth required) that queries all post statuses including draft, private, and trash.
Impact	If re-enabled, all draft posts, private pages, and trashed content become publicly readable via a simple HTTP GET request with no authentication.
Remediation	1. Delete the plugin file entirely. 2. If the functionality is needed, add proper authentication: 'permission_callback' => 'is_user_logged_in' 3. Restrict post_status to 'publish' only in the query.

F-12 — Dangling .htaccess Exception

MEDIUM · CVSS 6.5

Description	The .htaccess grants access to a specific PHP file path via a FilesMatch rule, but the referenced directory does not exist on the server.
Impact	Future developers may create a file at that path expecting it to be protected, when the .htaccess would actually expose it. Creates maintenance confusion and potential security gaps.
Remediation	1. Remove the dangling exception from .htaccess. 2. Document all .htaccess exceptions with inline comments explaining their purpose. 3. Conduct a quarterly .htaccess review.

F-13 — Azure Tenant ID Hardcoded in Plugin

MEDIUM · CVSS 6.2

Description	The Microsoft Azure tenant ID is hardcoded as a private class property in a plugin file. While not a secret by itself, it is sensitive configuration that belongs in the admin settings.
Impact	Enables attackers to craft targeted Microsoft 365 phishing pages for this specific tenant, or enumerate tenant users and applications via Microsoft Graph API.
Remediation	1. Move the tenant ID to the WordPress options system (stored via admin panel). 2. Treat all Azure/cloud configuration identifiers as sensitive configuration, not source code constants.

F-14 — PHP Error Log in Public Directory

MEDIUM · CVSS 5.9

Description	The PHP error log file is stored inside public_html and contains full server filesystem paths, plugin version strings, and admin usernames from PHP notices and warnings.
Impact	An attacker can map the entire server directory structure, identify vulnerable plugin versions, discover admin usernames, and understand internal application logic from error messages.
Remediation	1. Add to .htaccess: Require all denied 2. Configure error_log in php.ini to write outside public_html. 3. Set log_errors = On and display_errors = Off in production.

F-15 — Unlimited cURL Timeout

MEDIUM · CVSS 5.5

Description	CURLOPT_TIMEOUT is set to 0 (unlimited) for the employee list API call. If the external API becomes unresponsive, each PHP worker hangs indefinitely waiting for a response.
Impact	Under moderate traffic, all PHP workers exhaust waiting for an unresponsive API, causing complete denial of service for the entire intranet application.
Remediation	1. Set a reasonable timeout: CURLOPT_TIMEOUT => 15, CURLOPT_CONNECTTIMEOUT => 5 2. Implement response caching using WordPress transients to reduce API call frequency. 3. Add error handling for cURL timeout failures.

F-16 — 53+ Metadata Files Deployed to Production

MEDIUM · CVSS 5.2

Description	Over 53 macOS .DS_Store Finder metadata files were found across the wp-content directory tree. These files contain directory listings and file names from the developer's local machine.
Impact	An attacker can enumerate all filenames in affected directories, revealing plugin structure, theme files, custom code paths, and upload organization.
Remediation	1. Add to .htaccess: Require all denied 2. Add .DS_Store to .gitignore globally: git config --global core.excludesfile ~/.gitignore_global 3. Delete all existing .DS_Store files from the server.

F-17 — Hardcoded Session Cookie in API Headers

MEDIUM · CVSS 4.8

Description	A hardcoded PHPSESSID value and language cookies are sent with every API request to the external HR system. This session ID may grant persistent access independently of the API token.
Impact	The hardcoded session could grant persistent unauthorized access to the HR platform. Session IDs in source code are exposed to all developers and anyone with repository access.
Remediation	1. Remove the hardcoded Cookie header entirely. 2. The Bearer token authentication should be sufficient for the API. 3. If cookies are required by the API, obtain them dynamically per request.

F-18 — Conflicting Security Plugins

LOW · CVSS 3.9

Description	Three independent security plugins (Wordfence, Sucuri, and a hosting provider plugin) are active simultaneously, each applying its own WAF rules, file monitoring, and firewall configuration.
Impact	Potential rule conflicts causing false positives, double event logging, increased server resource usage, and a larger attack surface from three separate security code bases.
Remediation	1. Choose one primary security plugin. Recommended: the hosting provider's native plugin for best integration, or Wordfence for the most features. 2. Deactivate and delete the others.

F-19 — Client-Side Access Control

LOW · CVSS 3.5

Description	The visibility of a restricted section button is controlled via JavaScript that checks a server-provided variable. Any user can override this check using browser developer tools.
Impact	Non-authorized users can discover the restricted section's existence and bypass the client-side visibility check. Server-side protection is in place, but the section is revealed in page source.
Remediation	1. The server-side protection is correct and should remain. 2. Avoid revealing the restricted section name or access mechanism in client-side JavaScript. 3. Use a generic identifier instead of the actual section name in JS variables.

F-20 — WordPress Version Disclosure via readme.html

LOW · CVSS 3.2

Description	The default WordPress readme.html file is present at the web root and publicly accessible. It discloses the exact WordPress version number to unauthenticated visitors.
Impact	Version disclosure enables automated scanning tools to identify and target version-specific exploits. Combined with other vulnerabilities, it accelerates automated attacks.
Remediation	1. Delete readme.html from the server. 2. Add to functions.php: add_filter('the_generator', '__return_empty_string'); 3. Remove the version from RSS feeds and HTML meta tags.

5. Positive Security Controls

The following security controls were found correctly implemented and should be maintained:

✓ DISALLOW_FILE_EDIT enabled

Theme and plugin code cannot be edited from the WordPress dashboard, preventing code injection via admin panel compromise.

✓ XML-RPC blocked via .htaccess

XML-RPC is completely disabled, eliminating brute-force amplification attacks and historical RCE vectors.

✓ HTTPS enforced

All HTTP traffic is redirected to HTTPS with a 301 redirect, preventing cleartext credential interception.

✓ WP_DEBUG disabled in production

PHP errors are not displayed to end users, preventing information leakage via error messages.

✓ Backup directory protected

The backup archives directory has Require all denied, preventing direct web extraction of database backups.

✓ Generic login error messages

A custom login_errors filter returns generic messages, preventing username enumeration via login failures.

✓ Azure AD integration

The restricted section validates users against Microsoft Graph API, providing enterprise-grade identity verification.

✓ Admin bar hidden for non-admins

Non-admin users do not see the WordPress admin toolbar, reducing information exposure.

6. Remediation Roadmap

Phase	Findings	Actions
Immediate (< 24h)	F-01, F-02, F-03, F-04, F-05	Revoke all hardcoded credentials · Rotate database password · Define JWT secret key · Contact host to rotate encryption key
Short-term (< 1 week)	F-06 – F-11	Fix XSS output escaping · Patch path traversal · Fix logout CSRF · Enable SSL verification · Restrict REST API endpoint
Medium-term (< 1 month)	F-12 – F-17	Clean .htaccess · Move tenant ID to settings · Protect error log · Set cURL timeouts · Remove .DS_Store files · Remove hardcoded cookies
Ongoing	F-18 – F-20	Consolidate security plugins · Remove readme.html · Add JS access control note · Subscribe to WPScan CVE alerts · Conduct quarterly audits

7. Disclaimer

This assessment was conducted via static code analysis of a project backup. It does not constitute a full penetration test — dynamic exploitation, authenticated runtime testing, and network-layer assessment are outside the scope of this engagement. All findings were responsibly disclosed to the client. This report is confidential and intended solely for the authorized recipient.