INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-16 07:13 UTC

Awesome Motive CDN Supply Chain Attack Backdoors Up to 1.2 Million WordPress Sites via UpdraftPlus Initial Access

THREAT CAMPAIGN | **CRITICAL** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0475
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	7.5
Affected Products	OptinMonster, TrustPulse, PushEngage (Awesome Motive plugins); UpdraftPlus WordPress plugin (initial access vector); WordPress sites running any of these plugins
Published	2026-06-15T13:37:07
Discovery Source	Rss

Executive Summary

Threat actors compromised Awesome Motive's content delivery network to inject malicious code into files served by three widely deployed WordPress plugins, OptinMonster, TrustPulse, and PushEngage, potentially affecting up to 1.2 million WordPress sites. The injected scripts silently stole administrator authentication tokens, created unauthorized admin accounts, and installed self-concealing backdoor plugins with full remote code execution capability. CDN-side files have been cleaned, but any site that loaded tampered scripts during the exposure window must be treated as fully compromised; plugin updates alone do not remove site-resident backdoors or rogue accounts.

Technical Analysis

Attackers exploited a vulnerability in the UpdraftPlus WordPress plugin to gain initial access to a peripheral Awesome Motive marketing server. From that foothold, they exfiltrated CDN API credentials and tampered with JavaScript files distributed via Awesome Motive's CDN to users of OptinMonster, TrustPulse, and PushEngage. The malicious payload (MITRE T1195.002, Compromise Software Supply Chain; T1059.007, JavaScript execution) performed session token harvesting (T1539), rogue administrator account creation (T1136), and deployment of hidden backdoor plugins implementing web shells (T1505.003) and C2 communication over HTTP (T1071.001). Persistence mechanisms included file-hidden backdoors (T1564.001) and reuse of valid accounts (T1078.001, T1078.003). Initial access is mapped to T1190 (Exploit Public-Facing Application) via UpdraftPlus. Masquerading techniques (T1036.005) were used to conceal backdoor plugin identity. Relevant

weaknesses: CWE-798 (hardcoded/exfiltrated credentials), CWE-693 (protection mechanism failure), CWE-494 (download of code without integrity check), CWE-522 (insufficiently protected credentials). No CVE ID is assigned as of reporting date. The CDN payload has been remediated upstream, but site-resident implants, backdoor plugins and rogue admin accounts, persist independently of CDN or plugin updates and require per-site forensic investigation.

Action Checklist

1. Step 1: Containment. Immediately audit all WordPress administrator accounts on sites running OptinMonster, TrustPulse, or PushEngage. Suspend any accounts not recognized by your team. Disable the affected plugins and block CDN asset domains at the perimeter until site integrity is confirmed. Apply NIST AC-2 (Account Management) procedures: revoke unauthorized accounts without waiting for forensic confirmation.
2. Step 2: Detection. Query WordPress user tables and audit logs for accounts created during the exposure window that lack a corresponding provisioning record. Search server file systems for unknown plugins in /wp-content/plugins/ with obfuscated code or unusual file modification timestamps. Review web server access logs for POST requests to unfamiliar PHP files, unexpected admin-panel access from new IP addresses, and outbound connections to unrecognized hosts (MITRE T1071.001 C2 pattern). Apply CIS 8.2 (Collect Audit Logs): confirm logging was enabled across the exposure window; gaps in log coverage should be treated as a detection failure, not a clean signal. Check for JavaScript integrity mismatches on CDN-served assets loaded during the exposure window.
3. Step 3: Eradication. Sites confirmed or suspected to have loaded tampered CDN scripts must undergo full forensic investigation before returning to production. Remove all unrecognized plugins, including any with legitimate-appearing names that were installed during or after the exposure window (T1036.005 masquerading). Delete all rogue administrator accounts. Rotate all WordPress admin credentials, database passwords, and any API keys stored on the affected server (NIST IA controls; D3-CRO: Credential Rotation). Enforce D3-CH (Credential Hardening) for all remaining admin accounts. Update UpdraftPlus to its latest patched release as the confirmed initial access vector.
4. Step 4: Recovery. After eradication, restore WordPress from a known-clean backup predating the exposure window, or perform a clean reinstall of WordPress core, themes, and plugins from verified sources to eliminate any file-resident implants. Validate file integrity against known-good hashes (D3-SFA: System File Analysis; D3-FMBV: File Magic Byte Verification). Re-enable plugins only from verified, unmodified sources. Monitor for re-emergence of rogue accounts or unexpected outbound traffic for at least 30 days post-remediation. Enforce CIS 6.5 (Require MFA for Administrative Access) on all recovered admin accounts before returning sites to production.
5. Step 5: Post-Incident. Conduct a supply chain dependency audit across all WordPress installations to inventory third-party plugin CDN delivery paths (CIS 2.1: Establish and Maintain a Software Inventory). Implement Subresource Integrity (SRI) checks or equivalent integrity verification for all externally hosted scripts to detect future CDN tampering (CWE-494 remediation; CIS 4.6: Securely Manage Enterprise Assets and Software). Review plugin vetting and update approval processes. Establish baseline monitoring for WordPress admin account creation events as a standing detection rule (NIST AU-2: Event Logging; AU-6: Audit Record Review). Map UpdraftPlus access controls against NIST AC-6 (Least Privilege) to determine whether the initial access path could have been restricted.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to legal counsel and initiate breach notification assessment immediately if forensic evidence confirms that administrator authentication tokens were exfiltrated from sites processing PII, payment card data, or PHI — the silent token theft mechanism and rogue account creation constitute indicators of unauthorized access triggering notification obligations under GDPR, CCPA, and PCI DSS, or if the number of confirmed compromised sites exceeds your organization's defined critical-asset threshold or a single compromised site hosts customer-facing authentication.
Recovery Notes	Restoration must use only backups with timestamps predating the confirmed exposure window start, as UpdraftPlus — the confirmed initial access vector — may have included tampered plugin files in any backup taken during the exposure period. Validate restored installations using `wp core verify-checksums` and manual SHA-256 comparison of all Awesome Motive plugin files against freshly downloaded originals from wordpress.org before returning any site to production. Monitor all recovered sites for at least 30 days post-restoration for re-emergence of unexpected administrator accounts, unauthorized plugin installations, or outbound connections to non-wordpress.org CDN domains, as self-concealing backdoor plugins may have registered persistence mechanisms not fully removed by plugin deletion alone.
Forensic Artifacts	wp_users and wp_usermeta MySQL table dumps covering the exposure window: rogue accounts created by the injected scripts will have anomalous registration timestamps, non-organizational email domains, and usermeta entries storing tokens or role-escalation markers inconsistent with normal WordPress provisioning | Web server access logs (Apache access.log / Nginx access.log) for the exposure window: look for POST requests to PHP files within /wp-content/plugins/ directories that do not correspond to any known plugin endpoint — these are the backdoor interaction requests made by the threat actor after the injected script created the initial foothold | File system metadata for /wp-content/plugins/ directories: mtime and ctime values on PHP files created or modified during the exposure window identify backdoor plugin installations; files with legitimate plugin names but modification times during the attack window are the primary masquerading indicators | UpdraftPlus configuration data from the wp_options table (keys prefixed 'updraftplus_'): these entries contain remote backup destination credentials (AWS S3 access keys, Google Drive OAuth tokens, FTP credentials) that were accessible via the UpdraftPlus initial access path and represent credential exposure beyond the WordPress admin context | PHP session files from the server session directory (/var/lib/php/sessions/ or equivalent) and any application-level session tokens stored in wp_options or a custom sessions table: the injected CDN scripts specifically targeted administrator authentication tokens, and unexpired session artifacts in server-side storage may preserve evidence of token theft events with precise timestamps

Per-Action IR Details

Step 1: Containment — Immediately audit all WordPress administrator accounts on sites running OptinMonster, TrustPulse, or PushEngage. Suspend any accounts not recognized by your team. Disable the affected plugins and block CDN asset domains at the perimeter until site integrity is confirmed. Apply NIST AC-2 (Account Management) procedures: revoke unauthorized accounts without waiting for forensic confirmation.

NIST Phase: Containment

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

Controls: NIST AC-2 (Account Management), NIST AC-12 (Session Termination), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Run `wp user list --role=administrator --format=csv` via WP-CLI to export all current admin accounts; diff the output against your provisioning records or last-known-good user export. To block CDN domains, add firewall rules using iptables (`iptables -A OUTPUT -d -j DROP`) or update /etc/hosts to null-route the offending Awesome Motive CDN FQDNs. Disable plugins directly in the database if wp-admin is untrusted: `UPDATE wp_options SET option_value='a:0:{}' WHERE option_name='active_plugins';`

Evidence: Before suspending any accounts or blocking CDN domains, capture: (1) a full dump of wp_users and wp_usermeta tables (`mysqldump --single-transaction dbname wp_users wp_usermeta > wp_users_snapshot.sql`) to preserve the rogue account's creation timestamp, email, and any stored token metadata injected by the backdoor scripts; (2) active PHP session files from the server's session storage directory (typically /var/lib/php/sessions/) to preserve any stolen admin authentication tokens that have not yet been transmitted; (3) current outbound network connections (`ss -tunap` or `netstat -ano`) to capture any live C2 channels established by the backdoor before CDN blocking severs them. Volatile session state and active connections are destroyed the moment you revoke accounts or block the CDN.

Step 2: Detection — Query WordPress user tables and audit logs for accounts created during the exposure window that lack a corresponding provisioning record. Search server file systems for unknown plugins in /wp-content/plugins/ with obfuscated code or unusual file modification timestamps. Review web server access logs for POST requests to unfamiliar PHP files, unexpected admin-panel access from new IP addresses, and outbound connections to unrecognized hosts (MITRE T1071.001 C2 pattern). Apply CIS 8.2 (Collect Audit Logs): confirm logging was enabled across the exposure window; gaps in log coverage should be treated as a detection failure, not a clean signal. Check for JavaScript integrity mismatches on CDN-served assets loaded during the exposure window.

NIST Phase: Detection Analysis

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

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Query rogue account creation with: `SELECT user_login, user_email, user_registered FROM wp_users WHERE user_registered BETWEEN '' AND '';` — cross-reference against known provisioning records. Scan /wp-content/plugins/ for obfuscated code using: `grep -rn --include='*.php' -E '(base64_decode|eval|str_rot13|gzinflate)' /var/www/html/wp-content/plugins/` and flag any hits in plugin directories modified during the exposure window (`find /wp-content/plugins/ -newer /var/www/html/wp-config.php -name '*.php'`). For CDN JavaScript integrity, retrieve the SHA-256 hash of each cached/locally-served OptinMonster, TrustPulse, and PushEngage JS asset and compare against Awesome Motive's published clean-version hashes. Parse Apache/Nginx access logs with: `grep -E 'POST .*(wp-admin|\.php)' /var/log/apache2/access.log | grep -v '` to surface unauthorized wp-admin POSTs.

Evidence: This step is observational and does not alter live state, but preserve evidence before any subsequent action: (1) web server access logs (Apache: /var/log/apache2/access.log; Nginx: /var/log/nginx/access.log) covering the full exposure window — these will contain the specific POST requests used to interact with backdoor PHP files installed by the injected scripts; (2) file system metadata snapshot via `find /wp-content/plugins/ -printf '%T+ %p ' | sort > plugin_mtimes.txt` — modification timestamps on plugin PHP files are key forensic markers of backdoor installation timing; (3) DNS resolution logs or outbound proxy logs capturing connections from the WordPress server to Awesome Motive CDN domains during the exposure window, which will confirm whether tampered scripts were actually loaded (exposure ≠ compromise — loading the tampered asset is the trigger); (4) wp-cron job definitions (`wp cron event list`) and server crontabs (`crontab -l; cat /etc/cron*`) to identify any persistence mechanisms the backdoor may have registered.

Step 3: Eradication — Sites confirmed or suspected to have loaded tampered CDN scripts must undergo full forensic investigation before returning to production. Remove all unrecognized plugins, including any with legitimate-appearing names that were installed during or after the exposure window (T1036.005

masquerading). Delete all rogue administrator accounts. Rotate all WordPress admin credentials, database passwords, and any API keys stored on the affected server (NIST IA controls; D3-CRO: Credential Rotation). Enforce D3-CH (Credential Hardening) for all remaining admin accounts. Update UpdraftPlus to its latest patched release as the confirmed initial access vector.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), CIS 5.2 (Use Unique Passwords), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Delete rogue accounts with: `wp user delete --reassign=`. Rotate the WordPress database password in wp-config.php and update the MySQL user simultaneously: `ALTER USER 'wp_user'@'localhost' IDENTIFIED BY '';`. Scan for masquerading plugins with: `wp plugin list --format=csv` and manually verify each plugin's directory name matches its declared plugin header (`grep -r 'Plugin Name:' /wp-content/plugins/*/` — a mismatch between directory name and Plugin Name header is a masquerading indicator). Update UpdraftPlus via: `wp plugin update updraftplus`. Rotate any UpdraftPlus-stored cloud backup credentials (S3 keys, Google Drive tokens) from the UpdraftPlus settings, as the initial access through UpdraftPlus may have exposed those stored credentials to the threat actor.

Evidence: Before deleting rogue accounts, rotating credentials, or removing backdoor plugins, capture: (1) a forensic copy of each suspicious plugin directory (`tar -czf backdoor_plugins_evidence.tar.gz /wp-content/plugins//`) — PHP files in these directories are the primary payload artifacts and will be destroyed by removal; (2) wp_usermeta entries for all rogue accounts (`SELECT * FROM wp_usermeta WHERE user_id IN ();`) — metadata may contain injected tokens, stolen session hashes, or attacker-controlled email addresses used for persistence; (3) any UpdraftPlus backup logs and stored remote credential configurations (typically in wp_options under keys prefixed 'updraftplus_') — these reveal what data the threat actor may have accessed via the initial access vector and what credentials require rotation beyond WordPress itself; (4) current running PHP processes (`ps aux | grep php`) and loaded Apache/PHP modules to detect in-memory backdoor execution before process state is cleared by a service restart during credential rotation.

Step 4: Recovery — After eradication, restore WordPress from a known-clean backup predating the exposure window, or perform a clean reinstall of WordPress core, themes, and plugins from verified sources to eliminate any file-resident implants. Validate file integrity against known-good hashes (D3-SFA: System File Analysis; D3-FMBV: File Magic Byte Verification). Re-enable plugins only from verified, unmodified sources. Monitor for re-emergence of rogue accounts or unexpected outbound traffic for at least 30 days post-remediation. Enforce CIS 6.5 (Require MFA for Administrative Access) on all recovered admin accounts before returning sites to production.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AC-2 (Account Management), CIS 6.5 (Require MFA for Administrative Access), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: Verify WordPress core file integrity against official checksums using: `wp core verify-checksums` — any modified core files not explained by legitimate customization indicate residual implants. For plugins, download fresh copies from wordpress.org and compare SHA-256 hashes: `sha256sum /wp-content/plugins//*.php` against hashes of the freshly downloaded archive. For 30-day re-emergence monitoring without a SIEM, implement a daily cron job: `wp user list --role=administrator --format=csv >> /var/log/wp_admin_audit.log` and diff each day's output against the post-recovery baseline. Monitor outbound connections using netstat in a cron: `ss -tunap | grep ESTABLISHED >> /var/log/outbound_connections.log` and alert on any new external IPs beyond known WordPress update servers.

Evidence: Before restoring from backup or reinstalling WordPress core, confirm: (1) the backup predates the exposure window — verify the UpdraftPlus backup timestamp against the confirmed campaign start date, since UpdraftPlus was the initial access vector and any backup it created during or after compromise may itself contain tampered files; (2) capture a final file system hash manifest of the compromised installation (`find /var/www/html -type f -name '*.php' -exec md5sum {} \; > compromised_fs_hashes.txt`) as a forensic baseline to compare against the restored installation and confirm full eradication; (3) preserve the compromised database dump in its entirety before restoration, as the

wp_options table may contain serialized backdoor configurations or scheduled tasks registered by the injected scripts that would be overwritten by a clean restore.

Step 5: Post-Incident — Conduct a supply chain dependency audit across all WordPress installations to inventory third-party plugin CDN delivery paths (CIS 2.1: Establish and Maintain a Software Inventory). Implement Subresource Integrity (SRI) checks or equivalent integrity verification for all externally hosted scripts to detect future CDN tampering (CWE-494 remediation; CIS 4.6: Securely Manage Enterprise Assets and Software). Review plugin vetting and update approval processes. Establish baseline monitoring for WordPress admin account creation events as a standing detection rule (NIST AU-2: Event Logging; AU-6: Audit Record Review). Map UpdraftPlus access controls against NIST AC-6 (Least Privilege) to determine whether the initial access path could have been restricted.

NIST Phase: Post Incident

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

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AC-6 (Least Privilege), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Build a plugin CDN inventory by auditing each active plugin's source code for external script enqueue calls: `grep -rn 'wp_enqueue_script' /wp-content/plugins/ | grep 'http'` — document every external CDN domain loading assets into your WordPress front-end or admin panel. For SRI enforcement, generate integrity hashes for known-good versions of OptinMonster, TrustPulse, and PushEngage JS assets using: `openssl dgst -sha384 -binary | openssl base64 -A` and add the resulting hash to script tags or a CSP header (`Content-Security-Policy: require-sri-for script`). For standing admin-account monitoring, deploy a Sigma rule matching WordPress application log entries for new user_register events with role=administrator, triaged daily by a 2-person team using a simple grep-to-alert cron job.

Evidence: Post-incident evidence collection for lessons learned: (1) the full timeline reconstructed from web server access logs correlating first load of tampered CDN assets (OptinMonster/TrustPulse/PushEngage JS) to first appearance of rogue admin accounts in wp_users — this establishes the precise dwell time between initial load and account creation; (2) UpdraftPlus access and configuration logs to document the exact initial access path and determine whether least-privilege configuration of UpdraftPlus remote storage credentials could have limited the threat actor's lateral reach; (3) a diff of the pre- and post-incident plugin CDN asset hashes to confirm which specific Awesome Motive CDN-served file versions contained the injected credential-harvesting and backdoor-installation scripts, supporting threat intelligence sharing and future detection rule development.

Detection Guidance

Primary indicators: (1) WordPress user table entries for administrator-role accounts with creation timestamps falling within the exposure window and no corresponding provisioning record in your change management or user onboarding logs. (2) Presence of unfamiliar or obfuscated plugin directories under /wp-content/plugins/, particularly those with recently modified file timestamps coinciding with the exposure window. (3) Web server access logs showing POST requests to PHP files not associated with installed plugins, or GET requests to wp-admin from IP addresses with no prior authentication history. (4) Outbound HTTP/HTTPS connections from the web server to domains not in your approved CDN or vendor list, consistent with T1071.001 C2 beaconing. (5) JavaScript file hash mismatches on CDN-served OptinMonster, TrustPulse, or PushEngage assets when compared against vendor-published clean versions. (6) WordPress debug or error logs showing unexpected function calls or eval() execution in plugin context. Behavioral indicators: admin session tokens appearing in requests from IPs that never completed a standard login flow, and plugin activation events with no corresponding administrator action in audit logs. Per NIST AU-6, review audit records at increased frequency for the 30-day post-remediation period. CIS 8.2 compliance is a prerequisite; sites without continuous audit log collection cannot reliably determine exposure status.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://bleepingcomputer.com/news/security/optinmonster-wordpress-plugin-hacked-in-cdn-supply-chain-attack/`	BleepingComputer reporting — source article, not a threat IOC	HIGH
URL	`https://patchstack.com/articles/supply-chain-attack-on-optinmonster-trustpulse-and-pushengage-tampered-cdn-scripts-auto-creating-rogue-admins/`	Patchstack technical analysis — source article with IOC and remediation detail	HIGH
URL	`https://sansec.io/research/optinmonster-supply-chain-attack`	Sansec research — primary technical source for payload analysis and IOC detail	HIGH

Framework Mappings

MITRE-ATTACK

- **T1505.003** — Web Shell
- **T1071.001** — Web Protocols
- **T1078.003** — Local Accounts
- **T1136** — Create Account
- **T1078.001** — Default Accounts
- **T1195.002** — Compromise Software Supply Chain
- **T1190** — Exploit Public-Facing Application
- **T1036.005** — Match Legitimate Resource Name or Location
- **T1059.007** — JavaScript
- **T1539** — Steal Web Session Cookie
- **T1564.001** — Hidden Files and Directories

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
- **AC-2** — Account Management
- **AC-6** — Least Privilege

- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **IA-5** — Authenticator Management
- **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
- **A04:2021** — Insecure Design

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **5.2** — Use Unique Passwords
- **6.3** — Require MFA for Externally-Exposed Applications
- **15.1** — Establish and Maintain an Inventory of Service Providers

ISO-27001-2022

- **A.8.28** — Secure coding
- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

HIPAA-SECURITY

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

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1505.003	Web Shell	Persistence
T1071.001	Web Protocols	Command-And-Control
T1078.003	Local Accounts	Defense-Evasion
T1136	Create Account	Persistence
T1078.001	Default Accounts	Defense-Evasion
T1195.002	Compromise Software Supply Chain	Initial-Access
T1190	Exploit Public-Facing Application	Initial-Access
T1036.005	Match Legitimate Resource Name or Location	Defense-Evasion
T1059.007	JavaScript	Execution
T1539	Steal Web Session Cookie	Credential-Access
T1564.001	Hidden Files and Directories	Defense-Evasion

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/optinmonster-wordpre...	T3
Supply Chain Attack on OptinMonster, TrustPulse, and PushEngage	https://patchstack.com/articles/supply-chain-attack-on-optinmonster...	T3
Popular WordPress Plugin Scripts Tampered to Plant Hidden ...	https://thehackernews.com/2026/06/popular-wordpress-plugin-scripts....	T3
OptinMonster supply chain attack hits 1.2 million sites - Sansec	https://sansec.io/research/optinmonster-supply-chain-attack	T3
Attackers compromised Awesome Motive CDN files, backdooring ...	https://securityaffairs.com/193616/malware/supply-chain-attack-hits...	T3

Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-06-16 07:13 UTC by TJS Security Command Center