INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-15 19:02 UTC

Avada Builder WordPress Plugin Flaws Allow Credential Theft and Database Extraction Across One Million Sites

CVE VULNERABILITY | HIGH | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0185
Type	CVE Vulnerability
CVE ID	CVE-2026-4782, CVE-2026-4798
Severity	HIGH
CVSS Base Score	7.5
EPSS Score	0.0004 (12th percentile)
Affected Products	Avada Builder WordPress plugin <= 3.15.2 (CVE-2026-4782); Avada Builder WordPress plugin <= 3.15.1 with WooCommerce previously installed then deactivated (CVE-2026-4798); patched in version 3.15.3
Published	2026-05-15T11:56:56
Discovery Source	Rss

Executive Summary

Two vulnerabilities in the Avada Builder WordPress plugin expose approximately one million websites to credential theft and full database compromise. An authenticated attacker can read the site's configuration file, obtaining database passwords and encryption keys; a separate flaw allows an unauthenticated attacker to extract user password hashes without logging in, provided WooCommerce was ever installed on the site. The vendor released a fully patched version (3.15.3) on May 12, 2026; any site still running version 3.15.2 or earlier is at risk.

Technical Analysis

Two distinct vulnerabilities affect the Avada Builder WordPress plugin. CVE-2026-4782 (CWE-22, path traversal) affects versions <= 3.15.2. Any authenticated user with subscriber-level access can supply a crafted path to read arbitrary server files, including wp-config.php, exposing database credentials, authentication keys, and salts (CWE-522). CVE-2026-4798 (CWE-89, time-based blind SQL injection) affects versions <= 3.15.1 and is conditionally exploitable: WooCommerce must have been installed and subsequently deactivated on the target site. An unauthenticated attacker can submit crafted SQL payloads to extract password hashes from the WordPress database. MITRE ATT&CK techniques mapped: T1190 (Exploit Public-Facing Application), T1078

(Valid Accounts), T1005 (Data from Local System), T1552/T1552.001 (Unsecured Credentials / Credentials in Files), T1083 (File and Directory Discovery), T1110.002 (Password Cracking). CVSS base score: 7.5 (High). EPSS score: 0.0004 (12th percentile as of scoring date). Not listed in CISA KEV. Patch: version 3.15.3, released 2026-05-12. Sources: Wordfence researcher disclosure; NVD records for CVE-2026-4782 and CVE-2026-4798.

Action Checklist

1. Containment. Identify all WordPress installations in your environment running Avada Builder <= 3.15.2. If immediate patching is not possible, restrict subscriber-level account creation, limit external access to wp-admin and xmlrpc.php endpoints via WAF or server configuration, and disable public user registration where operationally feasible.
2. Detection. Review web server access logs and WAF logs for requests containing path traversal patterns (e.g., '../', '%2e%2e%2f') targeting Avada Builder endpoints, and for unusual time-delayed SQL responses consistent with time-based blind injection. Query your WordPress installations for plugin version: run 'SELECT option_value FROM wp_options WHERE option_name = 'active_plugins'' and cross-reference against known vulnerable version strings (<= 3.15.2 for CVE-2026-4782; <= 3.15.1 for CVE-2026-4798). Check for WooCommerce presence in plugin history on all affected sites to assess CVE-2026-4798 exposure.
3. Eradication. Update Avada Builder to version 3.15.3 on all affected WordPress installations. Verify the update via the WordPress admin dashboard under Plugins > Installed Plugins or via WP-CLI: 'wp plugin get fusion-builder --field=version'. Confirm no residual vulnerable instances remain across staging, development, and production environments.
4. Recovery. After patching, rotate the database credentials and all cryptographic keys and salts stored in wp-config.php on any site where exploitation cannot be ruled out. Force a password reset for all WordPress user accounts on affected sites, prioritizing administrator and editor roles. Monitor authentication logs for anomalous login attempts using extracted hashes (T1110.002) in the 30 days following remediation.
5. Post-Incident. This incident exposes two control gaps: insufficient file access controls allowing low-privilege users to read sensitive configuration files (CWE-22/CWE-522), and inadequate input sanitization in database query handling (CWE-89). Review your WordPress plugin inventory and establish a recurring process to track plugin versions against Wordfence and NVD advisories. Evaluate whether subscriber-level account creation on public-facing WordPress sites is operationally necessary and restrict registration where it is not.

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to legal and executive leadership if forensic evidence (MySQL binlog, web server logs, WAF alerts) confirms that CVE-2026-4782 or CVE-2026-4798 was actively exploited — specifically if wp-config.php was read (database credential exposure) or if wp_users password hashes were extracted — as these conditions likely trigger breach notification obligations under GDPR Article 33, CCPA, or applicable state privacy laws given that user PII and authentication credentials are compromised at scale across up to one million potentially affected sites.

Recovery Notes	After patching to fusion-builder 3.15.3 and rotating all wp-config.php credentials and salts, verify recovery by confirming the new database user credentials connect successfully and that no legacy credential references remain in environment variables, caching layers (object cache, Redis, Memcached), or deployment configuration files outside wp-config.php. Monitor WordPress authentication logs (wp-login.php POST requests) and MySQL authentication logs for 30 days post-remediation for signs of T1110.002 credential stuffing using extracted hashes — attackers who obtained wp_users hashes via CVE-2026-4798 prior to patching will attempt offline cracking and reuse regardless of the patch. Any successful admin login from a previously unseen IP or user agent within this window should be treated as a confirmed compromise until ruled out.
Forensic Artifacts	Web server access logs (Apache access.log / nginx access.log): Filter for POST and GET requests to /wp-admin/admin-ajax.php with fusion-builder action parameters and to /wp-json/fusion-builder/ REST endpoints — successful CVE-2026-4782 exploitation (authenticated file read) will appear as HTTP 200 responses to these endpoints from subscriber-level authenticated sessions containing path traversal sequences targeting wp-config.php. | MySQL slow query log and binary log (binlog): CVE-2026-4798 time-based blind SQL injection against wp_users will produce repeated SLEEP() or BENCHMARK() function calls logged in the slow query log (queries >3-5 seconds); the binlog records all SELECT statements against wp_users and wp_usermeta, providing a definitive record of whether user password hash extraction occurred and from which database session. | wp-config.php file access timestamps: On Linux hosts, check inode change time ('stat /var/www/[site]/wp-config.php') and correlate with auditd logs (if enabled) for open/read syscalls by the web server process (www-data/apache) at times not corresponding to legitimate admin activity — direct evidence of CVE-2026-4782 file disclosure exploitation. | wp_options table — recently_activated field: This WordPress database record preserves a history of previously installed and deactivated plugins including WooCommerce; its presence is the prerequisite condition for CVE-2026-4798 exposure and confirms whether each site in the environment was vulnerable to the unauthenticated hash extraction flaw regardless of WooCommerce's current activation state. | wp_users table dump (user_pass column): The phpass-hashed passwords in wp_users are the primary target of CVE-2026-4798 exploitation; a pre-incident baseline dump (if available) compared against post-incident state confirms whether hashes were altered (indicating account takeover) or allows defenders to identify which accounts are at risk of offline cracking and prioritize forced password resets accordingly.

Per-Action IR Details

Containment — Identify all WordPress installations in your environment running Avada Builder <= 3.15.2. If immediate patching is not possible, restrict authenticated access to subscriber-level accounts on affected sites and consider temporarily blocking external access to wp-admin and xmlrpc.php endpoints via WAF or server configuration.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: NIST IR-4 (Incident Handling), NIST AC-3 (Access Enforcement), NIST SC-7 (Boundary Protection), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Run WP-CLI across all installations: 'wp plugin get fusion-builder --field=version --path=/path/to/wordpress' or iterate with 'for dir in /var/www/*/; do wp --path=$dir plugin get fusion-builder --field=version 2>/dev/null && echo $dir; done'. Block wp-admin and xmlrpc.php at the nginx/Apache level immediately: add 'location ~ ^/(wp-admin|xmlrpc.php) { deny all; }' for unauthenticated CVE-2026-4798 exposure, and use UFW or

iptables to allowlist only known admin IPs to wp-admin. For WooCommerce exposure assessment, check 'wp db query "SELECT option_value FROM wp_options WHERE option_name = 'active_plugins' OR option_name = 'inactive_plugins'"' to surface sites where WooCommerce appears in either active or deactivated state.

Evidence: Before restricting access, capture a full snapshot of wp-config.php permissions and ownership ('ls -la /var/www/[site]/wp-config.php'), the current Avada Builder (fusion-builder) plugin file manifest ('find /wp-content/plugins/fusion-builder/ -type f -newer /tmp/ref_timestamp'), and a dump of the wp_options active_plugins value. Preserve Apache/nginx access logs from the past 30 days before any rotation — specifically any requests hitting Avada Builder REST endpoints or admin-ajax.php actions registered by fusion-builder, which would be the attack surface for CVE-2026-4782 file read. This baseline is your pre-containment evidence record per NIST 800-61r3 §3.2.

Detection — Review web server access logs and WAF logs for requests containing path traversal patterns (e.g., '../', '%2e%2e%2f') targeting Avada Builder endpoints, and for unusual time-delayed SQL responses consistent with time-based blind injection. Query your WordPress installations for plugin version: run 'SELECT option_value FROM wp_options WHERE option_name = 'active_plugins'' and cross-reference against known vulnerable version strings (<= 3.15.2 for CVE-2026-4782; <= 3.15.1 for CVE-2026-4798). Check for WooCommerce presence in plugin history on all affected sites to assess CVE-2026-4798 exposure.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-3 (Content of Audit Records), NIST IR-5 (Incident Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Use grep against Apache/nginx access logs to surface exploitation attempts: 'grep -iE "(\.\./|%2e%2e%2f|%252e|fusion-builder|fusion_builder)" /var/log/apache2/access.log | grep -v "200"'. For time-based blind SQLi detection (CVE-2026-4798), extract requests with response times >5 seconds from access logs using awk: 'awk '$NF > 5 {print}' /var/log/nginx/access.log' (requires %D or response time in log format). Deploy a Sigma rule targeting web logs for Avada Builder REST API endpoint abuse — specifically POST/GET requests to '/wp-json/fusion-builder/' or 'admin-ajax.php?action=fusion_*' from unauthenticated sources (no valid session cookie). For WooCommerce history, query wp_options: 'SELECT option_value FROM wp_options WHERE option_name IN ("active_plugins", "recently_activated")' — recently_activated will surface WooCommerce even after deactivation, which is the trigger condition for CVE-2026-4798.

Evidence: Capture the following before any log rotation: (1) Full Apache/nginx access logs with timestamps, source IPs, user agents, HTTP response codes, and response times for all requests to wp-admin/admin-ajax.php and /wp-json/ endpoints; (2) WAF alert logs filtered for the Avada Builder plugin path or fusion-builder strings; (3) MySQL slow query log ('SHOW VARIABLES LIKE "slow_query_log%"') for queries exceeding 3-5 seconds originating from WordPress DB user — time-based blind injection for CVE-2026-4798 will appear here as repeated SLEEP() or BENCHMARK() calls against wp_users or wp_usermeta; (4) Output of 'SELECT user_login, user_registered, user_email FROM wp_users ORDER BY user_registered DESC LIMIT 50' to identify any accounts created during or after suspected exploitation window; (5) wp_options table dump of recently_activated to establish WooCommerce installation history.

Eradication — Update Avada Builder to version 3.15.3 on all affected WordPress installations. Verify the update via the WordPress admin dashboard under Plugins > Installed Plugins or via WP-CLI: 'wp plugin get fusion-builder --field=version'. Confirm no residual vulnerable instances remain across staging, development, and production environments.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST CM-3 (Configuration Change Control), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: For multi-site environments without a centralized plugin management platform, use WP-CLI in a bash loop: 'for dir in /var/www/*/; do wp --path=$dir plugin update fusion-builder --version=3.15.3 2>&1 | tee -a

/tmp/avada_patch_log.txt; done'. After updating, verify file integrity by comparing SHA256 hashes of the patched fusion-builder plugin files against the known-good 3.15.3 release package from wordpress.org. Run 'find /wp-content/plugins/fusion-builder/ -name "*.php" -exec sha256sum {} \;' and diff against the official release manifest. Check staging and dev environments explicitly — these are frequently overlooked and may share database credentials with production, making them viable pivot points for credential reuse.

Evidence: Before executing the update, preserve: (1) A file listing with timestamps of the current fusion-builder plugin directory ('find /wp-content/plugins/fusion-builder/ -type f -printf "%T@ %p\n" | sort') to establish a pre-patch baseline; (2) MD5/SHA256 checksums of key plugin files, particularly those handling file read operations or database queries (the vulnerable code paths in <= 3.15.2); (3) Database dump of wp_options and wp_users tables as a pre-eradication state record. Post-update, document the new version hash and update timestamp as evidence of remediation for audit purposes per NIST SI-2 (Flaw Remediation) documentation requirements.

Recovery — After patching, rotate the database credentials and all cryptographic keys and salts stored in wp-config.php on any site where exploitation cannot be ruled out. Force a password reset for all WordPress user accounts on affected sites, prioritizing administrator and editor roles. Monitor authentication logs for anomalous login attempts using extracted hashes (T1110.002) in the 30 days following remediation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IA-5 (Authenticator Management), NIST AC-2 (Account Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 5.2 (Use Unique Passwords), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Generate new WordPress secret keys and salts from the WordPress API (https://api.wordpress.org/secret-key/1.1/salt/) and replace the eight SECURE_AUTH_KEY, LOGGED_IN_KEY, AUTH_KEY, SECURE_AUTH_SALT, LOGGED_IN_SALT, AUTH_SALT, NONCE_KEY, and NONCE_SALT constants in wp-config.php — this invalidates all active sessions and cookies, forcing re-authentication. Rotate MySQL credentials: 'ALTER USER 'wpuser'@'localhost' IDENTIFIED BY '[new_strong_password]'; FLUSH PRIVILEGES;' then update wp-config.php DB_PASSWORD accordingly. Force a bulk password reset for all admin/editor accounts via WP-CLI: 'wp user list --role=administrator --field=ID | xargs -I {} wp user update {} --user_pass=$(openssl rand -base64 16)' and notify users via out-of-band email. For T1110.002 (Password Spraying with Extracted Hashes) monitoring, filter nginx/Apache auth logs and WordPress auth.log for repeated wp-login.php POST failures from single IPs or distributed IPs against known admin usernames.

Evidence: Before rotating credentials, capture: (1) The existing wp-config.php (redacted of secrets for documentation, but full copy retained in secured IR evidence store) to document what was potentially exposed via CVE-2026-4782 file read; (2) A full dump of wp_users including user_pass (hashed) columns — if CVE-2026-4798 was exploited, attackers have these hashes and may attempt offline cracking followed by credential stuffing (MITRE ATT&CK T1110.002); (3) Last login timestamps from wp_usermeta ('SELECT user_id, meta_value FROM wp_usermeta WHERE meta_key = "last_login"') to establish a baseline for detecting anomalous authentication post-incident. Preserve MySQL binary logs (binlog) if enabled, as these record all database queries and can confirm whether SELECT queries against wp_users were issued by unauthorized sessions.

Post-Incident — This incident exposes two control gaps: insufficient file access controls allowing low-privilege users to read sensitive configuration files (CWE-22/CWE-522), and inadequate input sanitization in database query handling (CWE-89). Review your WordPress plugin inventory and establish a recurring process to track plugin versions against Wordfence and NVD advisories. Evaluate whether subscriber-level account creation on public-facing WordPress sites is operationally necessary and restrict registration where it is not.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-10 (Information Input Validation), NIST RA-3 (Risk Assessment), NIST SI-2 (Flaw Remediation), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.1 (Establish and

Maintain a Software Inventory), CIS 5.1 (Establish and Maintain an Inventory of Accounts)

Compensating: Establish a lightweight WordPress plugin vulnerability tracking workflow: subscribe to Wordfence Intelligence RSS feed (https://www.wordfence.com/feed/) and NVD JSON data feed filtered on 'wordpress' CPE — parse daily with a simple Python script or cron job that alerts on any plugin in your installed inventory. Build a canonical plugin inventory using WP-CLI: 'for dir in /var/www/*/; do wp --path=$dir plugin list --format=json 2>/dev/null; done > /opt/monitoring/wp_plugin_inventory_$(date +%F).json' and diff against prior day. For subscriber account restriction, use WP-CLI: 'wp option update users_can_register 0 --path=/path/to/wordpress'. Implement a YARA rule targeting wp-config.php read patterns in web server processes to catch future CWE-22 exploitation attempts against this or similar plugins.

Evidence: Compile the post-incident evidence package: (1) Timeline of all requests to fusion-builder endpoints from web server logs correlated with wp_users query timestamps from MySQL binlog or slow query log; (2) List of all subscriber-level accounts that existed on affected sites during the vulnerability window (from wp_users filtered by user_registered date and role); (3) Documented wp-config.php contents (credential exposure scope assessment) to support breach notification evaluation if PII or payment data was accessible via the compromised database credentials; (4) Before-and-after plugin version records confirming fusion-builder upgrade from <= 3.15.2 to 3.15.3 with patch date; (5) Network flow records or access logs showing any outbound database connections or data exfiltration patterns from the web server host during the exposure window.

Detection Guidance

For CVE-2026-4782: Search web server and WAF logs for requests to Avada Builder REST API or AJAX endpoints containing path traversal sequences, '../', '..%2f', '%2e%2e/', or encoded variants. Flag any requests that result in a 200 response and return content matching wp-config.php patterns (e.g., 'DB_PASSWORD', 'AUTH_KEY', 'table_prefix'). For CVE-2026-4798: Look for anomalous HTTP requests with unusual response-time patterns (repeated slow responses of consistent latency intervals) against Avada Builder endpoints, which is characteristic of time-based blind SQL injection. Neither CVE has published IOC hashes or IP indicators as of the source date. Wordfence plugin users on supported plans received a firewall rule as of the disclosure date; confirm rule deployment in Wordfence dashboard under Firewall > Firewall Rules. Assess WooCommerce install history per site to scope CVE-2026-4798 exposure: check wp_options for residual WooCommerce option keys even if the plugin is deactivated.

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application
- **T1078** — Valid Accounts
- **T1005** — Data from Local System
- **T1552** — Unsecured Credentials
- **T1083** — File and Directory Discovery
- **T1552.001** — Credentials In Files
- **T1110.002** — Password Cracking

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning

- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-10** — Information Input Validation
- **CM-7** — Least Functionality
- **AC-3** — Access Enforcement
- **SC-13** — Cryptographic Protection

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A04:2021** — Insecure Design
- **A07:2021** — Identification and Authentication Failures
- **A01:2021** — Broken Access Control

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **5.2** — Use Unique Passwords
- **16.12** — Implement Code-Level Security Checks
- **6.3** — Require MFA for Externally-Exposed Applications
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information
- **A.8.24** — Use of cryptography
- **A.5.23** — Information security for use of cloud services

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access
T1078	Valid Accounts	Defense-Evasion
T1005	Data from Local System	Collection
T1552	Unsecured Credentials	Credential-Access
T1083	File and Directory Discovery	Discovery
T1552.001	Credentials In Files	Credential-Access
T1110.002	Password Cracking	Credential-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/avada-builder-wordpr...	T3
CVE-2026-4782 - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-4782	T1
CVE-2026-4782 | Tenable®	https://www.tenable.com/cve/CVE-2026-4782	T3
CVE-2026-4798 - CVE Record	https://www.cve.org/CVERecord?id=CVE-2026-4798	T3
1,000,000 WordPress Sites Affected by Arbitrary File Read and SQL ...	https://www.wordfence.com/blog/2026/05/1000000-wordpress-sites-affe...	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-4782, CVE-2026-4798	T1

Generated 2026-05-15 19:02 UTC by TJS Security Command Center