INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-23 13:39 UTC

CVE-2026-3844: Cloudways Breeze Cache Plugin, Unauthenticated Arbitrary File Upload via Gravatar Function

CVE VULNERABILITY | CRITICAL | CVSS 9.8 | CISA KEV

SCC Item ID	SCC-CVE-2026-0073
Type	CVE Vulnerability
CVE ID	CVE-2026-3844
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0006 (19th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability
Affected Products	Cloudways Breeze Cache (WordPress plugin), all versions up to and including 2.4.4
Published	2026-04-23T00:00:00Z
Discovery Source	Vulncheck Kev

Executive Summary

A critical unauthenticated file upload vulnerability in the Cloudways Breeze Cache WordPress plugin (versions up to 2.4.4) allows attackers to upload malicious files to affected servers without any login credentials, potentially enabling full server compromise. Any WordPress site using Breeze Cache with the 'Host Files Locally - Gravatars' feature enabled is at direct risk of remote code execution. CISA and VulnCheck have both confirmed active exploitation in the wild, making this an immediate patching priority.

Technical Analysis

CVE-2026-3844 is a missing file type validation flaw (CWE-434) in the 'fetch_gravatar_from_remote' function of the Cloudways Breeze Cache WordPress plugin. Affected versions: all releases up to and including 2.4.4. The vulnerability is exploitable by unauthenticated attackers when the 'Host Files Locally - Gravatars' feature is enabled (disabled by default). An attacker can craft a request to the vulnerable function to upload an arbitrary file, including a web shell, to the server, achieving remote code execution (RCE). CVSS base score: 9.8 (Critical). MITRE ATT&CK mappings: T1505.003 (Server Software Component: Web Shell), T1190 (Exploit Public-Facing Application). Confirmed in both CISA KEV and VulnCheck KEV, indicating active in-the-wild

exploitation. CVSS vendor score not yet published by NVD; vendor advisory score pending. Patch status: update to Breeze 2.4.5 or later; verify the fixed release version in the WordPress plugin repository changelog.

Action Checklist

1. Step 1: Containment, Immediately identify all WordPress installations running Breeze Cache ≤ 2.4.4. If the 'Host Files Locally - Gravatars' feature is enabled, disable it now via the Breeze plugin settings panel. If patching cannot occur within hours, consider temporarily deactivating the plugin entirely on internet-facing sites.
2. Step 2: Detection, Search your web server access logs for anomalous POST requests targeting the Breeze Cache Gravatar-related endpoints (look for requests to wp-content/plugins/breeze/ paths). Examine request bodies for embedded executable file extensions (.php, .phtml, .php5, .phar). Review the WordPress uploads directory and the Breeze plugin's local Gravatar cache directory for recently written files with executable extensions. Check for new or modified files in wp-content/ directories created after the plugin was installed.
3. Step 3: Eradication, Update the Breeze Cache plugin to version 2.4.5 or later via the WordPress admin dashboard or WP-CLI ('wp plugin update breeze'). Verify the fixed version in the WordPress plugin repository changelog. Remove any suspicious files discovered during detection, particularly web shells in the Gravatar cache or uploads directories.
4. Step 4: Recovery, After patching, rescan the web root for web shells using a file integrity tool (e.g., Wordfence, WP-CLI checksums, or a host-based scanner). Confirm the 'Host Files Locally - Gravatars' setting remains disabled unless the patched version explicitly clears the risk. Monitor server process trees for unexpected child processes spawned by the web server user (e.g., www-data spawning bash or curl), which may indicate a pre-patch compromise.
5. Step 5: Post-Incident, Review plugin vetting and update processes: this class of vulnerability (CWE-434, missing file type validation in media-handling functions) is recurrent in WordPress plugins. Implement a WAF rule blocking file uploads with executable extensions to non-upload endpoints as a defense-in-depth control. Evaluate whether plugin auto-updates are feasible for your WordPress fleet to reduce exposure windows for future critical CVEs.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership, legal counsel, and potentially CISA (if critical infrastructure) if any of the following are confirmed: discovery of a web shell with evidence of execution (non-zero HTTP 200 response to shell path in access logs), evidence of lateral movement or privilege escalation beyond the www-data process context, presence of PII or PHI in the WordPress database or file system triggering GDPR/HIPAA breach notification obligations, or if more than one WordPress instance in the fleet shows indicators of compromise given CISA-confirmed active exploitation in the wild.

Recovery Notes	After eradication, maintain elevated monitoring of web server process trees and the 'wp-content/plugins/breeze/cache/gravatar/' directory for a minimum of 30 days, as web shells dropped before containment may have registered WordPress cron jobs or system crontab entries ('crontab -l -u www-data') that survive plugin removal and attempt periodic callbacks. Verify the patched Breeze plugin version explicitly disables or validates file type on the Gravatar local hosting function — if the vendor changelog does not explicitly reference CVE-2026-3844, treat the patch as unverified and keep the feature disabled. Confirm through 'wp option get' that the 'Host Files Locally - Gravatars' option has not been re-enabled by plugin initialization logic after the update.
Forensic Artifacts	Web server access logs (Apache /var/log/apache2/access.log or Nginx /var/log/nginx/access.log): Filter for POST requests to paths containing 'wp-content/plugins/breeze/' with HTTP 200 response codes — these represent the unauthenticated file upload requests that CVE-2026-3844 enables, and successful responses confirm exploitation rather than mere scanning. | Breeze plugin local Gravatar cache directory (typically 'wp-content/plugins/breeze/cache/gravatar/' or a configured custom path): Files with .php, .phtml, .phar, .php5 extensions, or .htaccess files with AddType directives, are direct artifacts of the CVE-2026-3844 upload mechanism and the attacker's post-exploitation web shell staging. | WordPress database wp_options table: The 'breeze_advanced_settings' option row containing the serialized plugin configuration, specifically the 'local_gravatars' key, establishes whether the vulnerable feature was enabled at time of exploitation — critical for scoping the incident and regulatory disclosure decisions. | PHP-FPM or mod_php error logs (/var/log/php-fpm/www-error.log or Apache error.log): Execution errors or fatal errors referencing file paths within the Breeze Gravatar cache directory indicate attempted or successful web shell execution post-upload, corroborating active post-exploitation activity beyond the initial file upload. | Auditd or Sysmon for Linux execve records for UID 33 (www-data): Process execution events showing the web server user spawning bash, sh, curl, wget, python, or nc are the primary indicator of web shell command execution following a successful CVE-2026-3844 upload, and distinguish active compromise from dormant shell staging.

Per-Action IR Details

Step 1: Containment — Immediately identify all WordPress installations running Breeze Cache ≤ 2.4.4. If the 'Host Files Locally - Gravatars' feature is enabled, disable it now via the Breeze plugin settings panel. If patching cannot occur within hours, consider temporarily deactivating the plugin entirely on internet-facing sites.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-7 (Least Functionality), CIS 2.3 (Address Unauthorized Software), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Run 'wp plugin list --name=breeze --fields=name,version,status --format=table' via WP-CLI across all managed WordPress installs to enumerate affected instances. If WP-CLI is unavailable, query the database directly: 'SELECT option_value FROM wp_options WHERE option_name = "active_plugins";' and grep for 'breeze'. To disable the Gravatar feature without admin UI access, run: 'wp option patch update breeze_advanced_settings local_gravatars 0' or set the corresponding option in wp_options directly. As an emergency network-layer control, use ModSecurity or Apache/Nginx deny rules to block POST requests to 'wp-content/plugins/breeze/' paths until the plugin is patched or deactivated.

Evidence: Before disabling the plugin, capture: (1) a full directory listing with timestamps of 'wp-content/plugins/breeze/cache/gravatar/' using 'ls -laR --time-style=full-iso' to establish a pre-containment baseline; (2) the current value of the Breeze Gravatar option from the database ('SELECT option_name, option_value FROM

wp_options WHERE option_name LIKE "%breeze%';'); (3) a snapshot of currently running web server processes ('ps auxf | grep www-data') to detect any already-spawned shells; (4) web server access logs (Apache: /var/log/apache2/access.log, Nginx: /var/log/nginx/access.log) covering the period from Breeze plugin installation date through now, preserved offline before any rotation occurs.

Step 2: Detection — Search your web server access logs for anomalous POST requests targeting the Breeze Cache Gravatar-related endpoints (look for requests to wp-content/plugins/breeze/ paths with unexpected file extensions in the body). Review the WordPress uploads directory and the Breeze plugin's local Gravatar cache directory for recently written files with executable extensions (.php, .phtml, .php5, .phar). Check for new or modified files in wp-content/ directories created after the plugin was installed.

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-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Run the following to find suspicious files written to Breeze's Gravatar cache and wp-content since plugin install date: 'find /var/www/html/wp-content/ -newer /var/www/html/wp-content/plugins/breeze/breeze.php -type f \(-name "*.php" -o -name "*.phtml" -o -name "*.phar" -o -name "*.php5" \) -ls'. For log analysis without a SIEM, parse access logs with: 'grep -E "POST.*wp-content/plugins/breeze/" /var/log/nginx/access.log | grep -v " 404 "' to surface successful uploads. Deploy a YARA rule scanning the Gravatar cache directory targeting PHP web shell signatures (e.g., 'eval(base64_decode', 'system($_', 'passthru($_'). The Sigma rule 'web_shell_detection' from SigmaHQ can be converted to grep-compatible patterns for teams without a SIEM.

Evidence: Capture before analysis: (1) web server access logs filtered for POST requests to any path containing 'breeze' or the configured local Gravatar directory path, with HTTP response codes 200 and 201 indicating successful upload; (2) file metadata (inode creation time, mtime, atime) for all files in 'wp-content/plugins/breeze/' and the Gravatar cache subdirectory using 'stat' on each file — inode creation time is harder to spoof than mtime; (3) PHP-FPM or mod_php error logs (/var/log/php-fpm/error.log or Apache error.log) for execution errors that would appear if a newly uploaded shell was invoked unsuccessfully; (4) any WAF logs (ModSecurity audit log at /var/log/modsec_audit.log) capturing the multipart POST body content of requests to Breeze endpoints.

Step 3: Eradication — Update the Breeze Cache plugin to the latest version confirmed to patch CVE-2026-3844 via the WordPress admin dashboard or WP-CLI ('wp plugin update breeze'). Verify the fixed version in the WordPress plugin repository changelog. Remove any suspicious files discovered during detection, particularly web shells in the Gravatar cache or uploads directories.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST CM-7 (Least Functionality), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Apply the patch via WP-CLI: 'wp plugin update breeze --version=' and confirm with 'wp plugin get breeze --field=version'. Verify the patched version against the WordPress.org plugin repository changelog at wordpress.org/plugins/breeze/#developers — confirm the version explicitly references CVE-2026-3844 in its changelog entry. For web shell removal, do not simply delete discovered files — first copy them to an offline evidence directory ('cp -a /var/www/html/wp-content/plugins/breeze/cache/gravatar/ /root/ir-evidence/breeze-gravatar-$(date +%Y%m%d)/') before deletion to preserve forensic artifacts. Use ClamAV ('clamscan -r --detect-pua /var/www/html/wp-content/') to surface additional malicious files beyond those with obvious executable extensions, as attackers may use double extensions (e.g., 'image.jpg.php') or htaccess manipulation to execute non-.php files.

Evidence: Before eradication, preserve: (1) full copies of all suspicious files found in 'wp-content/plugins/breeze/cache/gravatar/' and 'wp-content/uploads/' with their original directory structure and timestamps intact; (2) a cryptographic hash (SHA-256) of each suspicious file ('sha256sum ') for evidentiary chain of custody and future threat intelligence sharing; (3) any .htaccess files in the Breeze plugin directory or Gravatar cache directory that may have been modified to permit execution of uploaded files (attackers commonly drop 'AddType

application/x-httpd-php .jpg' htaccess rules alongside shells); (4) the output of 'wp cron event list' to check for malicious cron jobs registered by a web shell to maintain persistence.

Step 4: Recovery — After patching, rescan the web root for web shells using a file integrity tool (e.g., Wordfence, WP-CLI checksums, or a host-based scanner). Confirm the 'Host Files Locally - Gravatars' setting remains disabled unless the patched version explicitly clears the risk. Monitor server process trees for unexpected child processes spawned by the web server user (e.g., www-data spawning bash or curl), which may indicate a pre-patch compromise.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST SI-4 (System Monitoring), NIST IR-4 (Incident Handling), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Run 'wp core verify-checksums' and 'wp plugin verify-checksums breeze' via WP-CLI to validate plugin file integrity against WordPress.org repository hashes. For the web root scan, use 'php -l' recursively to syntax-check all PHP files ('find /var/www/html -name "*.php" -exec php -l {} \; 2>&1 | grep -v "No syntax errors"') — obfuscated web shells will frequently fail PHP syntax checks or reveal eval/base64 patterns. To monitor for www-data spawning interactive shells post-patch, deploy Sysmon for Linux (or auditd rules) with a rule targeting execve calls by uid matching www-data: 'auditctl -a always,exit -F arch=b64 -S execve -F uid=33 -k webshell_exec'. Alert on www-data spawning bash, sh, curl, wget, nc, or python processes.

Evidence: Before declaring recovery complete, capture: (1) a post-patch 'wp plugin verify-checksums breeze' output confirming all plugin files match repository hashes; (2) a comparative file listing of 'wp-content/plugins/breeze/cache/gravatar/' between the pre-containment snapshot (captured in Step 1) and the post-eradication state to confirm all unauthorized files are removed; (3) web server access logs from the 24–72 hours post-patch, specifically filtering for any HTTP 200 responses to paths within the Breeze plugin or Gravatar cache directory that were not present pre-compromise; (4) auditd or Sysmon logs showing process ancestry for the www-data user to confirm no persistent shell callbacks are occurring after eradication.

Step 5: Post-Incident — Review plugin vetting and update processes: this class of vulnerability (CWE-434, missing file type validation in media-handling functions) is recurrent in WordPress plugins. Implement a WAF rule blocking file uploads with executable extensions to non-upload endpoints as a defense-in-depth control. Evaluate whether plugin auto-updates are feasible for your WordPress fleet to reduce exposure windows for future critical CVEs.

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 SI-5 (Security Alerts, Advisories, and Directives), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For WAF rule implementation without a commercial WAF, add a ModSecurity rule (or Nginx location block) that returns 403 on POST requests to any path under 'wp-content/plugins/' with a Content-Type of 'multipart/form-data' and a filename parameter containing extensions .php, .phtml, .phar, .php5, .php7. Example Nginx snippet: 'location ~* ^/wp-content/plugins/.*\.(php|phtml|phar|php5)$ { deny all; }'. For plugin vetting, subscribe to the WPScan Vulnerability Database API (free tier available at wpscan.com) and configure a weekly cron job running 'wpscan --url --api-token ' to surface newly disclosed plugin CVEs before the next patch cycle. Enable WordPress auto-updates for plugins by adding 'add_filter("auto_update_plugins", "__return_true");' to wp-config.php or manage via WP-CLI: 'wp plugin auto-updates enable --all'.

Evidence: For the post-incident lessons-learned record, preserve: (1) the timeline of Breeze Cache plugin installation date versus CVE-2026-3844 disclosure date, sourced from 'wp plugin get breeze --field=update_date' and WordPress.org changelog, to calculate the exposure window; (2) the full web server access log archive covering the window between plugin install and containment, stored offline and hashed, to support any future regulatory notification determination; (3) documentation of the Breeze plugin's 'Host Files Locally - Gravatars' configuration state at time of discovery (from the wp_options database snapshot captured in Step 1) to establish whether the vulnerable feature was

active and the organization's exposure was confirmed versus theoretical.

Detection Guidance

Check web server access logs (Apache access.log or Nginx access.log) for POST requests to Breeze plugin paths, particularly any endpoint associated with the Gravatar fetch function, from unauthenticated sessions (no valid WordPress auth cookie). Look for HTTP 200 responses to these requests combined with subsequent GET requests to newly created files in wp-content/plugins/breeze/ or wp-content/uploads/ directories. On the filesystem, audit for files with double extensions (e.g., image.php.jpg misidentified by a broken validator) or unexpected .php files in Gravatar cache directories. Behavioral indicator: web server worker processes (www-data, apache) spawning shell interpreters or making outbound network connections to non-CDN IP addresses. If you have EDR coverage on the WordPress host, alert on file writes to wp-content/ paths followed by execution of that file by the web server process.

Framework Mappings

MITRE-ATTACK

- **T1505.003** — Web Shell
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **CM-2** — Baseline Configuration
- **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
- **SI-10** — Information Input Validation
- **IR-5** — Incident Monitoring

OWASP-TOP10-2021

- **A04:2021** — Insecure Design

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.23** — Information security for use of cloud services

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
T1505.003	Web Shell	Persistence
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
vulncheck_kev	https://nvd.nist.gov/vuln/detail/CVE-2026-3844	T1
CVE-2026-3844 | Tenable®	https://www.tenable.com/cve/CVE-2026-3844	T3
CVE-2026-3844 - Exploits & Severity - Feedly	https://feedly.com/cve/CVE-2026-3844	T3
CVE-2026-3844 - Critical Vulnerability - TheHackerWire	https://www.thehackerwire.com/vulnerability/CVE-2026-3844/	T3
CVE-2026-3844 - CVE Record	https://www.cve.org/CVERecord?id=CVE-2026-3844	T3
CISA KEV	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1

Generated 2026-04-23 13:39 UTC by TJS Security Command Center