INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-07 05:59 UTC

Active Exploitation of Critical CVE-2026-3300 Vulnerability in Everest Forms Pro Plugin Threatens WordPress Sites Globally

CVE VULNERABILITY | **CRITICAL** | CVSS 9.8

SCC Item ID	SCC-CVE-2026-0271
Type	CVE Vulnerability
CVE ID	CVE-2026-3300
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0031 (55th percentile)
Affected Products	Everest Forms Pro WordPress Plugin (specific version range unconfirmed, source verification required)
Published	1 hour ago
Discovery Source	Serper

Executive Summary

A critical remote code execution vulnerability (CVE-2026-3300, CVSS 9.8) in the Everest Forms Pro WordPress plugin is being actively exploited, allowing unauthenticated attackers to seize full control of affected WordPress installations. Any organization running a public-facing WordPress site with this plugin installed faces immediate risk of complete site compromise, data theft, and malware deployment. Specific affected version ranges and patch availability require direct verification with Wordfence and the plugin vendor before definitive remediation can be confirmed.

Technical Analysis

CVE-2026-3300 is a critical-severity remote code execution vulnerability affecting the Everest Forms Pro WordPress plugin (CVSS 9.8). The vulnerability is associated with CWE-94 (Code Injection) and CWE-434 (Unrestricted Upload of File with Dangerous Type), indicating attackers can inject and execute arbitrary code, likely via malicious file upload through the plugin's form handling functionality. Exploitation maps to MITRE ATT&CK techniques T1190 (Exploit Public-Facing Application), T1059 (Command and Scripting Interpreter), and T1505.003 (Server Software Component: Web Shell), consistent with a post-exploitation web shell deployment pattern. No authentication or elevated privilege is required per current reporting. Wordfence, The Hacker News, SentinelOne, and Rescana have all reported active exploitation in the wild. IMPORTANT

VERIFICATION GAPS: Specific affected version ranges are unconfirmed in source data. NVD record was not available at time of curation. Patch availability and exploitation payload specifics require direct verification at the Wordfence advisory (wordfence.com) and NVD (nvd.nist.gov/vuln/detail/CVE-2026-3300) before acting on version-specific guidance. EPSS score of 0.00313 (54.8th percentile) reflects limited automated model data, likely due to recency; active exploitation reporting supersedes this score for prioritization purposes. Confirmation status: cross-referenced by Wordfence, Hacker News, and SentinelOne, but pending NVD official record publication for authoritative validation.

Action Checklist

1. Step 1: Containment. Immediately disable or remove the Everest Forms Pro plugin on all WordPress installations until patch status is confirmed. If disabling is not operationally feasible, place a WAF rule to block unauthenticated POST requests to plugin endpoints (wp-content/plugins/everest-forms-pro/*). Verify the specific affected version range directly at the Wordfence advisory before scoping containment. CIS 2.3 (Address Unauthorized Software) and NIST AC-4 (Information Flow Enforcement) support blocking or isolating the vulnerable component.
2. Step 2: Detection. Review web server access logs for anomalous POST requests targeting Everest Forms Pro form submission endpoints, file upload directories (wp-content/uploads/), and PHP execution in upload paths. Hunt for newly created .php files in wp-content/uploads/ and wp-content/plugins/everest-forms-pro/, these are web shell indicators consistent with T1505.003. Check for unexpected outbound connections from the web server process (T1059). CIS 8.2 (Collect Audit Logs) and NIST AU-6 (Audit Record Review, Analysis, and Reporting) govern this log review process. D3-SFA (System File Analysis) is the applicable D3FEND countermeasure for web shell detection.
3. Step 3: Eradication. Apply the vendor-released patch once confirmed via the Wordfence advisory and WordPress plugin repository. If a patch is not yet available, maintain plugin deactivation and implement WAF rules blocking file upload requests to the plugin. Remove any suspicious PHP files discovered in upload directories. Rotate credentials for the WordPress admin accounts and database user as a precautionary measure. NIST SI-2 (Flaw Remediation), CIS 7.3 (Perform Automated Operating System Patch Management), and CIS 7.4 (Perform Automated Application Patch Management) govern patch application and remediation.
4. Step 4: Recovery. After patching or confirmed safe plugin version is in place, scan all WordPress file directories for unauthorized modifications using a file integrity tool. Verify no new admin accounts were created (WordPress user table audit). Restore from a known-clean backup if web shell placement or unauthorized admin account creation is confirmed. Monitor web server logs for continued exploitation attempts post-remediation. NIST AU-9 (Protection of Audit Information) applies to ensuring post-recovery log integrity.
5. Step 5: Post-Incident. Evaluate the plugin vetting process for future third-party WordPress plugin adoption. Implement file integrity monitoring on WordPress installations to detect unauthorized file changes proactively. Ensure WAF coverage for all public-facing WordPress properties as a standing control. Review whether all externally exposed web applications have MFA enforced on administrative interfaces per CIS 6.3 (Require MFA for Externally-Exposed Applications) and CIS 6.5 (Require MFA for Administrative Access). Apply lessons from this incident to implement a formal plugin vetting and monitoring cadence to catch active vulnerability disclosures before they reach production.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and breach notification workflow immediately if forensic evidence confirms web shell deployment, unauthorized WordPress admin account creation, or database access — any of which indicate successful post-exploitation and potential exfiltration of PII, form submission data, or credentials stored in the WordPress database, which may trigger regulatory breach notification obligations under GDPR, CCPA, or HIPAA depending on data classification.
Recovery Notes	Before returning any patched WordPress instance to production, verify via WP-CLI that the installed Everest Forms Pro version matches the confirmed patched release from the Wordfence advisory, that no .php files exist in wp-content/uploads/, and that the wp_users table contains only pre-authorized administrator accounts. Post-recovery, maintain elevated log review frequency — daily review of web server access logs filtering on POST requests to wp-content/plugins/everest-forms-pro/ and wp-content/uploads/ — for a minimum of 30 days to detect re-exploitation attempts or persistence mechanisms not identified during eradication. If any evidence of successful pre-patch exploitation exists, treat the database as potentially compromised and audit wp_options for injected malicious serialized objects or rogue cron entries before certifying recovery complete.
Forensic Artifacts	Web server access logs (Apache: /var/log/apache2/access.log*, NGINX: /var/log/nginx/access.log*): filter for POST requests to paths matching 'everest-forms', 'evf', or 'evf-upload' endpoints — successful exploitation of CVE-2026-3300 via unauthenticated file upload would appear as POST requests returning HTTP 200 to the plugin's form handler, followed by subsequent GET requests to a .php file in wp-content/uploads/ indicating web shell access. | wp-content/uploads/ directory — filesystem timestamps and contents: CVE-2026-3300 RCE via file upload would result in attacker-controlled .php files written here; collect 'find wp-content/uploads/ -name "*.php" -ls' output and preserve file contents with hashes as primary web shell evidence. | WordPress database — wp_users and wp_usermeta tables: post-exploitation persistence commonly includes creation of rogue administrator accounts; export 'SELECT ID, user_login, user_registered, user_email FROM wp_users;' and compare against authorized account inventory to identify attacker-created accounts. | PHP error log and web server error log (/var/log/apache2/error.log or php_errors.log): failed or partially successful exploitation of CVE-2026-3300 would generate PHP warnings or fatal errors from the plugin's file handling code — these logs establish exploitation attempt timeline even when access logs show non-200 responses. | wp_options table — 'active_plugins', 'cron', and any option containing serialized PHP objects: post-exploitation persistence via WordPress may include malicious cron job registration or serialized backdoor payloads injected into wp_options; export with 'SELECT option_name, option_value FROM wp_options WHERE option_name IN ("active_plugins","cron") OR option_value LIKE "%eval%" OR option_value LIKE "%base64_decode%";' to surface injected persistence.

Per-Action IR Details

Step 1: Containment — Immediately disable or remove the Everest Forms Pro plugin on all WordPress installations until patch status is confirmed. If disabling is not operationally feasible, place a WAF rule to block unauthenticated POST requests to plugin endpoints (wp-content/plugins/everest-forms-pro/*). Verify the specific affected version range directly at the Wordfence advisory before scoping containment. CIS 2.3 (Address Unauthorized Software) and NIST AC-4 (Information Flow Enforcement) support blocking or isolating the vulnerable component.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 2.3 (Address Unauthorized Software)

Compensating: On Apache: add 'Deny from all' in .htaccess under wp-content/plugins/everest-forms-pro/ to block all HTTP access to plugin files. On NGINX: add 'location ~* ^/wp-content/plugins/everest-forms-pro/ { deny all; }' in the server block. For WAF-less environments, use ModSecurity (free) with a rule matching 'REQUEST_URI' containing 'everest-forms-pro' and METHOD 'POST' — action DENY. Verify block with: 'curl -X POST https://[target]/wp-content/plugins/everest-forms-pro/index.php' and confirm 403 response. Two-person task: one implements rule, second validates from an external IP.

Evidence: Before disabling the plugin, capture: full directory listing with timestamps of wp-content/plugins/everest-forms-pro/ ('ls -laR' or 'dir /T:W /S' on Windows), MD5/SHA256 hashes of all .php files in the plugin directory ('find wp-content/plugins/everest-forms-pro/ -name "*.php" -exec md5sum {} \;'), current WordPress active plugin list from wp_options table ('SELECT option_value FROM wp_options WHERE option_name = "active_plugins";'), and a snapshot of wp-content/uploads/ with file timestamps to establish a pre-containment baseline for later comparison.

Step 2: Detection — Review web server access logs for anomalous POST requests targeting Everest Forms Pro form submission endpoints, file upload directories (wp-content/uploads/), and PHP execution in upload paths. Hunt for newly created .php files in wp-content/uploads/ and wp-content/plugins/everest-forms-pro/ — these are web shell indicators consistent with T1505.003. Check for unexpected outbound connections from the web server process (T1059). CIS 8.2 (Collect Audit Logs) and NIST AU-6 (Audit Record Review, Analysis, and Reporting) govern this log review process. D3-SFA (System File Analysis) is the applicable D3FEND countermeasure for web shell detection.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs)

Compensating: Run this grep against Apache/NGINX access logs to surface exploitation attempts targeting CVE-2026-3300 form submission endpoints: 'grep -E "POST.*(everest-forms|evf-upload|evf_nonce|wp-content/uploads/).*(200|201|500)" /var/log/apache2/access.log'. Identify PHP files written to upload directories post-exploitation: 'find wp-content/uploads/ -name "*.php" -newer wp-config.php -ls'. For outbound callback detection (post-exploit reverse shell), run 'ss -tulpn' and 'netstat -anp | grep php' to identify PHP processes with established outbound connections. Use YARA rule targeting web shell signatures (e.g., eval(base64_decode), system(), passthru() in files under wp-content/uploads/) — YARA is free and can be run as: 'yara webshell_rules.yar wp-content/uploads/'.

Evidence: Preserve before analysis: raw web server access logs covering at least 14 days prior to discovery (Apache: /var/log/apache2/access.log*, NGINX: /var/log/nginx/access.log*), WAF logs if present filtering on URIs containing 'everest-forms', 'evf', or 'wp-content/uploads' with POST method, PHP error logs (/var/log/apache2/error.log or php_errors.log) for file write or include errors indicative of exploit failure or success, and output of 'find wp-content/ -name "*.php" -newer wp-settings.php' to identify files written after the plugin was installed or after the CVE-2026-3300 public disclosure date.

Step 3: Eradication — Apply the vendor-released patch once confirmed via the Wordfence advisory and WordPress plugin repository. If a patch is not yet available, maintain plugin deactivation and implement WAF rules blocking file upload requests to the plugin. Remove any suspicious PHP files discovered in upload directories. Rotate credentials for the WordPress admin accounts and database user as a precautionary measure. NIST SI-2 (Flaw Remediation) governs patch application — note: SI-2 is not in the provided knowledge base control set; no mapped control from the verified knowledge base applies specifically to patch application here. CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management) are the closest verified mapped controls for the remediation process.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Apply the patched Everest Forms Pro version via WP-CLI to avoid interactive login on a potentially compromised admin panel: 'wp plugin update everest-forms-pro --version=[patched-version] --path=/var/www/html'. Before running, verify the plugin slug and patched version number against the Wordfence advisory. Remove discovered web shells: 'find wp-content/uploads/ -name "*.php" -delete' — log each deleted file path and hash before removal. Rotate WordPress admin passwords via WP-CLI: 'wp user update [admin-user] --user_pass="[new-strong-password]"'. Rotate the database user password in wp-config.php and in the MySQL/MariaDB server: 'ALTER USER "wpdbuser"@"localhost" IDENTIFIED BY "[new-password]";'. Confirm no additional admin accounts exist: 'wp user list --role=administrator'.

Evidence: Before eradication actions begin, preserve forensic copies of: every .php file identified in wp-content/uploads/ (these are potential web shells — hash and archive before deletion: 'tar czf webshells-evidence-[date].tar.gz [file list]'), the full WordPress database dump ('wp db export incident-db-[date].sql') to preserve any attacker-created admin accounts, persistent backdoors injected into wp_options (malicious cron jobs or serialized payloads), and modified plugin files, and the output of 'wp user list --format=csv > user-audit-[date].csv' to document account state at time of eradication.

Step 4: Recovery — After patching or confirmed safe plugin version is in place, scan all WordPress file directories for unauthorized modifications using a file integrity tool. Verify no new admin accounts were created (WordPress user table audit). Restore from a known-clean backup if web shell placement or unauthorized admin account creation is confirmed. Monitor web server logs for continued exploitation attempts post-remediation. NIST AU-9 (Protection of Audit Information) applies to ensuring post-recovery log integrity. D3-LAM (Local Account Monitoring) supports detecting unauthorized account creation during and after recovery.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-9 (Protection Of Audit Information)

Compensating: Run Wordfence (free tier) or WPScan ('wpscan --url https://[target] --enumerate u,p,t') post-patch to confirm no residual web shells or rogue plugins remain. For file integrity, generate a fresh baseline hash manifest after patching: 'find /var/www/html/wp-content/ -type f -exec sha256sum {} \; > baseline-post-patch-[date].txt' and diff against the pre-incident baseline. Audit the wp_users and wp_usermeta tables directly: 'wp user list --role=administrator --fields=ID,user_login,user_registered,user_email'. Forward web server access logs to an append-only remote syslog destination (rsyslog with remote target) to satisfy AU-9 log integrity requirements without a SIEM.

Evidence: Before restoring from backup or certifying recovery, document: the diff output between pre-incident file hash baseline and post-eradication state to confirm all unauthorized files were removed, the WordPress user table export ('wp user list --format=csv') showing current admin accounts compared to the pre-incident account inventory, web server access logs from the 24-hour window immediately following patch application to detect resumed CVE-2026-3300 exploitation attempts against the now-patched endpoint, and confirmation that wp-config.php database credentials match the rotated credentials set during eradication (verify the file was not re-modified post-rotation).

Step 5: Post-Incident — Evaluate the plugin vetting process for future third-party WordPress plugin adoption. Implement file integrity monitoring on WordPress installations to detect unauthorized file changes proactively. Ensure WAF coverage for all public-facing WordPress properties as a standing control. Review whether all externally exposed web applications have MFA enforced on administrative interfaces per CIS 6.3 (Require MFA for Externally-Exposed Applications) and CIS 6.5 (Require MFA for Administrative Access). Apply D3-MFA (Multi-factor Authentication) and D3-CH (Credential Hardening) as standing improvements. This incident exposes a gap in third-party plugin risk management — document a formal review cadence for

installed plugins against active vulnerability disclosures.

NIST Phase: Post Incident

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

Controls: CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), 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 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Establish a recurring monthly task: run 'wp plugin list --format=csv' across all WordPress installations and cross-reference each plugin slug against the WPVulnDB API (free tier available) or the Wordfence Intelligence feed for active CVEs. Implement file integrity monitoring using the free Wordfence plugin (file change detection feature) or inotifywait on Linux: 'inotifywait -m -r -e create,modify,delete wp-content/ --format "%T %w%f %e" --timefmt "%Y-%m-%d %H:%M"' piped to a log file. For wp-admin MFA, deploy the free WP 2FA plugin or enforce MFA at the SSO/IdP level if WordPress is federated. Document the plugin vetting checklist to include: active maintenance status, CVE history, install count, and last audit date — review before any new plugin is approved for production.

Evidence: For the lessons-learned record, compile: the full timeline reconstructed from web server access logs showing first observed CVE-2026-3300 exploitation attempt through containment, the complete list of WordPress installations in the environment cross-referenced against which were running Everest Forms Pro (from CIS 2.1 software inventory — identify the inventory gap this incident exposed), the list of any rogue admin accounts or web shells discovered during the incident with their creation timestamps from the WordPress database and file system respectively, and a gap analysis comparing time-to-detection against the CVE-2026-3300 public disclosure date to measure detection latency and inform future monitoring thresholds.

Detection Guidance

Primary detection focus: web shell deployment and unauthenticated file upload activity consistent with CWE-434 exploitation. Query web server access logs (Apache access.log, nginx access.log, or equivalent) for: POST requests to paths matching */everest-forms-pro/* or */wp-admin/admin-ajax.php* with form upload parameters from unauthenticated sessions (no valid session cookie). File system: search for .php files created or modified after the plugin's known installation date within wp-content/uploads/; PHP execution in this directory indicates successful exploitation. Process monitoring: flag unexpected child processes spawned by the web server user (www-data, apache, nginx), indicative of T1059 command execution post-RCE. WordPress audit logs (if enabled via a logging plugin): flag new administrator account creation, theme/plugin file edits, or option value changes following form submission events. SIEM correlation: chain T1190 (exploit attempt in access logs) → T1059 (process execution) → T1505.003 (new PHP file in upload directory) as a high-confidence detection sequence. D3-SFA (System File Analysis) is the primary D3FEND countermeasure. IOCs: specific payload hashes, C2 domains, and web shell signatures are not confirmed in source data; monitor Wordfence Threat Intelligence feed for updates as active exploitation details emerge. NVD and CISA KEV records were not available at curation time; check both sources for updated detection signatures before finalizing rules.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`wp-content/uploads/*.php`	PHP files in upload directory indicate web shell deployment post-exploitation of CWE-434 unrestricted file upload; pattern-based, not a confirmed hash	MEDIUM

Type	Value	Context	Confidence
URL	`wp-content/plugins/everest-forms-pro/*`	Unauthenticated POST requests to plugin endpoints are the likely exploit delivery path per CWE-94/CWE-434 vulnerability class	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1059** — Command and Scripting Interpreter
- **T1190** — Exploit Public-Facing Application
- **T1505.003** — Web Shell

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **CM-2** — Baseline Configuration
- **SI-10** — Information Input Validation
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A04:2021** — Insecure Design

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

NIST-CSF-2

- **DE.AE-08** — Incidents are declared when adverse events meet the defined incident criteria

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1059	Command and Scripting Interpreter	Execution
T1190	Exploit Public-Facing Application	Initial-Access
T1505.003	Web Shell	Persistence

Sources

Source	URL	Tier
	https://www.rescana.com/post/active-exploitation-of-critical-cve-20...	T3
(consolidated)	https://thehackernews.com/2026/06/hackers-exploit-critical-everest-...	T3
Attackers Actively Exploiting Critical Vulnerability in Everest Forms ...	https://www.wordfence.com/blog/2026/06/attackers-actively-exploitin...	T3
CVE-2026-3300: Everest Forms Pro WordPress RCE Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2026-3300/	T3
Critical Everest Forms Pro flaw exploited to take over WordPress sites	https://www.instagram.com/p/DZP4xmvDiG8/	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-3300	T1

Generated 2026-06-07 05:59 UTC by TJS Security Command Center