INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-18 13:45 UTC

Open-Sourced Shai-Hulud Worm Fuels Multi-Payload npm Campaign: Infostealers, Phantom Bot, and a BreachForums Competition Signal What's Coming

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0331
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	npm ecosystem, GitHub, Windows, Linux, cloud credential stores, cryptocurrency wallets, SSH environments, IDE/coding agents (Claude Code)
Published	2026-05-18T04:57:26
Discovery Source	Rss

Executive Summary

A threat actor published four malicious packages to npm, collectively downloaded nearly 3,000 times, delivering credential-stealing malware and a DDoS botnet. The campaign targets cloud credentials, cryptocurrency wallets, SSH keys, and AI-assisted development environments including Claude Code. OX Security analysts assess this as a likely precursor to broader supply chain attacks, correlated with an active BreachForums competition incentivizing supply chain attack development.

Technical Analysis

Threat actor 'deadcode09284814' published four malicious npm packages with approximately 3,000 combined downloads. The payload arsenal includes: (1) an infostealer targeting cloud credential stores, cryptocurrency wallets, and SSH keys; (2) Phantom Bot, a Golang-based DDoS botnet; and (3) a direct clone of the recently open-sourced Shai-Hulud worm, reconfigured with a new C2 server. The Shai-Hulud clone demonstrates rapid weaponization, the worm was open-sourced, forked, and redeployed with a replacement C2 within the same campaign window. The campaign extends attack surface into IDE and AI coding agent environments by deploying hook-based session backdoors targeting Claude Code. Relevant CWEs: CWE-506 (Embedded Malicious Code), CWE-798 (Hardcoded Credentials), CWE-312 (Cleartext Storage of Sensitive Information), CWE-494 (Download of Code Without Integrity Check). MITRE ATT&CK coverage includes T1195.001 (Supply Chain Compromise: Compromise Software Dependencies), T1555 (Credentials from Password Stores),

T1552.001 (Unsecured Credentials: Credentials in Files), T1041 (Exfiltration Over C2 Channel), T1496 (Resource Hijacking), T1498 (Network Denial of Service), T1547.001 (Boot or Logon Autostart: Registry Run Keys), T1059 (Command and Scripting Interpreter), and T1071.001 (Application Layer Protocol: Web Protocols). No CVE assigned. Source quality is primarily T3 (news outlets and vendor blogs); specific package names, hashes, C2 endpoints, and IOCs should be verified directly against OX Security's published research and the npm registry before acting on specifics.

Action Checklist

1. Step 1: Containment, Audit all npm dependencies installed or updated in the past 30 days. Search your dependency manifests and lock files for packages published by 'deadcode09284814'. Block outbound connections to any unrecognized C2 endpoints identified in OX Security's published IOC list. Isolate any developer workstation that installed one of the four flagged packages.
2. Step 2: Detection, Query package manager logs and CI/CD pipeline logs for installs of the four malicious packages (cross-reference against OX Security's full disclosure and SafeDep report for confirmed package names). On affected Linux and Windows hosts, check for new scheduled tasks (T1053.005), registry run key modifications (T1547.001), and unexpected outbound connections on non-standard ports. In Claude Code and other IDE environments, audit installed hooks and extensions for unauthorized session interceptors. Review SSH known_hosts and authorized_keys files for unauthorized entries.
3. Step 3: Eradication, Remove identified malicious packages using 'npm uninstall' and clear the npm cache. Rotate all credentials accessible from affected developer environments: cloud provider API keys (AWS, GCP, Azure), SSH private keys, and cryptocurrency wallet seeds. Revoke and reissue any CI/CD pipeline tokens or secrets exposed in affected environments. Remove any persistence mechanisms identified (scheduled tasks, run keys, hook files).
4. Step 4: Recovery, Revalidate dependency integrity across affected projects using 'npm audit' and a software composition analysis (SCA) tool against a known-clean baseline. Confirm no unauthorized packages remain in lock files. Monitor outbound network traffic from recovered workstations for 72 hours for residual C2 beaconing. Verify cloud provider access logs for unauthorized API calls following credential rotation.
5. Step 5: Post-Incident, This campaign exposes three control gaps: (1) absence of pre-install package vetting or allowlisting in developer pipelines, (2) no integrity verification (checksums or signed packages) at install time (CWE-494), and (3) insufficient secrets management, credentials accessible in developer environments rather than vault-isolated. Implement a dependency review gate in CI/CD pipelines. Evaluate adoption of npm package provenance attestation. Extend SCA scanning to IDE plugin ecosystems and AI coding agent hook directories.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate immediately to CISO and legal counsel if AWS CloudTrail, GCP Audit Logs, or Azure Activity Logs confirm unauthorized API calls using harvested credentials during the exposure window, if any affected developer had access to production secrets or signing keys (indicating potential supply chain contamination of production artifacts), or if the organization's CI/CD pipelines built and published packages or container images from a compromised build environment — any of these conditions may trigger breach notification obligations under GDPR, CCPA, or SOC 2 contractual requirements.
Recovery Notes	After credential rotation, conduct a 72-hour network monitoring window specifically watching for Phantom Bot C2 beaconing patterns and any DNS resolution attempts against IOC domains from the OX Security/safedep.io disclosure, as incomplete persistence removal (scheduled tasks, run keys, or surviving hook files) will cause resumed C2 contact even after package removal. Audit all production artifacts (npm packages, container images, binaries) built within the 30-day exposure window for evidence of tampering — the Shai-Hulud worm's capability to pivot via SSH means a compromised developer workstation may have been used as a lateral movement vector into build infrastructure beyond the initial npm install target. Cloud provider access logs should be reviewed for the full exposure window to identify any resources created, modified, or exfiltrated using harvested API keys before rotation, and those findings must be documented regardless of whether external breach notification thresholds are met.
Forensic Artifacts	npm install debug logs at ~/.npm/_logs/*.log — contain timestamped postinstall script execution output showing the exact commands run by the malicious packages' lifecycle hooks (preinstall/postinstall entries in package.json), establishing the initial execution timeline for this campaign | Sysmon Event ID 1 (Process Creation) and Event ID 3 (Network Connection) entries filtered to parent process node.exe — the Shai-Hulud worm and infostealer components spawn child processes and initiate outbound connections during npm install lifecycle hook execution, leaving a precise execution chain trace | Browser extension local storage directories (Chrome: %APPDATA%\Local\Google\Chrome\User Data\Default\Local Extension Settings\, Firefox: %APPDATA%\Roaming\Mozilla\Firefox\Profiles*.default\storage\) — the infostealer component specifically targets cryptocurrency wallet browser extensions, and these directories contain the harvested wallet data or evidence of access | ~/.claude/hooks/ directory and ~/.claude/claude_desktop_config.json modification timestamps and contents — the campaign specifically targets Claude Code AI coding agent environments for session interception, and unauthorized hook entries or MCP server additions in these files are campaign-specific artifacts not present in standard IR investigations | SSH ~/.ssh/authorized_keys modification timestamps and new entries across all accounts on developer workstations and any servers accessible via SSH from affected environments — the Shai-Hulud worm's SSH pivoting capability leaves forensic traces in authorized_keys files that establish lateral movement scope beyond the initially compromised developer workstation

Per-Action IR Details

Step 1: Containment — Audit all npm dependencies installed or updated in the past 30 days. Search your dependency manifests and lock files for packages published by 'deadcode09284814'. Block outbound connections to any unrecognized C2 endpoints identified in OX Security's published IOC list. Isolate any developer workstation that installed one of the four flagged packages.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SI-3 (Malicious Code Protection), CIS 2.3 (Address Unauthorized Software), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Run 'npm ls --all 2>/dev/null | grep deadcode09284814' across all developer workstations and CI/CD build agents. On Windows, use PowerShell: 'Get-Content package-lock.json | Select-String deadcode09284814'. For network blocking without enterprise firewall infrastructure, add C2 IP/domain IOCs from the OX Security/safedep.io disclosure to Windows Hosts file (C:\Windows\System32\drivers\etc\hosts) or /etc/hosts on Linux, and create an outbound drop rule via 'netsh advfirewall' (Windows) or 'iptables -A OUTPUT -d -j DROP' (Linux). Isolate affected workstations by disabling their network adapter ('Disable-NetAdapter -Name * -Confirm:$false' on Windows) rather than full shutdown, to preserve volatile memory evidence.

Evidence: Before isolating, capture: (1) full npm global and local package lists via 'npm list -g --depth=0' and 'npm list --depth=0' in each project directory; (2) active network connections showing any established sessions to C2 endpoints using 'netstat -anob' (Windows) or 'ss -tulnp' (Linux) — the Shai-Hulud worm and Phantom Bot component will have established outbound connections on non-standard ports; (3) running process list via 'Get-Process' or 'ps aux' to capture any in-memory malware processes spawned during package install lifecycle hooks (preinstall/postinstall scripts in package.json); (4) contents of ~/.npm/_logs/ for install-time script execution evidence.

Step 2: Detection — Query package manager logs and CI/CD pipeline logs for installs of the four malicious packages (names to be confirmed against OX Security's full disclosure — see safedep.io report). On affected Linux and Windows hosts, check for new scheduled tasks (T1053.005), registry run key modifications (T1547.001), and unexpected outbound connections on non-standard ports. In Claude Code and other IDE environments, audit installed hooks and extensions for unauthorized session interceptors. Review SSH known_hosts and authorized_keys files for unauthorized entries.

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), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Deploy Sysmon with SwiftOnSecurity config to capture process creation (Event ID 1) and network connections (Event ID 3) on Windows developer workstations — filter for node.exe and npm spawning child processes (cmd.exe, powershell.exe, curl, wget) as evidence of malicious postinstall hook execution. Query Windows Security Event Log for Event ID 4698 (Scheduled Task Created) and Event ID 4657 (Registry Value Modified) filtered to HKCU\Software\Microsoft\Windows\CurrentVersion\Run. On Linux, run 'crontab -l' and 'ls -la /etc/cron.*/' for all user accounts. For Claude Code hook audit, inspect ~/.claude/hooks/ and any MCP server configurations in ~/.claude/claude_desktop_config.json for unauthorized entries. For SSH audit: 'cat ~/.ssh/authorized_keys' and 'cat ~/.ssh/known_hosts' on all affected developer accounts, diffing against a known-good baseline if available. Use osquery query: 'SELECT * FROM scheduled_tasks' (Windows) and 'SELECT * FROM crontab' (Linux).

Evidence: Capture before analysis: (1) CI/CD pipeline build logs from the past 30 days — GitHub Actions logs at .github/workflows/ artifact store, Jenkins build console output — filtering for npm install commands that resolved packages from the 'deadcode09284814' publisher account; (2) npm install debug logs at ~/.npm/_logs/*.log showing postinstall script execution for the four flagged packages; (3) Sysmon Event ID 3 (Network Connection) entries showing node.exe initiating outbound connections, particularly during 'npm install' execution windows — the Phantom Bot C2 beaconing and infostealer exfiltration would appear here; (4) ~/.ssh/authorized_keys and ~/.ssh/known_hosts modification timestamps — the worm's SSH pivoting capability (Shai-Hulud) may have added entries; (5) Claude Code session logs and MCP tool invocation history if the IDE hook interceptor component was active.

Step 3: Eradication — Remove identified malicious packages using 'npm uninstall' and clear the npm cache. Rotate all credentials accessible from affected developer environments: cloud provider API keys (AWS, GCP, Azure), SSH private keys, and cryptocurrency wallet seeds. Revoke and reissue any CI/CD pipeline tokens or secrets exposed in affected environments. Remove any persistence mechanisms identified (scheduled tasks, run keys, hook files).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST AC-2 (Account Management), CIS 5.1 (Establish and Maintain an Inventory of Accounts), CIS 6.2 (Establish an Access Revoking Process)

Compensating: Eradication sequence: (1) 'npm uninstall && npm cache clean --force' on all affected workstations and in CI/CD environments; (2) For AWS key rotation, use 'aws iam create-access-key' then 'aws iam delete-access-key --access-key-id ' — check ~/.aws/credentials and environment variables (printenv | grep -i aws) for stored keys the infostealer would have harvested; (3) For GCP: 'gcloud iam service-accounts keys delete' and revoke Application Default Credentials at ~/.config/gcloud/; (4) Revoke GitHub/GitLab CI tokens via repository settings and regenerate — the infostealer targets .env files and CI secret stores; (5) Remove SSH keys: generate new keypairs with 'ssh-keygen -t ed25519', update authorized_keys on all servers the affected developer had access to, and revoke old public keys; (6) Delete scheduled tasks: 'schtasks /delete /tn /f' (Windows) or 'crontab -e' (Linux); remove run key entries via 'reg delete HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v /f'.

Evidence: Before eradication, preserve forensic copies of: (1) the malicious package contents from npm cache (~/.npm/cache/) — the actual malicious JavaScript files, package.json with lifecycle hooks, and any dropped binaries constitute primary malware evidence; (2) any files dropped to temp directories (%TEMP%, /tmp, /var/tmp) by postinstall scripts — the Shai-Hulud worm drops additional payloads post-install; (3) full contents of ~/.aws/credentials, ~/.azure/, ~/.config/gcloud/ to document the credential exposure scope before rotation; (4) cryptocurrency wallet files targeted — browser extension storage (Chrome: %APPDATA%\Local\Google\Chrome\User Data\Default\Local Extension Settings\) for wallet extensions, and any wallet.dat or keystore files in scope; (5) a memory dump of any suspicious node.exe or spawned child processes using ProcDump ('procdump.exe -ma ') to capture in-memory credential harvesting state.

Step 4: Recovery — Revalidate dependency integrity across affected projects using 'npm audit' and a software composition analysis (SCA) tool against a known-clean baseline. Confirm no unauthorized packages remain in lock files. Monitor outbound network traffic from recovered workstations for 72 hours for residual C2 beaconing. Verify cloud provider access logs for unauthorized API calls following credential rotation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: For dependency revalidation without enterprise SCA tooling: run 'npm audit --audit-level=high' and cross-reference package-lock.json against the OX Security/safedep.io IOC list using 'grep -f ioc_packages.txt package-lock.json'. Use the free Socket.dev CLI ('npx socket scan .') to detect malicious packages by behavioral analysis beyond CVE matching. For 72-hour C2 beaconing detection on a budget, capture outbound traffic with Wireshark ('tshark -i eth0 -w /tmp/recovery_capture.pcap') or enable Windows Firewall verbose logging to %systemroot%\system32\LogFiles\Firewall\pfirewall.log. For cloud provider audit: run 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue=' for the 30-day window pre-rotation; for GCP use 'gcloud logging read "protoPayload.methodName=\"google.iam.admin.v1.*\"" to identify any IAM changes made with compromised credentials.

Evidence: During recovery validation, collect: (1) npm audit output and lock file diffs showing clean state post-eradication — these document completeness of removal; (2) AWS CloudTrail, GCP Audit Logs, and Azure Activity Log entries from the credential exposure window showing any unauthorized API calls (resource creation, IAM policy changes, S3/GCS bucket access) that would indicate the infostealer-harvested credentials were actively used by the threat actor; (3) Wireshark/tcpdump packet captures from the 72-hour monitoring window specifically filtering for DNS queries and TCP connections to domains/IPs in the Shai-Hulud/Phantom Bot C2 IOC list — residual beaconing would indicate incomplete eradication of persistence mechanisms; (4) SSH authentication logs (/var/log/auth.log or /var/log/secure) on servers accessible from affected developer accounts, covering the entire exposure window through 72 hours post-key rotation.

Step 5: Post-Incident — This campaign exposes three control gaps: (1) absence of pre-install package vetting or allowlisting in developer pipelines, (2) no integrity verification (checksums or signed packages) at install time (CWE-494), and (3) insufficient secrets management — credentials accessible in developer environments

rather than vault-isolated. Implement a dependency review gate in CI/CD pipelines. Evaluate adoption of npm package provenance attestation. Extend SCA scanning to IDE plugin ecosystems and AI coding agent hook directories.

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-7 (Software, Firmware, and Information Integrity), NIST CM-2 (Baseline Configuration) — implied by lock file integrity requirement, CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Implement three durable controls without enterprise tooling budget: (1) Add a pre-install gate to CI/CD pipelines using a free GitHub Action ('naugtur/check-dependency-author@v1' or equivalent) that blocks packages from publishers with fewer than N total downloads or accounts created within the past 90 days — 'deadcode09284814' is a newly created account, a pattern detectable at install time; (2) Enable npm package provenance verification where available ('npm install --prefer-offline' with 'npm config set ignore-scripts true' as a default to prevent postinstall hook execution in CI — the primary delivery mechanism for this campaign); (3) Migrate developer secrets from .env files and ~/.aws/credentials to a secrets manager — HashiCorp Vault Community Edition (free, self-hosted) or AWS Secrets Manager with short-lived credentials via IAM roles, eliminating the static credential files the infostealer component harvests. For Claude Code and AI coding agent hook directories, create a weekly cron job that checksums ~/.claude/hooks/ and alerts on changes.

Evidence: For lessons-learned documentation, preserve and analyze: (1) the full timeline of package download counts (available via npm registry API: 'https://api.npmjs.org/downloads/point/last-month/') correlating the ~3,000 downloads to affected internal systems — this quantifies blast radius for incident reporting; (2) the BreachForums competition post referenced in OX Security's report as threat intelligence context — document this as a threat actor motivation indicator for future supply chain attack prediction; (3) all rotated credential identifiers (key IDs, not values) and the timestamp of last successful authentication per credential, to bound the unauthorized access window for any required breach notification assessment; (4) CI/CD pipeline build history showing which production artifacts were built on affected developer workstations or build agents during the exposure window — if a compromised developer environment contributed to a production build, that artifact integrity is now unverifiable and may require rebuild and redeployment.

Detection Guidance

Primary detection surface is the npm install chain and network egress. Query CI/CD and workstation package manager logs for installs of packages attributed to publisher 'deadcode09284814', cross-reference against OX Security and SafeDep disclosures for confirmed package names and validate against the npm registry before actioning. On Windows hosts, check for new scheduled tasks via 'schtasks /query' and inspect HKCU\Software\Microsoft\Windows\CurrentVersion\Run and HKLM equivalents for unfamiliar entries (T1547.001, T1053.005). On Linux, review /etc/cron* and user crontabs. For the Shai-Hulud worm component, look for unexpected process spawning with outbound TCP connections to non-organizational IPs, consistent with worm self-propagation behavior (T1041). For Phantom Bot (Golang DDoS botnet), look for high-volume outbound UDP or TCP flood traffic from developer workstations, anomalous for that asset class. In Claude Code and similar IDE environments, audit hook configuration files for entries not added by the development team. For credential theft indicators, check cloud provider CloudTrail/audit logs for API calls from unexpected source IPs or at unusual times following any confirmed package install. IOC specifics including package names, hashes, and C2 endpoints should be obtained directly from OX Security's and SafeDep's published research; T3 source limitations require direct verification against authoritative sources before operational deployment.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`[C2 domain — not reproduced: obtain from OX Security or SafeDep published disclosure]`	Shai-Hulud worm clone C2 server, reconfigured by deadcode09284814	LOW
HASH	`[Package hashes — not reproduced: obtain from OX Security or SafeDep published disclosure]`	Four malicious npm packages published by deadcode09284814	LOW

Framework Mappings

MITRE-ATTACK

- **T1041** — Exfiltration Over C2 Channel
- **T1496** — Resource Hijacking
- **T1547.001** — Registry Run Keys / Startup Folder
- **T1587.001** — Malware
- **T1078** — Valid Accounts
- **T1059.004** — Unix Shell
- **T1053.005** — Scheduled Task
- **T1498** — Network Denial of Service
- **T1078.001** — Default Accounts
- **T1071.001** — Web Protocols
- **T1552.001** — Credentials In Files
- **T1555** — Credentials from Password Stores
- **T1087** — Account Discovery
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059** — Command and Scripting Interpreter

NIST-800-53R5

- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-7** — Software, Firmware, and Information Integrity

- **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

- **16.10** — Apply Secure Design Principles in Application Architectures
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **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.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

HIPAA-SECURITY

- **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
T1041	Exfiltration Over C2 Channel	Exfiltration
T1496	Resource Hijacking	Impact
T1547.001	Registry Run Keys / Startup Folder	Persistence
T1587.001	Malware	Resource-Development
T1078	Valid Accounts	Defense-Evasion
T1059.004	Unix Shell	Execution
T1053.005	Scheduled Task	Execution

Technique ID	Technique Name	Tactic
T1498	Network Denial of Service	Impact
T1078.001	Default Accounts	Defense-Evasion
T1071.001	Web Protocols	Command-And-Control
T1552.001	Credentials In Files	Credential-Access
T1555	Credentials from Password Stores	Credential-Access
T1087	Account Discovery	Discovery
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1059	Command and Scripting Interpreter	Execution

Sources

Source	URL	Tier
Security News	https://thehackernews.com/2026/05/four-malicious-npm-packages-deliv...	T3
Malicious npm Packages Backdoor Claude Code Sessions - SafeDep	https://safedep.io/malicious-npm-packages-claude-code-hooks	T3
GitHub takes aim at Claude Code and Codex with its new…	https://www.develeap.com/news/github-takes-aim-at-claude-code-and-c...	T3
Developers warned of massive supply chain attack - Facebook	https://www.facebook.com/groups/icthubkenya/posts/4193274857482502/	T3
Financial Advisors Warned of Active NPM Exploit - LinkedIn	https://www.linkedin.com/posts/anand951_financialadvisors-activity-...	T3

Generated 2026-05-18 13:45 UTC by TJS Security Command Center