INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-18 18:49 UTC

Shai-Hulud Malware Leak Enables Second-Actor npm Typosquatting Campaign with DDoS and Infostealer Payloads

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0332
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	npm packages: chalk-tempalte, @deadcode09284814/axios-util, axois-utils, color-style-utils; Node.js developers; Axios library users; cryptocurrency wallets; cloud configuration credentials (AWS, GCP, Azure config files)
Published	2026-05-18T13:28:02
Discovery Source	Rss

Executive Summary

A leaked infostealer toolkit called Shai-Hulud has been weaponized by a second, unidentified threat actor who published four malicious packages to the npm registry, targeting Node.js developers who use the Axios HTTP library. The packages steal cloud credentials (AWS, GCP, Azure), cryptocurrency wallet data, and developer secrets, while at least one package also recruits infected machines into a DDoS botnet. Any organization with Node.js development pipelines that installed one of the four typosquatted packages may have exposed cloud infrastructure credentials and is at elevated risk of further compromise.

Technical Analysis

Four malicious npm packages were published using typosquatting against the legitimate Axios library: chalk-tempalte, @deadcode09284814/axios-util, axois-utils, and color-style-utils. The packages carry a modified variant of the Shai-Hulud infostealer, originally developed by the group tracked as TeamPCP, whose source code was leaked publicly on GitHub. A second, unattributed actor incorporated the leaked code and extended it with a persistent DDoS botnet module. Payloads execute via CWE-506 (embedded malicious code) upon npm install or import, exfiltrating credentials stored in plaintext config files (CWE-312) without integrity verification (CWE-494). Relevant MITRE ATT&CK techniques include T1195.001 (Compromise Software Supply Chain), T1059.007 (JavaScript execution), T1552.001 (Credentials in Files), T1498 (Network DoS), T1567/T1567.001 (Exfiltration Over Web Service), and T1071.001 (Application Layer Protocol C2). The attacker acquired the

malware source via T1588.001 (Obtain Capabilities: Malware). No CVE has been assigned. A representative CVSS score for the infostealer payload is 7.5, though this campaign-level item is rated editorially as High based on attack scope and exposure conditions. Confidence in full artifact-level details is medium, sourced from BleepingComputer (T3) corroborated by vendor reports from Microsoft, Huntress, Trend Micro, and Cloudsmith. Primary artifact analysis has not been independently confirmed in this session.

Action Checklist

1. Step 1: Containment, Audit all npm package manifests (package.json and package-lock.json) across dev, CI/CD, and production environments for the four malicious packages: chalk-tempalte, @deadcode09284814/axios-util, axois-utils, color-style-utils. Remove any match immediately and revoke all secrets accessible from affected build environments. Reference: Microsoft advisory on axios npm supply chain compromise (expected publication 2026-04-01; consult Microsoft Security Blog for confirmation).
2. Step 2: Detection, Search npm install logs, CI/CD pipeline logs, and SIEM telemetry for installs of the four package names. Look for outbound HTTP/HTTPS connections to unknown endpoints initiated by Node.js processes at install time (T1071.001). Check for unexpected reads of ~/.aws/credentials, ~/.config/gcloud/, Azure config directories, and cryptocurrency wallet files. Behavioral indicators include JavaScript spawning shell processes (T1059.007) and bulk file reads followed by outbound POST requests.
3. Step 3: Eradication, Remove the malicious packages using npm uninstall for each identified package. Rotate all cloud credentials (AWS IAM keys, GCP service account keys, Azure service principals) that were present on any system where the packages were installed. Rotate any API keys, tokens, or secrets stored in environment files accessible during the compromised build. Verify the legitimate axios package version in use is sourced from the official registry and matches expected hashes via npm audit.
4. Step 4: Recovery, After credential rotation, verify no unauthorized IAM users, roles, or cloud resources were created (T1136.003) using cloud provider audit logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Log). Monitor for anomalous outbound traffic patterns consistent with DDoS botnet C2 beaconing. Confirm CI/CD pipelines rebuild from clean dependency lockfiles. Run npm audit on all affected projects post-remediation.
5. Step 5: Post-Incident, This campaign exposes gaps in dependency vetting and supply chain integrity verification. Implement controls including: npm package allowlisting or private registry mirroring; automated typosquatting detection in CI pipelines; Subresource Integrity or lockfile pinning enforcement; developer secrets management (replace plaintext config files with secrets managers). Map gaps to CIS Control 2 (Inventory and Control of Software Assets) and NIST SP 800-161 (Supply Chain Risk Management).

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate immediately to CISO and legal counsel if AWS CloudTrail, GCP Audit Logs, or Azure Activity Log confirm any API calls using the compromised credentials after the install timestamp, or if cryptocurrency wallet files (e.g., Metamask vault, `~/.bitcoin/wallet.dat`) were present on affected systems, as either condition establishes confirmed data exfiltration and may trigger breach notification obligations under applicable data protection regulations.
Recovery Notes	After credential rotation, maintain heightened monitoring of all newly issued cloud credentials and CI/CD pipeline executions for a minimum of 30 days, as Shai-Hulud-sourced campaigns have demonstrated persistence mechanisms (cron, systemd) that survive package removal and may re-establish C2 contact using pre-staged credentials or tokens not covered by the initial rotation. Verify clean rebuild by running `npm ci --ignore-scripts` on all affected projects from a fresh working directory and confirming `npm audit` returns zero high/critical findings against the restored lockfile. Specifically validate that no npm scripts in the dependency tree reference external URLs or spawn shell commands, using `npm audit --json | jq '.metadata'` and manual review of `scripts` fields in all first-level dependency package.json files.
Forensic Artifacts	npm install logs and CI/CD pipeline stdout/stderr logs containing install timestamps for chalk-tempalte, @deadcode09284814/axios-util, axois-utils, or color-style-utils — typically found at ~/.npm/_logs/ on developer workstations and in CI runner job logs (GitHub Actions: workflow run logs; Jenkins: build console output at $JENKINS_HOME/jobs//builds//log) | File access-time (atime) metadata on ~/.aws/credentials, ~/.config/gcloud/application_default_credentials.json, ~/.azure/accessTokens.json, and any .env files in the project root — a modified atime within seconds of the npm install timestamp is direct evidence that the Shai-Hulud infostealer payload accessed these files | Sysmon Event ID 1 (Process Create) or auditd EXECVE records showing node.exe or node spawning sh, bash, cmd.exe, or powershell.exe as a child process during the npm install window — this is the behavioral signature of Shai-Hulud's postinstall script execution (ATT&CK T1059.007) | Network flow records or pcap captures showing outbound HTTP POST requests from node processes to non-npm-registry endpoints during or immediately after install, particularly to hardcoded C2 IPs or domains embedded in the Shai-Hulud toolkit — these represent the credential exfiltration channel (ATT&CK T1071.001) | AWS CloudTrail management event logs, GCP Admin Activity audit logs, and Azure Activity Log entries for the IAM/identity actions CreateUser, CreateRole, AttachRolePolicy, AssumeRole (AWS), CreateServiceAccount, SetIAMPolicy (GCP), and New-AzADServicePrincipal (Azure) occurring after the install timestamp and before credential rotation — these confirm whether the stolen credentials were actively leveraged for cloud account persistence (ATT&CK T1136.003)

Per-Action IR Details

Step 1: Containment — Audit all npm package manifests (package.json and package-lock.json) across dev, CI/CD, and production environments for the four malicious packages: chalk-tempalte, @deadcode09284814/axios-util, axois-utils, color-style-utils. Remove any match immediately and revoke all secrets accessible from affected build environments. Reference: Microsoft advisory (https://www.microsoft.com/en-us/security/blog/2026/04/01/mitigating-the-axios-npm-supply-chain-compromise/).

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software)

Compensating: Run the following one-liner across all repositories and CI workspaces to surface hits before manual removal: `find . -name 'package.json' -o -name 'package-lock.json' | xargs grep -l 'chalk-tempalte\|axois-utils\|color-style-utils\|@deadcode09284814'`. For CI/CD runners (GitHub Actions, GitLab CI, Jenkins), grep build artifact caches and node_modules directories: `find /home/runner /var/lib/jenkins -type d -name

'chalk-tempalte' -o -name 'axois-utils' 2>/dev/null`. Immediately revoke AWS IAM keys visible in any .env or credentials file on the affected host using `aws iam delete-access-key --access-key-id `.

Evidence: Before removing packages, snapshot the full node_modules directory tree (`ls -laR node_modules/ > node_modules_snapshot.txt`) and capture the resolved dependency tree (`npm ls --all > npm_tree.txt`). Preserve the original package-lock.json with file hash (`sha256sum package-lock.json`) as tamper evidence. If the malicious package was executed, capture a process list (`ps aux` or `Get-Process`) and active network connections (`ss -tunp` or `netstat -ano`) before killing any processes, since Shai-Hulud payloads establish outbound C2 connections at install time.

Step 2: Detection — Search npm install logs, CI/CD pipeline logs, and SIEM telemetry for installs of the four package names. Look for outbound HTTP/HTTPS connections to unknown endpoints initiated by Node.js processes at install time (T1071.001). Check for unexpected reads of ~/.aws/credentials, ~/.config/gcloud/, Azure config directories, and cryptocurrency wallet files. Behavioral indicators include JavaScript spawning shell processes (T1059.007) and bulk file reads followed by outbound POST requests.

NIST Phase: Detection Analysis

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

Controls: NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs)

Compensating: Without a SIEM, use osquery to hunt for file access artifacts: `SELECT * FROM file WHERE path LIKE '/home/%/.aws/credentials' AND atime > (SELECT strftime('%s','now','-7 days'));`. On Linux developer workstations, use auditd rules to detect reads of credential files: `auditctl -w /home -p r -k shai_hulud_cred_access`. For network detection, run Wireshark or tcpdump during a controlled npm install in an isolated VM: `tcpdump -i eth0 -w shai_hulud_capture.pcap 'host not '`. Deploy the Sigma rule concept manually: parse CI/CD logs (e.g., GitHub Actions runner logs at `~/.local/share/github-runner/_diag/`) for lines containing any of the four package names alongside timestamps, then correlate with firewall or proxy logs for outbound POSTs from the same host within a 60-second window.

Evidence: Pull npm install logs from CI runners — on GitHub Actions these are stored in the workflow run logs and locally in `~/.npm/_logs/`. Check Node.js process audit trail: on Linux, auditd syscall logs (event type EXECVE) will show `node` spawning `sh` or `bash` child processes (ATT&CK T1059.007) if Shai-Hulud's postinstall script executed. On Windows developer machines, query Sysmon Event ID 1 (Process Create) for `node.exe` parent with `cmd.exe` or `powershell.exe` child. Capture DNS query logs or proxy logs for domains resolved by `node` processes during the install window, as the Shai-Hulud infostealer component exfiltrates via outbound POST to a hardcoded C2. Access-time metadata on `~/.aws/credentials`, `~/.config/gcloud/application_default_credentials.json`, and any `.env` files is direct evidence of credential harvesting.

Step 3: Eradication — Remove the malicious packages using npm uninstall for each identified package. Rotate all cloud credentials (AWS IAM keys, GCP service account keys, Azure service principals) that were present on any system where the packages were installed. Rotate any API keys, tokens, or secrets stored in environment files accessible during the compromised build. Verify the legitimate axios package version in use is sourced from the official registry and matches expected hashes via npm audit.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST AC-2 (Account Management), CIS 5.3 (Disable Dormant Accounts), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For credential rotation without enterprise tooling: AWS — use `aws iam list-access-keys --user-name ` to enumerate all keys on affected IAM identities, then `aws iam delete-access-key` for each compromised key and `aws iam create-access-key` for replacement; enable AWS CloudTrail and immediately run `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=AssumeRole` to detect any lateral movement using stolen keys before rotation. GCP — revoke service account keys via `gcloud iam service-accounts keys delete --iam-account=`. For lockfile integrity verification without a paid registry: `npm ci --ignore-scripts` with a pinned package-lock.json prevents postinstall script execution during reinstall; verify axios integrity with `npm view

axios dist.tarball` and compare `sha512` integrity field in package-lock.json against the published registry value.

Evidence: Before rotating credentials, document all currently active AWS IAM access keys (`aws iam list-users | xargs -I{} aws iam list-access-keys --user-name {}`), GCP service account keys (`gcloud iam service-accounts keys list`), and Azure service principal secrets (`az ad sp credential list`) as a baseline for post-rotation comparison. Preserve any `.env`, `.envrc`, or CI/CD environment variable exports that were readable during the compromised build — these define the exact secret scope that was accessible to the Shai-Hulud payload. Verify npm cache integrity: malicious packages may persist in `~/.npm/_cacache/`; enumerate with `npm cache verify` and cross-reference cached package hashes against the four known-bad package names.

Step 4: Recovery — After credential rotation, verify no unauthorized IAM users, roles, or cloud resources were created (T1136.003) using cloud provider audit logs (AWS CloudTrail, GCP Audit Logs, Azure Activity Log). Monitor for anomalous outbound traffic patterns consistent with DDoS botnet C2 beaconing. Confirm CI/CD pipelines rebuild from clean dependency lockfiles. Run npm audit on all affected projects post-remediation.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (System Monitoring), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Query AWS CloudTrail (free, enabled by default for management events in us-east-1) for unauthorized account creation: `aws cloudtrail lookup-events --lookup-attributes AttributeKey=EventName,AttributeValue=CreateUser --start-time `. For DDoS botnet C2 detection without a SIEM, use Wireshark or tcpdump to capture 30 minutes of outbound traffic from affected hosts and look for periodic beaconing patterns (regular interval connections to a single external IP, characteristic of the color-style-utils DDoS recruiter component). Write a Sigma-style detection as a cron job that tails `/var/log/syslog` or Windows Event Log (Event ID 5156, Network Connection) for `node.exe` making connections to IPs not in your npm registry or CDN allowlist. For CI/CD pipeline verification, enforce `npm ci` (which requires and validates package-lock.json) over `npm install` in all pipeline definitions, and add `--ignore-scripts` flag to block postinstall execution during recovery builds.

Evidence: Pull AWS CloudTrail event history filtered to the time window between initial malicious package install and credential rotation — specifically query for `CreateUser`, `CreateRole`, `AttachRolePolicy`, `RunInstances`, and `CreateBucket` API calls originating from the compromised IAM key. In GCP Audit Logs, filter for `google.iam.admin.v1.CreateServiceAccount` and `google.iam.admin.v1.SetIAMPolicy` events. For DDoS botnet persistence, check for cron entries (`crontab -l`, `/etc/cron.d/`, `/var/spool/cron/`) or systemd unit files added post-install, as the color-style-utils botnet recruiter may establish persistence to survive a simple npm uninstall. Capture current network connection state (`ss -tunp | grep node`) as a clean-state baseline for comparison during the monitoring window.

Step 5: Post-Incident — This campaign exposes gaps in dependency vetting and supply chain integrity verification. Implement controls including: npm package allowlisting or private registry mirroring; automated typosquatting detection in CI pipelines; Subresource Integrity or lockfile pinning enforcement; developer secrets management (replace plaintext config files with secrets managers). Map gaps to CIS Control 2 (Inventory and Control of Software Assets) and NIST SP 800-161 (Supply Chain Risk Management).

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-7 (Software, Firmware, and Information Integrity), NIST SI-2 (Flaw Remediation), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: For teams without a private registry budget: configure npm to use an `.npmrc` file with `audit=true` and `fund=false` and add a pre-install hook using `npx npm-audit-resolver` to block known-bad packages. Implement free typosquatting detection in CI by scripting a Levenshtein distance check against your approved package list — a 50-line Python script using the `difflib` module can flag packages within edit-distance 2 of known-good names (would

have caught `chalk-tempalte` vs `chalk-template` and `axois-utils` vs `axios`). Replace plaintext `~/.aws/credentials` files with `aws configure --profile` using IAM roles for EC2/container workloads (free), or HashiCorp Vault open-source edition for developer secrets. Add a YARA rule to your CI pipeline to scan node_modules at build time for known Shai-Hulud payload strings (e.g., credential path patterns like `/.aws/credentials`, `gcloud`, `metamask`) using `yara -r shai_hulud.yar ./node_modules/`.

Evidence: Compile the full timeline of the incident for the lessons-learned record: first install timestamp from npm logs, first credential access timestamp from auditd or Sysmon, first outbound exfiltration connection from network logs, and rotation timestamp from cloud provider audit logs. Preserve copies of the four malicious package tarballs (if recoverable from npm cache at `~/.npm/_cacache/`) for internal YARA rule development targeting Shai-Hulud payload signatures. Document which developer workstations and CI runners had the plaintext credential files accessible, as this defines the exact data exposure scope for any required breach notification assessment.

Detection Guidance

Search package manifests and install logs for exact matches on: chalk-tempalte (note the transposed letters vs legitimate chalk-template), @deadcode09284814/axios-util, axois-utils, color-style-utils. In SIEM or EDR, query for Node.js processes (node, npm) making outbound connections to non-allowlisted IPs or domains at install time. Flag file access events targeting credential paths: ~/.aws/credentials, ~/.azure/, ~/.config/gcloud/application_default_credentials.json, and common cryptocurrency wallet paths. Alert on JavaScript processes spawning child processes (sh, bash, cmd) unexpectedly. For DDoS botnet indicators, look for Node.js processes generating high-volume outbound UDP or TCP traffic to rotating IP ranges. IOC-level indicators (specific C2 domains, hashes) were not confirmed in session-accessible sources; consult the Microsoft and Huntress advisories linked in sources for artifact-level IOCs.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`chalk-tempalte (npm package name)`	Typosquatted npm package carrying Shai-Hulud infostealer variant	MEDIUM
DOMAIN	`@deadcode09284814/axios-util (npm package name)`	Typosquatted npm package carrying Shai-Hulud infostealer variant	MEDIUM
DOMAIN	`axois-utils (npm package name)`	Typosquatted npm package carrying Shai-Hulud infostealer variant	MEDIUM
DOMAIN	`color-style-utils (npm package name)`	Typosquatted npm package carrying Shai-Hulud infostealer variant with DDoS botnet module	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1567.001** — Exfiltration to Code Repository
- **T1552.001** — Credentials In Files
- **T1588.001** — Malware

- **T1027** — Obfuscated Files or Information
- **T1136.003** — Cloud Account
- **T1204.002** — Malicious File
- **T1498** — Network Denial of Service
- **T1567** — Exfiltration Over Web Service
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1071.001** — Web Protocols
- **T1059.007** — JavaScript

NIST-800-53R5

- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **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

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication
- **164.312(e)(1)** — Transmission Security

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

ISO-27001-2022

- **A.5.21** — Managing information security in the ICT supply chain
- **A.5.23** — Information security for use of cloud services

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1567.001	Exfiltration to Code Repository	Exfiltration
T1552.001	Credentials In Files	Credential-Access
T1588.001	Malware	Resource-Development
T1027	Obfuscated Files or Information	Defense-Evasion
T1136.003	Cloud Account	Persistence
T1204.002	Malicious File	Execution
T1498	Network Denial of Service	Impact
T1567	Exfiltration Over Web Service	Exfiltration
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1071.001	Web Protocols	Command-And-Control
T1059.007	JavaScript	Execution

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/leaked-shai-hulud-ma...	T3
Mitigating the Axios npm supply chain compromise - Microsoft	https://www.microsoft.com/en-us/security/blog/2026/04/01/mitigating...	T1
Supply Chain Compromise of axios npm Package - Huntress	https://www.huntress.com/blog/supply-chain-compromise-axios-npm-pac...	T3
Axios NPM Package Compromised: Supply Chain Attack Hits ...	https://www.trendmicro.com/en_us/research/26/c/axios-npm-package-co...	T3
Axios 1.4.1 and 0.30.4 NPM packages compromised - Cloudsmith	https://cloudsmith.com/blog/axios-npm-attack-response	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 2026-05-18 18:49 UTC by TJS Security Command Center