INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-30 14:08 UTC

ThemeREX Addons WordPress Plugin - Unauthenticated Arbitrary File Upload (CVE-2026-1969)

CVE VULNERABILITY | **CRITICAL** | CVSS 9.8 | **CISA KEV**

SCC Item ID	SCC-CVE-2026-0098
Type	CVE Vulnerability
CVE ID	CVE-2026-1969
Severity	CRITICAL
CVSS Base Score	9.8
EPSS Score	0.0007 (22th percentile)
KEV Status	Yes — CISA Known Exploited Vulnerability
Affected Products	ThemeREX Addons (trx_addons) WordPress plugin before version 2.38.5
Published	2026-04-30T00:00:00Z
Discovery Source	Vulncheck Kev

Executive Summary

A critical vulnerability in the ThemeREX Addons WordPress plugin (CVE-2026-1969) allows any unauthenticated attacker to upload malicious files directly to affected web servers, bypassing all login requirements. Organizations running WordPress sites with the trx_addons plugin prior to version 2.38.5 face immediate risk of full server compromise, including data theft and ransomware deployment. CISA KEV and VulnCheck KEV both list this vulnerability with confirmed active exploitation, meaning attacks are occurring now, not hypothetically.

Technical Analysis

CVE-2026-1969 affects the ThemeREX Addons (trx_addons) WordPress plugin before version 2.38.5. The flaw resides in an AJAX action handler that fails to validate file types (CWE-434: Unrestricted Upload of File with Dangerous Type), allowing unauthenticated HTTP POST requests to deposit arbitrary files, including PHP web shells, on the server. This vulnerability is a failed patch regression: it results from an incomplete fix applied to CVE-2024-13448, which addressed a prior file upload flaw in the same plugin. Successful exploitation maps to MITRE ATT&CK T1190 (Exploit Public-Facing Application) for initial access and T1505.003 (Server Software Component: Web Shell) for persistence. CVSS Base Score: 9.8 (Critical), confirmed by NVD assessment. The

CVSS vector string is pending publication from authoritative sources; severity is corroborated by CISA KEV listing. The AJAX endpoint requires no authentication, making mass automated exploitation trivial. The vulnerability is listed in both CISA KEV and VulnCheck KEV, confirming in-the-wild exploitation. EPSS score is 0.00074 (22nd percentile) as of data collection, though KEV listing supersedes EPSS as a prioritization signal. Public IOCs (malicious IPs, file hashes, C2 domains) have not been published for this vulnerability as of the data collection date; prioritize detection of the attack vector (unauthenticated file upload) rather than known-bad indicators. Patch: upgrade to trx_addons version 2.38.5 or later.

Action Checklist

1. Step 1 (Containment), Immediately identify all WordPress instances running trx_addons (ThemeREX Addons) plugin versions prior to 2.38.5 across your environment. If patching cannot occur within hours, use your WAF to block unauthenticated POST requests to WordPress AJAX endpoints (wp-admin/admin-ajax.php) targeting trx_addons action handlers as an interim measure.
2. Step 2 (Detection), Search web server access logs for POST requests to wp-admin/admin-ajax.php with trx_addons-related action parameters originating from external IPs, particularly those with non-standard file extensions in the request body. Scan the WordPress uploads directory and plugin directories for newly created PHP files, especially files with names inconsistent with media uploads. Check file system modification timestamps for any .php files created in wp-content/ after your last known-clean baseline.
3. Step 3 (Eradication), Update the trx_addons plugin to version 2.38.5 or later via the WordPress admin panel or WP-CLI. If a web shell was discovered, remove the malicious file, rotate all WordPress admin credentials, database credentials, and hosting credentials. Audit all user accounts on the affected WordPress instance for unauthorized additions.
4. Step 4 (Recovery), After patching, confirm the plugin version in the WordPress admin panel under Plugins. Re-scan the server file system for residual web shells using a file integrity monitoring tool or a WordPress security scanner (e.g., Wordfence, WPScan). Monitor web server logs and WAF alerts for continued exploitation attempts against the patched endpoint for at least 72 hours post-remediation.
5. Step 5 (Post-Incident), Review your WordPress plugin update workflow: this vulnerability is a failed patch regression from CVE-2024-13448, indicating the prior fix was not validated. Implement mandatory regression testing or vendor patch verification before closing CVEs in plugins with prior file upload history. Evaluate whether unauthenticated AJAX endpoints across your WordPress estate require stricter WAF rules as a standing control.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership and legal/compliance immediately if forensic evidence confirms a web shell was successfully deployed (HTTP 200 on upload request + PHP file present in wp-content/), any WordPress database containing PII or PHI was accessible from the compromised host, or if attacker-created administrator accounts are discovered — each condition independently triggers breach notification assessment obligations under applicable data protection regulations.

Recovery Notes	Before declaring recovery complete, verify that trx_addons version 2.38.5 is confirmed installed and that the specific file upload validation functions patched in this release are present in the deployed code — do not rely solely on the version number given this vulnerability's regression history from CVE-2024-13448. Monitor web server access logs for POST requests to wp-admin/admin-ajax.php with trx_addons action parameters for a minimum of 72 hours post-patch, as active exploitation confirmed by CISA means automated scanning tools may continue probing patched endpoints. If a web shell was confirmed deployed at any point, treat the entire WordPress hosting environment (database, credentials, adjacent virtual hosts) as compromised and conduct a full credential rotation and lateral movement assessment before resuming normal operations.
Forensic Artifacts	Web server access logs (Apache: /var/log/apache2/access.log; Nginx: /var/log/nginx/access.log) — filter for HTTP POST requests to /wp-admin/admin-ajax.php with action parameters containing 'trx_addons', particularly requests from external IPs receiving HTTP 200 responses, which indicate successful AJAX handler invocation and potential successful file upload. | File system scan of wp-content/uploads/ and wp-content/plugins/trx_addons/ for PHP files with extensions .php, .phtml, .php5, .phar — this exploit deposits web shells in the uploads directory (world-writable by design) or plugin subdirectories; use 'find /var/www/html/wp-content -name "*.php" -newer -ls' to surface post-exploitation artifacts. | WordPress database wp_users and wp_usermeta tables — attacker-created administrator accounts are a common post-exploitation persistence mechanism following web shell deployment via unauthenticated upload; export with 'wp user list --role=administrator --format=json' and cross-reference registration timestamps against the earliest confirmed exploit window. | PHP error log (/var/log/php_errors.log or php.ini error_log path) — failed upload attempts and successful but partially broken web shell executions generate stack traces referencing trx_addons upload handler functions, providing evidence of exploit attempts even when the web server access log POST entry alone is ambiguous. | WAF logs and ModSecurity audit log (/var/log/modsec_audit.log) — if WAF was active during exploitation attempts, the full POST body captured in WAF logs may contain the uploaded file content (web shell source), the multipart form-data boundary, and the original attacker filename, all of which constitute primary forensic evidence of the payload delivered via CVE-2026-1969.

Per-Action IR Details

Step 1: Containment — Immediately identify all WordPress instances running trx_addons (ThemeREX Addons) plugin versions prior to 2.38.5 across your environment. If patching cannot occur within hours, use your WAF to block unauthenticated POST requests to WordPress AJAX endpoints (wp-admin/admin-ajax.php) targeting trx_addons action handlers as an interim measure.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Run 'wp plugin list --name=trx_addons --fields=name,version --format=csv' via WP-CLI across all managed WordPress sites, or scan wp-content/plugins/ directories with 'find /var/www -type d -name trx_addons -exec grep -r "Version:" {}/trx_addons.php \;' to enumerate installed versions. For WAF-less environments, use ModSecurity with a custom rule blocking POST to wp-admin/admin-ajax.php where the request body contains 'action=trx_addons' or known upload action strings (e.g., 'trx_addons_uploads', 'trx_addons_save_options'). As a last resort, add 'deny from all' for admin-ajax.php in .htaccess scoped to external IPs while allowlisting your admin IP range.

Evidence: Before applying WAF rules or any network-layer block, capture a snapshot of current active connections to port 80/443 on affected WordPress hosts using 'ss -tnp' or 'netstat -antp'; preserve web server access logs (Apache:

/var/log/apache2/access.log; Nginx: /var/log/nginx/access.log) with timestamps intact. Export a directory listing with timestamps from wp-content/uploads/ and wp-content/plugins/trx_addons/ using 'find /var/www/html/wp-content -name "*.php" -newer /var/www/html/wp-content/plugins/trx_addons/trx_addons.php -ls' to establish a pre-containment file system baseline.

Step 2: Detection — Search web server access logs for POST requests to wp-admin/admin-ajax.php with trx_addons-related action parameters originating from external IPs, particularly those with non-standard file extensions in the request body. Scan the WordPress uploads directory and plugin directories for newly created PHP files, especially files with names inconsistent with media uploads. Check file system modification timestamps for any .php files created in wp-content/ after your last known-clean baseline.

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Query Apache/Nginx access logs with: 'grep -E "POST.*admin-ajax\.php" /var/log/apache2/access.log | grep -i "trx_addons"' to surface exploit attempts. For web shell discovery, run 'find /var/www/html/wp-content -name "*.php" -newer /var/www/html/wp-login.php -not -path "*/trx_addons/*" -ls' to flag PHP files created outside the plugin's own directory after the plugin install date. Use YARA with a rule targeting PHP web shell patterns (e.g., eval(base64_decode), system(), shell_exec()) against wp-content/uploads/ — the Laudanum or NeoPI YARA rulesets cover common web shell signatures. For network-based detection, deploy a Sigma rule matching HTTP 200 responses to POST requests at admin-ajax.php from external IPs where the response body size exceeds 0 bytes (indicating successful upload acknowledgment). MITRE ATT&CK T1190 (Exploit Public-Facing Application) and T1505.003 (Server Software Component: Web Shell) are the relevant techniques for detection tuning.

Evidence: Preserve raw web server access logs before any log rotation occurs — specifically look for HTTP 200 responses to POST /wp-admin/admin-ajax.php containing action parameters associated with trx_addons file handling. Capture PHP error logs (/var/log/php_errors.log or as configured in php.ini) for stack traces that may reveal the upload handler function invoked. Run 'find /var/www/html/wp-content/uploads -name "*.php" -o -name "*.phtml" -o -name "*.php5"' and hash all discovered files with 'md5sum' or 'sha256sum' for integrity evidence. Preserve WAF logs if present, retaining the full URI, POST body, source IP, user-agent, and response code for each request targeting admin-ajax.php.

Step 3: Eradication — Update the trx_addons plugin to version 2.38.5 or later via the WordPress admin panel or WP-CLI. If a web shell was discovered, remove the malicious file, rotate all WordPress admin credentials, database credentials, and hosting credentials. Audit all user accounts on the affected WordPress instance for unauthorized additions.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST IR-4 (Incident Handling), NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Apply the patch via WP-CLI: 'wp plugin update trx_addons --version=2.38.5' and verify with 'wp plugin get trx_addons --field=version'. Because CVE-2026-1969 is a regression from CVE-2024-13448, diff the upload-related functions in the new 2.38.5 plugin files against 2.38.4 using 'diff -rq wp-content/plugins/trx_addons/includes/ /includes/' to confirm the file upload validation code was actually reinstated. For credential rotation without a password manager, use 'wp user update --user_pass=' for all WordPress admin accounts and regenerate the WordPress secret keys in wp-config.php using the WordPress.org secret key generator (https://api.wordpress.org/secret-key/1.1/salt/). Audit WordPress user accounts with 'wp user list --role=administrator --fields=ID,user_login,user_email,user_registered' and flag any accounts registered after the earliest possible exploit date.

Evidence: Before removing any discovered web shell, preserve a forensic copy: capture the file with 'cp -p /secure/evidence/' retaining original timestamps, then hash it ('sha256sum'), and record the full directory listing showing creation time. Capture the WordPress database user table snapshot with 'wp user list --all --format=json > user_audit_pre_eradication.json' to document any attacker-created administrator accounts. Export wp-config.php

(read-only copy) to preserve database credentials that may need forensic correlation with database access logs. Review MySQL/MariaDB query logs or binary logs for any queries executed via the web shell (look for INSERT into wp_users or option updates to wp_siteurl) — binary log path is typically /var/lib/mysql/mysql-bin.* .

Step 4: Recovery — After patching, confirm the plugin version in the WordPress admin panel under Plugins. Re-scan the server file system for residual web shells using a file integrity monitoring tool or a WordPress security scanner (e.g., Wordfence, WPScan). Monitor web server logs and WAF alerts for continued exploitation attempts against the patched endpoint for at least 72 hours post-remediation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: Verify the patch with 'wp plugin get trx_addons --field=version' and cross-reference against the trx_addons changelog at wordpress.org/plugins/trx-addons/#developers to confirm 2.38.5 addresses the file upload validation regression. For file integrity validation without a commercial FIM tool, run 'find /var/www/html/wp-content -name "*.php" -newer /var/www/html/wp-content/plugins/trx_addons/trx_addons.php' daily for 72 hours and compare output against your pre-patch baseline hash list. Set up a lightweight cron-based monitor: 'crontab -e' with '*/15 * * * * grep -c "POST.*admin-ajax" /var/log/nginx/access.log >> /var/log/trx_monitor.log' to track continued POST volume against the endpoint. Use WPScan (free tier) with 'wpscan --url https://yoursite.com --enumerate p --plugins-detection aggressive' to confirm no residual vulnerable plugin state.

Evidence: Capture a post-patch plugin version confirmation screenshot or CLI output and store it as a timestamped remediation record per NIST IR-5 (Incident Monitoring) documentation requirements. Preserve all web server access logs from the 72-hour post-remediation monitoring window — any continued POST requests to admin-ajax.php with trx_addons action parameters after patching may indicate a second attacker or a persistence mechanism not yet eradicated. Generate a final file system scan report from the wp-content/ tree using 'find /var/www/html/wp-content -name "*.php" | xargs md5sum > post_remediation_hashes.txt' and store alongside the pre-eradication baseline for comparison.

Step 5: Post-Incident — Review your WordPress plugin update workflow: this vulnerability is a failed patch regression from CVE-2024-13448, indicating the prior fix was not validated. Implement mandatory regression testing or vendor patch verification before closing CVEs in plugins with prior file upload history. Evaluate whether unauthenticated AJAX endpoints across your WordPress estate require stricter WAF rules as a standing control.

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-2 (Flaw Remediation), 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: Conduct a lessons-learned session specifically examining the CVE-2024-13448 patch closure record — document why regression testing did not catch the re-introduction of the unauthenticated upload path in versions between 2.x and 2.38.4. Build a standing YARA rule and Sigma detection for unauthenticated PHP file upload attempts via WordPress AJAX (action parameters containing 'upload', 'save', 'import', or 'file') and deploy it as a permanent WAF or ModSecurity rule across all WordPress hosts. Add trx_addons to a monitored plugin watchlist using WPScan's API (free tier, 25 requests/day) with a scheduled weekly check: 'wpscan --url https://yoursite.com --api-token --enumerate vp'. For the AJAX endpoint audit, enumerate all registered unauthenticated AJAX actions with 'grep -r "add_action.*wp_ajax_nopriv" /var/www/html/wp-content/plugins/ --include="*.php" -l' to surface other potential attack surface across your WordPress estate.

Evidence: Retain the complete incident timeline, all preserved log files, web shell forensic copies, and credential rotation records in a closed incident package per NIST IR-5 (Incident Monitoring) and NIST AU-11 (Audit Record Retention) requirements — minimum 90-day retention recommended given active exploitation status. Document the

regression delta between CVE-2024-13448's patch and CVE-2026-1969's re-introduction, including the specific trx_addons PHP functions responsible for file type validation, as this evidence justifies workflow process changes and may be required for regulatory breach notification documentation if PII was accessible on the affected WordPress instance.

Detection Guidance

Primary detection target: unauthenticated POST requests to wp-admin/admin-ajax.php on WordPress servers running trx_addons. Look for multipart/form-data POST requests from unauthenticated sessions where the action parameter maps to trx_addons handlers. In web server access logs (Apache/nginx), filter for POST to /wp-admin/admin-ajax.php returning HTTP 200 from sessions without valid WordPress authentication cookies. In file system monitoring, alert on creation of .php files in wp-content/uploads/ or trx_addons plugin subdirectories, these directories should not contain executable PHP. If you have EDR on the web host, alert on PHP process spawning child processes (shell, curl, wget) as an indicator of web shell execution. Treat any newly created PHP file outside the plugin's expected structure as a high-confidence indicator pending verification.

Framework Mappings

MITRE-ATTACK

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

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
- **CM-2** — Baseline Configuration
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **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

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

Sources

Source	URL	Tier
vulncheck_kev	https://nvd.nist.gov/vuln/detail/CVE-2026-1969	T1
trx_addons Plugin Vulnerability (CVE-2026-1969) | Freshy	https://freshysites.com/security-bulletins/wordpress-security-bulle...	T3
ThemeREX Addons < 2.38.5 – Unauthenticated Arbitrary File Upload	https://wpscan.com/vulnerability/762530ae-80a5-4ff8-9725-6adab9498c33/	T3
CVE-2026-1969 - Vulnerability-Lookup	https://db.gcve.eu/vuln/cve-2026-1969	T3
CVE-2026-1969 - Info Vulnerability - TheHackerWire	https://www.thehackerwire.com/vulnerability/CVE-2026-1969/	T3
CISA KEV	https://www.cisa.gov/known-exploited-vulnerabilities-catalog	T1

Generated 2026-04-30 14:08 UTC by TJS Security Command Center