INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-15 13:50 UTC

CDN Key Theft Turns Three WordPress Plugins Into Backdoor Delivery Networks Across 1.2 Million Sites

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0465
Type	Threat Campaign
CVE ID	CVE-2026-10795
Severity	CRITICAL
CVSS Base Score	9.5
EPSS Score	0.0034 (25th percentile)
Affected Products	PushEngage, OptinMonster, TrustPulse (Awesome Motive); UpdraftPlus WordPress plugin; BunnyNet CDN, all versions serving CDN-hosted JS during June 12-14 UTC
Published	2026-06-15T05:59:38
Discovery Source	Rss

Executive Summary

Between June 12-14 UTC, attackers compromised CDN-hosted JavaScript files delivered by BunnyNet for three widely deployed WordPress plugins from Awesome Motive (PushEngage, OptinMonster, and TrustPulse), collectively installed on approximately 1.2 million sites. Any WordPress administrator who loaded the tampered script during that window had their session privileges silently hijacked to create hidden admin accounts and install server-side web shells, leaving those sites under attacker control. Organizations running affected plugins must treat impacted sites as fully compromised and initiate server-side forensic investigation immediately, as the payload was engineered to evade detection through native WordPress tooling.

Technical Analysis

CVE-2026-10795 (CVSS 9.5 Critical) describes a supply chain compromise in which an attacker obtained BunnyNet CDN credentials and tampered with JavaScript assets served for PushEngage, OptinMonster, and TrustPulse, all Awesome Motive products. The malicious JS executed in authenticated administrator browser sessions, exploiting valid session context (CWE-287: Improper Authentication) to issue privileged WordPress API calls. The payload created covert administrative accounts (T1098, T1078.001) and deployed server-side

web shells (T1505.003) engineered to avoid visibility in the WordPress admin dashboard (T1564, T1036.005). Initial access was achieved through CDN key theft rather than a plugin-layer vulnerability, classifying this as a supply chain compromise upstream of the plugin vendors (T1195.002, CWE-494: Download of Code Without Integrity Check). Secondary CWEs include CWE-693 (Protection Mechanism Failure) and CWE-506 (Embedded Malicious Code). The attack window is June 12-14 UTC; all plugin versions serving CDN-hosted JS during that period are affected. No vendor patch closes the initial vector; remediation is forensic, not update-based. EPSS score reflects low automated exploitation probability post-window, but sites compromised during the window retain active backdoors until manually remediated. CISA KEV listing not confirmed as of this writing.

Action Checklist

1. Step 1: Containment, Immediately identify all WordPress sites running PushEngage, OptinMonster, or TrustPulse that were active between June 12-14 UTC. Take those sites offline or block external access at the network perimeter until forensic review is complete. Do not rely on the WordPress dashboard to assess compromise status; the payload is designed to conceal itself from native admin tooling.
2. Step 2: Detection, Conduct server-side forensic review across filesystem, database, and web server logs. Query the WordPress database (wp_users table) for administrator-role accounts created between June 12-14 UTC that are not in your authorized account inventory (aligns with NIST AC-2 account management review and CIS 5.1 account inventory). Scan the filesystem for anomalous PHP files in wp-content/uploads, wp-content/themes, and plugin directories; use file integrity monitoring (e.g., AIDE, Tripwire, or native OS file auditing) and file type verification (magic byte analysis) to surface modifications. Review web server access logs for POST requests to unknown endpoints or atypical admin-context API calls during the exposure window. Check for outbound connections to unrecognized domains or IPs from the web server process (T1071.001, T1102).
3. Step 3: Eradication, Remove any unauthorized administrator accounts identified in the database. Delete web shell files discovered during filesystem review. Rotate all WordPress secret keys and salts (wp-config.php). Revoke and regenerate all API keys and credentials stored in the WordPress database or configuration files, including plugin-specific API tokens for PushEngage, OptinMonster, and TrustPulse. Rotate all privileged account credentials associated with affected sites. Confirm with BunnyNet that CDN-served assets for affected plugins have been re-validated and served from clean sources before re-enabling plugin functionality.
4. Step 4: Recovery, Before bringing sites back online, verify filesystem integrity against known-good backups predating June 12 UTC. Restore from pre-compromise backups where feasible. Re-enable only after confirming no residual web shells or unauthorized accounts remain. Implement file integrity monitoring and enable audit logging for all privileged account creation and plugin file changes (NIST AU-2, AU-12, CIS 8.2). Monitor web server and application logs continuously for 30 days post-remediation for signs of re-entry or C2 activity.
5. Step 5: Post-Incident, This attack exposed a dependency on CDN-served third-party JavaScript without integrity verification. Implement Subresource Integrity (SRI) checks for all externally hosted scripts across web properties (CWE-494 mitigation). Review third-party plugin and CDN vendor risk assessments; require vendors to document CDN key management and rotation procedures (NIST AC-2, AC-20). Enforce least-privilege for WordPress administrator accounts: no shared admin sessions, dedicated admin accounts per operator (NIST AC-6, CIS 5.4). Establish a software inventory process (CIS 2.1) to enable rapid identification of all sites running affected plugin versions during future supply chain events.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership, legal counsel, and potentially regulatory authorities if forensic review confirms exfiltration of PII, payment card data, or PHI from any of the 1.2 million potentially affected sites, if rogue admin accounts persist after initial eradication indicating active attacker re-entry, or if the organization lacks the internal capability to perform safe forensic review without further altering evidence — particularly given the payload's active concealment of compromise indicators from native WordPress tooling.
Recovery Notes	Restoration must use only backups with a verified timestamp predating June 12 UTC 00:00 — any backup taken during the June 12–14 window may contain already-installed web shells or rogue accounts and must be treated as potentially compromised. Post-restoration, monitor Nginx/Apache access logs daily for 30 days for POST requests to `wp-content/uploads/*.php`, `wp-admin/admin-ajax.php` with anomalous parameter patterns, and outbound connections from the web server process to non-whitelisted CDN or API endpoints, as the installed web shells may have included secondary persistence mechanisms not fully captured during eradication. Re-enable PushEngage, OptinMonster, and TrustPulse plugins only after Awesome Motive and BunnyNet have issued written confirmation that CDN-served JS assets have been re-signed and served from validated clean sources.
Forensic Artifacts	wp_users and wp_usermeta database tables: Rogue administrator accounts injected by the payload during June 12–14 UTC will appear here with `user_registered` timestamps in the exposure window and `wp_capabilities` meta values set to `administrator`; these are the primary persistence artifacts left by this campaign's privilege escalation mechanism. | Web server access logs (Nginx: /var/log/nginx/access.log; Apache: /var/log/apache2/access.log): POST requests to wp-admin/admin-ajax.php and to PHP files in wp-content/uploads/ during June 12–14 UTC will record the browser-side execution of the tampered BunnyNet JS payload performing admin session hijack and web shell upload, including source IPs and user-agent strings. | PHP files in wp-content/uploads/, wp-content/themes/, and plugin subdirectories: The payload's web shell installation stage drops executable PHP files in world-writable directories; inode change times (ctime) on files in these paths timestamped June 12–14 UTC that contain eval(), base64_decode(), system(), or passthru() function calls are direct indicators of attacker-installed backdoors from this specific campaign. | wp-config.php and wp_options table entries for plugin API tokens: The PushEngage, OptinMonster, and TrustPulse API credentials stored in wp-config.php or in the wp_options table (option_names matching '%pushengage%', '%optinmonster%', '%trustpulse%') represent high-value credential artifacts that the attacker could have exfiltrated during the session hijack window for downstream API abuse. | BunnyNet CDN pull zone access logs (vendor-side): If obtainable from BunnyNet via support request, access logs for the pull zones serving PushEngage, OptinMonster, and TrustPulse JS assets during June 12–14 UTC will show the scope of tampered-asset delivery, including volume of requests served from the compromised CDN origin, which correlates directly to the number of WordPress admin sessions exposed to the hijack payload.

Per-Action IR Details

Step 1: Containment — Immediately identify all WordPress sites running PushEngage, OptinMonster, or TrustPulse that were active between June 12–14 UTC. Take those sites offline or block external access at the network perimeter until forensic review is complete. Do not rely on the WordPress dashboard to assess

compromise status — the payload is designed to conceal itself from native admin tooling.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory)

Compensating: Query WordPress database directly via WP-CLI or direct MySQL access to identify active plugin installations: `wp plugin list --status=active --fields=name,version --path=/var/www/html` across all vhosts. Block external HTTP/S access at the perimeter using iptables or ufw rules (`ufw deny in on eth0 to any port 80,443`) rather than trusting WP-Admin, since the BunnyNet-delivered payload specifically suppresses admin-panel visibility of rogue accounts and web shells. Use a bash loop over `/etc/nginx/sites-enabled/` or Apache vhost configs to enumerate all WordPress installations before blocking.

Evidence: Before taking any site offline or blocking perimeter access, capture: (1) active PHP processes and parent relationships via `ps auxf | grep php` and `lsof -p ` to identify any live web shell sessions; (2) active outbound connections from the web server process via `ss -tulpn` and `netstat -antp | grep -E 'php|apache|nginx'` to identify live C2 channels established by the BunnyNet-delivered backdoor; (3) `last` and `lastb` output plus `/var/log/auth.log` for SSH sessions that may have been initiated post-compromise; (4) running cron jobs (`crontab -l -u www-data`) that the payload may have installed for persistence. These volatile artifacts are lost the moment the host is isolated or the web server process is terminated.

Step 2: Detection — Conduct server-side forensic review across filesystem, database, and web server logs. Query the WordPress database (wp_users table) for administrator-role accounts created between June 12–14 UTC that are not in your authorized account inventory (aligns with NIST AC-2 account management review and CIS 5.1 account inventory). Scan the filesystem for anomalous PHP files in wp-content/uploads, wp-content/themes, and plugin directories; use file integrity monitoring or D3-SFA (System File Analysis) to surface modifications. Review web server access logs for POST requests to unknown endpoints or atypical admin-context API calls during the exposure window. Check for outbound connections to unrecognized domains or IPs from the web server process (T1071.001, T1102).

NIST Phase: Detection Analysis

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

Controls: NIST AC-2 (Account Management), NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 8.2 (Collect Audit Logs)

Compensating: Run the following MySQL query directly to surface rogue admin accounts injected during the compromise window: `SELECT user_login, user_registered, user_email FROM wp_users u JOIN wp_usermeta m ON u.ID=m.user_id WHERE m.meta_key='wp_capabilities' AND m.meta_value LIKE '%administrator%' AND u.user_registered BETWEEN '2026-06-12 00:00:00' AND '2026-06-14 23:59:59';`. For filesystem web shell detection without a commercial tool, run `find /var/www -name '*.php' -newer /var/www/html/wp-config.php -mtime -30 -exec grep -l 'eval\|base64_decode\|system\|passthru\|shell_exec' {} \;` to identify PHP files modified after plugin installation that contain execution primitives characteristic of web shells dropped by this campaign. Parse Nginx/Apache access logs with `awk` filtering for POST requests to wp-admin/admin-ajax.php and wp-content/uploads/*.php between June 12–14 UTC.

Evidence: Before running any filesystem scan or database query that modifies access timestamps, preserve: (1) web server access logs in their current state — copy `/var/log/nginx/access.log*` and `/var/log/apache2/access.log*` to write-protected storage, as these will show the POST requests from the tampered BunnyNet JS executing admin-context privilege escalation and web shell upload; (2) the inode change times (`stat`) on all PHP files in `wp-content/uploads`, `wp-content/themes`, and plugin subdirectories, which record when the attacker dropped web shells even if mtime was manipulated; (3) a full dump of the `wp_users` and `wp_usermeta` tables before any remediation alters account records; (4) `/proc//net/tcp` for current socket state if the host is still live. The `find` command with `-newer` alters atime on traversed directories on some filesystem configurations — use `noatime` mount options or capture inode metadata first.

Step 3: Eradication — Remove any unauthorized administrator accounts identified in the database. Delete web shell files discovered during filesystem review. Rotate all WordPress secret keys and salts (wp-config.php). Revoke and regenerate all API keys and credentials stored in the WordPress database or configuration files, including plugin-specific API tokens for PushEngage, OptinMonster, and TrustPulse. Apply D3-CRO (Credential Rotation) across all privileged accounts associated with affected sites. Confirm with BunnyNet that CDN-served assets for affected plugins have been re-validated and served from clean sources before re-enabling plugin functionality.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), NIST AC-12 (Session Termination), CIS 5.3 (Disable Dormant Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Delete rogue admin accounts via WP-CLI: `wp user delete --reassign=1 --path=/var/www/html`. Rotate WordPress secret keys by fetching fresh values from the WordPress.org secret key API (`curl https://api.wordpress.org/secret-key/1.1/salt/`) and replacing the eight `define('AUTH_KEY'...` lines in `wp-config.php` — this immediately invalidates all active WordPress session cookies, including any the attacker's rogue accounts hold. For PushEngage and OptinMonster API key rotation, log into each vendor's dashboard directly (not via the compromised WordPress admin) and regenerate API tokens; update `wp_options` table entries (`SELECT option_name, option_value FROM wp_options WHERE option_name LIKE '%pushengage%' OR option_name LIKE '%optinmonster%' OR option_name LIKE '%trustpulse%'`) to confirm old tokens are replaced. For BunnyNet CDN validation, contact BunnyNet support to confirm the pull zone serving the affected plugin JS has been flushed and re-served from clean origin.

Evidence: Before deleting any account or rotating any credential, preserve: (1) a complete database dump (`mysqldump wordpress > pre_eradication_$(date +%Y%m%d).sql`) capturing rogue account details, their `wp_usermeta` capability assignments, and any option values modified by the payload for post-incident analysis; (2) full copies of all identified web shell files with their inode metadata and SHA-256 hashes — these are evidence of the attacker's TTPs and may be needed for law enforcement or insurance; (3) `wp-config.php` current state before key rotation, as the existing keys confirm which sessions were valid during the compromise window; (4) all active PHP session files in the session storage path (typically `/var/lib/php/sessions/`) which may contain session tokens the attacker used post-privilege-escalation. Credential rotation invalidates live sessions — capture session state first.

Step 4: Recovery — Before bringing sites back online, verify filesystem integrity against known-good backups predating June 12 UTC. Restore from pre-compromise backups where feasible. Re-enable only after confirming no residual web shells or unauthorized accounts remain. Implement file integrity monitoring (D3-SFA) and enable audit logging for all privileged account creation and plugin file changes (NIST AU-2, AU-12, CIS 8.2). Monitor web server and application logs continuously for 30 days post-remediation for signs of re-entry or C2 activity.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-2 (Event Logging), NIST AU-12 (Audit Record Generation), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Verify pre-June-12 backup integrity by computing SHA-256 hashes of all PHP files in the restored backup and comparing against the current filesystem: `find /var/www/html -name '*.php' -exec sha256sum {} \; > current_hashes.txt` then diff against a hash manifest generated from the backup. Deploy OSSEC or Wazuh (free, open-source HIDS) to monitor `wp-content/uploads`, `wp-content/themes`, and plugin directories for new PHP file creation in real time. Enable WordPress action hooks for `user_register` and log all new admin-role account creations to a write-protected external syslog target so a recurrence of the rogue account injection pattern is immediately detectable. For 30-day post-recovery monitoring, configure a daily cron to re-run the rogue account and web shell detection queries and email results to the security team.

Evidence: Before restoring from backup and before bringing the site back online, verify the backup itself predates June 12 UTC by checking backup metadata and file timestamps — a backup taken during the June 12–14 window may itself contain already-compromised files. Document the SHA-256 hash of the restored `wp-config.php`, all plugin PHP files for PushEngage, OptinMonster, and TrustPulse, and the `wp_users` table row count as a post-recovery baseline for future integrity comparison. Retain all pre-eradication forensic copies (database dumps, web shell files, log archives) in offline write-protected storage for a minimum of 90 days to support post-incident review and any downstream regulatory notification requirements.

Step 5: Post-Incident — This attack exposed a dependency on CDN-served third-party JavaScript without integrity verification. Implement Subresource Integrity (SRI) checks for all externally hosted scripts across web properties (CWE-494 mitigation). Review third-party plugin and CDN vendor risk assessments; require vendors to document CDN key management and rotation procedures (NIST AC-2, AC-20). Enforce least-privilege for WordPress administrator accounts — no shared admin sessions, dedicated admin accounts per operator (NIST AC-6, CIS 5.4). Establish a software inventory process (CIS 2.1) to enable rapid identification of all sites running affected plugin versions during future supply chain events.

NIST Phase: Post Incident

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

Controls: NIST AC-6 (Least Privilege), NIST AC-20 (Use Of External Systems), NIST AU-11 (Audit Record Retention), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts)

Compensating: Implement SRI for BunnyNet-served plugin assets by extracting the SHA-384 hash of each known-good JS file (`openssl dgst -sha384 -binary pushengage.min.js | openssl base64 -A`) and adding `integrity` and `crossorigin` attributes to the script tags injected by PushEngage, OptinMonster, and TrustPulse — browsers will refuse to execute the script if the CDN-served content hash does not match. For software inventory without a commercial tool, build a WordPress installation manifest using `find /var/www -name 'wp-config.php' -exec dirname {} \;` piped to `wp plugin list` per vhost, stored in a version-controlled CSV that includes plugin name, version, and install path — enabling sub-hour blast-radius assessment the next time a supply-chain event targets an Awesome Motive or BunnyNet-dependent plugin. Conduct a formal lessons-learned session within 5 business days documenting the detection gap (no SRI, no CDN asset integrity alerting) and assign owners to SRI implementation and vendor risk questionnaire updates for BunnyNet and Awesome Motive.

Evidence: For the post-incident record, preserve and archive: (1) the full timeline of the June 12–14 compromise window correlated across web server logs, database account creation timestamps, and BunnyNet CDN pull logs if obtainable from the vendor; (2) the final list of rogue admin accounts and web shell file paths and hashes as the definitive indicator set for this campaign; (3) all vendor communications with BunnyNet and Awesome Motive regarding CDN key compromise and asset re-validation, which establish the third-party risk chain for regulatory or cyber-insurance purposes; (4) the pre- and post-remediation `wp_users` table states as evidence of account lifecycle for any breach notification obligation assessment. No volatile capture requirement applies to this phase — all evidence at this stage is non-volatile and should be retained per NIST AU-11 (Audit Record Retention) organizational policy.

Detection Guidance

Detection must occur at the server layer; the WordPress admin dashboard cannot be trusted on potentially compromised sites. Key indicators: (1) WordPress database - query wp_users and wp_usermeta for admin-role accounts (meta_key = 'wp_capabilities' containing 'administrator') created between 2026-06-12T00:00Z and 2026-06-14T23:59Z that do not appear in your authorized account inventory. (2) Filesystem - scan for PHP files modified or created in wp-content/uploads, wp-content/themes, and wp-content/plugins between June 12-14 UTC; uploads directories should contain no executable PHP. Use file integrity monitoring and file type verification (magic byte analysis) to identify files misrepresenting their type. (3) Web server access logs - look for POST requests to non-standard wp-admin endpoints or direct PHP file requests in uploads directories,

particularly from IPs not associated with legitimate administrator sessions. (4) Outbound network - identify web server process connections to external IPs or domains not in your allowlist, consistent with T1071.001 (Application Layer Protocol) and T1102 (Web Service) C2 patterns. (5) Authentication logs - flag any privileged account creation events (T1098) not correlated with a change management ticket during the June 12-14 UTC window. Correlate findings against NIST AU-6 review cadence. Monitor threat intelligence feeds (CISA, Tenable, vendor advisories) for IOC updates. Treat all anomalies matching the detection window as high-confidence indicators pending further attribution.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`BunnyNet CDN-served JS assets for PushEngage, OptinMonster, TrustPulse — June 12–14 UTC`	CDN-hosted JavaScript files for three Awesome Motive plugins were tampered with during this window; specific tampered asset URLs not confirmed in available source material	HIGH

Framework Mappings

MITRE-ATTACK

- **T1078.003** — Local Accounts
- **T1059.007** — JavaScript
- **T1195.002** — Compromise Software Supply Chain
- **T1556** — Modify Authentication Process
- **T1071.001** — Web Protocols
- **T1102** — Web Service
- **T1505.003** — Web Shell
- **T1564** — Hide Artifacts
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1098** — Account Manipulation
- **T1078.001** — Default Accounts

NIST-800-53R5

- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SI-4** — System Monitoring

- **CM-2** — Baseline Configuration
- **SI-3** — Malicious Code Protection
- **IA-8** — Identification and Authentication (Non-Organizational Users)
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A07:2021** — Identification and Authentication Failures
- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications
- **6.4** — Require MFA for Remote Network Access
- **6.5** — Require MFA for Administrative Access
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **15.1** — Establish and Maintain an Inventory of Service Providers
- **8.2** — Collect Audit Logs

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program
- **DE.CM-01** — Networks and network services are monitored

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1078.003	Local Accounts	Defense-Evasion
T1059.007	JavaScript	Execution
T1195.002	Compromise Software Supply Chain	Initial-Access
T1556	Modify Authentication Process	Credential-Access

Technique ID	Technique Name	Tactic
T1071.001	Web Protocols	Command-And-Control
T1102	Web Service	Command-And-Control
T1505.003	Web Shell	Persistence
T1564	Hide Artifacts	Defense-Evasion
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1098	Account Manipulation	Persistence
T1078.001	Default Accounts	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/06/popular-wordpress-plugin-scripts....	T3
CVE-2026-10795 Detail - NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-10795	T1
CVE-2026-10795 | Tenable®	https://www.tenable.com/cve/CVE-2026-10795	T3
CVE-2026-30795: RustDesk Client Information Disclosure Bug	https://www.sentinelone.com/vulnerability-database/cve-2026-30795/	T3
CVE-2026-46595 - Red Hat Customer Portal	https://access.redhat.com/security/cve/CVE-2026-46595	T3

Generated 2026-06-15 13:50 UTC by TJS Security Command Center