INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-01 06:05 UTC

# Claude Code npm Package Exposes Source Architecture; Trojanized Axios RAT Compounds Supply Chain Risk

**THREAT CAMPAIGN** | **CRITICAL** | CVSS 9.5

| | |
|---|---|
| **SCC Item ID** | SCC-CAM-2026-0132 |
| **Type** | Threat Campaign |
| **Severity** | CRITICAL |
| **CVSS Base Score** | 9.5 |
| **Affected Products** | Anthropic Claude Code npm v2.1.88; Axios HTTP client (npm, version affected during 2026-03-31 00:21-03:29 UTC window) |
| **Published** | 2026-04-01T02:12:00 |
| **Discovery Source** | Rss |

## Executive Summary

Two supply chain incidents converged within 48 hours to create compounding risk for organizations running Anthropic's Claude Code npm package. The first incident, an Anthropic packaging error in v2.1.88, exposed approximately 512,000 lines of internal TypeScript source code via embedded source maps, disclosing internal architecture to any consumer of the package. The second and more severe incident involved a trojanized Axios HTTP client dependency, reportedly delivered through a hijacked npm maintainer account during a roughly three-hour window on March 31, 2026 (per T3 reports pending official confirmation); based on available reporting, developers who updated Claude Code during that window may have received a cross-platform remote access trojan, giving attackers potential persistent access to build environments, credentials, and potentially production secrets. **Severity Note:** All current sources are T3 (secondary news and community reports). Details are subject to change pending official Anthropic and Axios maintainer security advisories.

## Technical Analysis

Two discrete supply chain events share a common delivery vector: the npm ecosystem without package integrity verification.

**Event 1, Source Map Exposure (Anthropic Claude Code npm v2.1.88):**

Anthropic's build pipeline inadvertently bundled TypeScript source maps (~60 MB) into the published npm package, exposing roughly 2,000 internal TypeScript files and over 512,000 lines of source code. This constitutes an intellectual property disclosure and security posture leak. Attackers with access to this source can conduct targeted exploitation research against Claude Code's internal logic. No active code execution is involved in this event. CWE-312 (Cleartext Storage of Sensitive Information) applies to any secrets or implementation details surfaced in the maps.

**Event 2, Trojanized Axios Dependency (RAT implant, reported 2026-03-31 00:21-03:29 UTC):**

According to available T3 reporting, a threat actor compromised an Axios npm package maintainer account and published a malicious Axios release containing a cross-platform RAT. Any Claude Code installation updated during the reported ~3-hour exposure window would have pulled this version as a transitive dependency. The RAT provides persistent remote access capabilities and likely supports credential harvesting (CWE-312, T1056), command execution (T1059), and C2 communication over standard protocols (T1071.001). Post-compromise, cloud service discovery (T1526) and unsecured credential access (T1552) are likely follow-on objectives against developer and pipeline environments. Official Axios maintainer confirmation and detailed IOC data are pending.

**Relevant CWEs:** CWE-494 (Download of Code Without Integrity Check), CWE-829 (Inclusion of Functionality from Untrusted Control Sphere), CWE-1357 (Reliance on Insufficiently Trustworthy Component), CWE-312 (Cleartext Storage of Sensitive Information).

**MITRE ATT&CK:** T1195.001 (Compromise Software Dependencies and Development Tools), T1554 (Compromise Host Software Binary), T1059 (Command and Scripting Interpreter), T1071.001 (Application Layer Protocol: Web Protocols), T1056 (Input Capture), T1552 (Unsecured Credentials), T1526 (Cloud Service Discovery), T1546 (Event Triggered Execution), T1566 (Phishing / Dependency Confusion Social Engineering), T1608.001 (Stage Capabilities: Upload Malware).

**Patch/Remediation Status:** Axios maintainers have removed the malicious release; organizations must verify the installed Axios version does not fall within the reported compromised window. Anthropic has acknowledged the source map exposure; verify your installed Claude Code version and check the Anthropic GitHub issue tracker for remediation confirmation. No CVE IDs are currently assigned to either event per available source data. Official advisories from Axios and Anthropic are expected and should be consulted before finalizing remediation decisions.

**Source quality note:** All current sources are T3 (community reporting, secondary news). Confirm details against Anthropic's official advisory and the Axios npm security advisory when available. This assessment may be updated as official guidance is published.

## Action Checklist

1. Step 1: Containment. Immediately identify all developer workstations and CI/CD pipeline nodes where Claude Code npm was installed or updated between 2026-03-30 and 2026-03-31 03:29 UTC. Isolate any system confirmed to have pulled an Axios version published during the 00:21-03:29 UTC window on 2026-03-31. Suspend automated pipeline jobs that run on potentially affected nodes pending verification. Do not execute new builds on unverified hosts.
2. Step 2: Detection. Audit installed Axios versions across all affected environments: run 'npm list axios' or 'cat node_modules/axios/package.json' and compare the 'version' and 'integrity' fields against the known-good Axios release hashes from the official npm registry. Check npm audit logs and

package-lock.json history for Axios version changes occurring in the UTC window. On affected hosts, look for anomalous outbound network connections from node processes (unexpected IPs/domains, beaconing patterns), new scheduled tasks or cron entries, and persistence mechanisms consistent with T1546 (Event Triggered Execution). Review CI/CD runner logs for unexpected process spawning or network activity during that window.

3. Step 3: Eradication. On hosts confirmed to have received the trojanized Axios release AND showing active compromise indicators (T1059 process spawning, T1546 persistence artifacts, network beaconing), treat as fully compromised and perform clean OS reimaging. For hosts with trojanized Axios but no active compromise indicators detected, isolate pending forensic investigation before committing to reimaging. Update Claude Code to the latest verified version after confirming Anthropic has issued a clean build without embedded source maps. Pin Axios to a verified clean release with a known-good integrity hash from the npm registry. Purge npm cache on all developer machines ('npm cache clean --force') to prevent reinstallation from cached malicious packages.

4. Step 4: Recovery. After reimaging affected hosts, validate the restored environment: run 'npm audit' and confirm zero high/critical findings. Verify Axios and Claude Code package integrity hashes against npm registry records. Rotate all credentials, API keys, tokens, and secrets accessible from affected developer workstations and CI/CD environments. Prioritize: (1) cloud provider credentials and API keys, (2) code signing and artifact signing keys, (3) VCS tokens (GitHub, GitLab), (4) CI/CD pipeline secrets, (5) third-party service tokens (SaaS, container registries). Consider phased rotation to minimize pipeline disruption, completing high-priority credentials within 24 hours. Monitor for re-compromise indicators for a minimum of 30 days post-remediation using EDR telemetry and network flow analysis.

5. Step 5: Post-Incident. This event exposed two control gaps: absence of npm package integrity verification (Subresource Integrity / lockfile enforcement) and lack of transitive dependency monitoring. Remediate by enforcing 'npm ci' with committed lockfiles in all pipelines, implementing a private npm proxy (e.g., Artifactory, Verdaccio) with mandatory integrity checks, and deploying a software composition analysis (SCA) tool to monitor transitive dependencies continuously. Review your CI/CD pipeline trust boundaries and apply least-privilege to pipeline runner credentials. Document this incident as a case study for your third-party component risk policy (maps to NIST SP 800-161 supply chain risk management).

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | IMMEDIATE |
| **Escalation Criteria** | Escalate to CISO and legal counsel immediately if any evidence of successful RAT C2 communication, credential exfiltration, or code signing key access is confirmed, or if affected CI/CD pipelines had access to production environment credentials, customer data repositories, or regulated data (PII/PHI/PCI) — the latter triggers breach notification obligations under applicable data protection regulations. |

| | |
|---|---|
| **Recovery Notes** | Post-containment recovery must prioritize credential rotation before restoring any CI/CD pipeline to active status, as the trojanized Axios RAT had HTTP client access to all environment variables and secrets accessible during the compromise window — any unrotated credential should be treated as fully compromised. Verify Claude Code is updated to a clean version explicitly confirmed by Anthropic to exclude embedded source maps, and validate via 'find node_modules/@anthropic-ai/claude-code -name "*.js.map"' returning zero results. Maintain enhanced monitoring of all developer workstations and CI/CD nodes for 30 days minimum, specifically watching for MITRE ATT&CK T1546 (Event Triggered Execution) persistence indicators and anomalous outbound connections from node processes to non-npm-registry, non-Anthropic infrastructure. |
| **Forensic Artifacts** | npm cache tarball for malicious Axios version at '~/.npm/_cacache/' (Linux/macOS) or '%AppData%\npm-cache_cacache\' (Windows) — contains the trojanized package tarball with embedded RAT payload, preserving the exact malicious build for payload analysis and IOC extraction even after node_modules deletion \| package-lock.json git history ('git log -p package-lock.json') showing the '_resolved' URL and '_integrity' SHA-512 hash for Axios as it existed during the 2026-03-31 00:21–03:29 UTC compromise window — the resolved URL will point to the malicious tarball rather than the legitimate npmjs.org CDN path \| Sysmon Event ID 1 (Process Creation) and Event ID 3 (Network Connection) logs on affected hosts filtered for node.exe as parent process — the Axios RAT, executing within the Node.js runtime, would generate child process spawns and outbound HTTP/HTTPS connections to C2 infrastructure that are directly attributable to the trojanized package's postinstall or runtime hooks \| CI/CD pipeline job execution logs (GitHub Actions workflow run artifacts, GitLab CI job traces, Jenkins build logs) from the 2026-03-30 to 2026-03-31 03:29 UTC window showing the full npm install output including resolved Axios tarball URL, confirming which pipeline runs fetched the malicious package and what downstream build artifacts or deployments those runs produced \| Process memory dump of any running node process that imported the trojanized Axios module — captured via 'gcore ' (Linux) or Windows Task Manager dump — containing the decoded in-memory RAT payload, C2 configuration, and any credential or environment variable material staged for exfiltration that would not persist to disk |

### Per-Action IR Details

**Step 1: Containment — Immediately identify all developer workstations and CI/CD pipeline nodes where Claude Code npm was installed or updated between 2026-03-30 and 2026-03-31 03:29 UTC. Isolate any system confirmed to have pulled an Axios version published during the 00:21–03:29 UTC window on 2026-03-31. Suspend automated pipeline jobs that run on potentially affected nodes pending verification. Do not execute new builds on unverified hosts.**

**NIST Phase:** Containment

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

**Controls:** NIST IR-4 (Incident Handling), NIST CM-3 (Configuration Change Control), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 4.6 (Securely Manage Enterprise Assets and Software)

**Compensating:** On each suspected developer workstation, run: 'npm list axios --depth=0' and cross-reference install timestamps via 'stat node_modules/axios/package.json' (Linux/macOS) or 'Get-Item node_modules\axios\package.json | Select-Object LastWriteTime' (PowerShell). For CI/CD nodes, parse runner job logs (GitHub Actions: .github/workflows run logs; GitLab CI: job artifacts) for any npm install/ci invocations timestamped between 2026-03-31 00:21–03:29 UTC. Use osquery query 'SELECT * FROM npm_packages WHERE name="axios";' to enumerate installed versions fleet-wide without requiring a SIEM.

**Evidence:** Before isolating hosts, capture: (1) full output of 'npm list --all --json > npm_tree_$(hostname)_$(date +%s).json' to preserve the entire dependency tree including the trojanized Axios version string and resolved tarball URL; (2) 'cat node_modules/axios/package.json' to record the exact version, _resolved, and _integrity fields that will

confirm whether the 00:21–03:29 UTC malicious publish was fetched; (3) network connection state via 'ss -tnp' or 'netstat -anb' to capture any active outbound connections from node processes at time of isolation — the RAT payload embedded in Axios would likely beacon immediately upon require(); (4) running process list ('ps aux' or 'Get-Process') filtered for node/npm processes to identify any persistent RAT processes already executing.

**Step 2: Detection — Audit installed Axios versions across all affected environments: run 'npm list axios' or 'cat node_modules/axios/package.json' and compare the 'version' and 'integrity' fields against the known-good Axios release hashes from the official npm registry. Check npm audit logs and package-lock.json history for Axios version changes occurring in the UTC window. On affected hosts, look for anomalous outbound network connections from node processes (unexpected IPs/domains, beaconing patterns), new scheduled tasks or cron entries, and persistence mechanisms consistent with T1546 (Event Triggered Execution). Review CI/CD runner logs for unexpected process spawning or network activity during that window.**

**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 IR-5 (Incident Monitoring), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 8.2 (Collect Audit Logs)

**Compensating:** Integrity verification without a SIEM: (1) Pull the known-good Axios integrity hash from 'https://registry.npmjs.org/axios' (the 'dist.integrity' field for the legitimate release) and compare against 'cat node_modules/axios/package.json | grep _integrity' on each host — any SHA-512 mismatch confirms the trojanized build. (2) Deploy a Sigma rule targeting process creation events where parent process is 'node' and child is an unexpected shell (bash, sh, cmd.exe, powershell.exe) — MITRE ATT&CK T1059 (Command and Scripting Interpreter) — using Sysmon Event ID 1 (Process Creation) with ParentImage matching node.exe. (3) For network beaconing detection without EDR, run Wireshark or tcpdump on the host NIC for 15 minutes post-isolation replay: 'tcpdump -i any -w axios_rat_capture.pcap host ' and inspect for periodic outbound connections to non-CDN, non-npm-registry IPs. (4) Check cron persistence via 'crontab -l && ls -la /etc/cron.*/' and Windows scheduled tasks via 'schtasks /query /fo LIST /v | findstr /i "node npm axios"'.

**Evidence:** Forensic artifacts specific to a trojanized npm package RAT: (1) 'package-lock.json' and 'npm-shrinkwrap.json' diff history in git — 'git log -p package-lock.json | grep -A5 -B5 axios' will show the exact commit that introduced the malicious resolved URL and SHA integrity value, timestamped to the 00:21–03:29 UTC window; (2) npm cache directory contents at '~/.npm/_cacache/' (Linux/macOS) or '%AppData%\npm-cache_cacache\' (Windows) — the trojanized tarball will be cached here with its malicious SHA-512 hash, preserving the payload even after node_modules deletion; (3) Sysmon Event ID 3 (Network Connection) and Event ID 1 (Process Creation) logs filtered for node.exe spawning child processes or making outbound connections to non-Anthropic, non-npm infrastructure — indicative of RAT C2 beaconing via the hijacked Axios HTTP client; (4) CI/CD pipeline artifact logs (GitHub Actions workflow run logs, GitLab CI job traces) showing the exact npm install command output including resolved Axios tarball URL during the compromise window.

**Step 3: Eradication — On hosts confirmed to have received the trojanized Axios release, treat the host as fully compromised: perform clean OS reimaging rather than in-place remediation. Update Claude Code to the latest verified version after confirming Anthropic has issued a clean build without embedded source maps. Pin Axios to a verified clean release with a known-good integrity hash from the npm registry. Purge npm cache on all developer machines ('npm cache clean --force') to prevent reinstallation from cached malicious packages.**

**NIST Phase:** Eradication

**Reference:** NIST 800-61r3 §3.4 — Eradication

**Controls:** NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST CM-3 (Configuration Change Control), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 7.4 (Perform Automated Application Patch Management)

**Compensating:** Before reimaging, use ClamAV with a custom YARA rule targeting the RAT payload signature embedded in the trojanized Axios package to confirm infection scope: write a YARA rule matching known malicious strings or byte patterns from the Axios RAT (extracted during analysis of the malicious tarball from npm cache) and run 'clamscan --database=/path/to/custom_axios_rat.yar -r /home /opt /var'. For the npm cache purge verification, confirm cache is empty post-purge: 'npm cache verify' should return zero cached packages. When pinning Axios in package.json, use the exact SRI hash format: add '"axios": "npm:axios@"' with the lockfile integrity field set to the SHA-512 from 'https://registry.npmjs.org/axios/' — do not rely on version strings alone.

**Evidence:** Capture before reimaging: (1) Full memory dump of any node process that loaded the trojanized Axios module using 'gcore ' (Linux) or Task Manager > 'Create dump file' (Windows) — the RAT's in-memory payload, decoded C2 configuration, and any exfiltrated credential material may only exist in RAM; (2) Copy the malicious Axios tarball from npm cache ('~/.npm/_cacache/') before purging — this is the primary forensic artifact confirming the exact malicious build and enables payload analysis and IOC extraction; (3) Enumerate all environment variables accessible to node processes via '/proc//environ' (Linux) or process environment block dump — the Claude Code source map exposure in v2.1.88 combined with RAT access means internal API keys, tokens, and architecture details in process environment may have been exfiltrated; (4) Image the disk before reimaging using 'dd' or 'dc3dd' to preserve the full filesystem timeline for post-incident forensic review and potential regulatory evidence retention.

**Step 4: Recovery — After reimaging affected hosts, validate the restored environment: run 'npm audit' and confirm zero high/critical findings. Verify Axios and Claude Code package integrity hashes against npm registry records. Rotate all credentials, API keys, tokens, and secrets accessible from affected developer workstations and CI/CD environments, including cloud provider credentials, code signing keys, and pipeline secrets. Monitor for re-compromise indicators for a minimum of 30 days post-remediation using EDR telemetry and network flow analysis.**

**NIST Phase:** Recovery

**Reference:** NIST 800-61r3 §3.5 — Recovery

**Controls:** NIST IR-4 (Incident Handling), NIST IA-5 (Authenticator Management), NIST AU-11 (Audit Record Retention), CIS 5.2 (Use Unique Passwords), CIS 6.2 (Establish an Access Revoking Process), CIS 6.5 (Require MFA for Administrative Access)

**Compensating:** Credential rotation verification without a PAM tool: (1) For GitHub/GitLab CI pipeline secrets, use the respective API to list all repository and organization-level secrets ('gh secret list --repo ') and confirm each was rotated post-incident by checking creation timestamps; (2) For AWS credentials accessible from compromised CI/CD runners, run 'aws iam list-access-keys --user-name ' and verify no access keys predate the rotation event; (3) Establish a 30-day re-compromise monitoring baseline using osquery scheduled queries: 'SELECT * FROM npm_packages WHERE name="axios" AND version NOT IN ("");' running every 6 hours to alert on any unauthorized Axios version drift. (4) Use auditd (Linux) or Windows Security Event Log Event ID 4698/4702 (Scheduled Task Created/Modified) to detect persistence re-establishment attempts post-recovery.

**Evidence:** Post-recovery validation artifacts to preserve: (1) Output of 'npm audit --json > post_recovery_audit_$(date +%s).json' for each restored host — provides a timestamped baseline confirming zero vulnerabilities at recovery completion, serving as evidence of due care; (2) Git diff of package-lock.json before compromise vs. post-recovery pinned state ('git diff HEAD -- package-lock.json') documenting exactly which Axios resolved URL and integrity hash changed — critical for supply chain incident reporting; (3) Cloud provider credential rotation audit logs (AWS CloudTrail CreateAccessKey/DeleteAccessKey events; GCP Admin Activity logs for ServiceAccount key operations) confirming all pipeline credentials were rotated within the incident response window; (4) CI/CD pipeline run logs for the first three clean builds post-recovery, confirming npm install resolves only to the pinned clean Axios version and that 'npm ci' produces no integrity failures.

**Step 5: Post-Incident — This event exposed two control gaps: absence of npm package integrity verification (Subresource Integrity / lockfile enforcement) and lack of transitive dependency monitoring. Remediate by enforcing 'npm ci' with committed lockfiles in all pipelines, implementing a private npm proxy (e.g., Artifactory, Verdaccio) with mandatory integrity checks, and deploying a software composition analysis (SCA) tool to monitor transitive dependencies continuously. Review your CI/CD pipeline trust boundaries and apply**

**least-privilege to pipeline runner credentials. Document this incident as a case study for your third-party component risk policy (maps to NIST SP 800-161 supply chain risk management).**

**NIST Phase:** Post Incident

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

**Controls:** NIST SI-7 (Software, Firmware, and Information Integrity), NIST SI-2 (Flaw Remediation), NIST IR-8 (Incident Response Plan), NIST SA-12 (Supply Chain Protection), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

**Compensating:** Deploy Verdaccio (free, open-source npm proxy) as a local registry mirror: configure '.npmrc' with 'registry=http://localhost:4873' and set Verdaccio's uplink to only cache packages after integrity verification against npmjs.org. For transitive dependency monitoring without a commercial SCA tool, implement a GitHub Actions or GitLab CI step using 'npm audit --audit-level=high --json' combined with 'node --experimental-vm-modules' to run a custom script that compares all resolved package SRI hashes in package-lock.json against the npm registry API on every PR. For the Claude Code v2.1.88 source map exposure specifically, add a CI step: 'find node_modules/@anthropic-ai/claude-code -name "*.js.map" | wc -l' and fail the build if source map files are present in the installed package — this detects any future recurrence of embedded source maps exposing Anthropic's internal TypeScript architecture.

**Evidence:** Post-incident documentation artifacts: (1) Lessons-learned report documenting the dual failure mode — Anthropic packaging error exposing 512,000 lines of internal TypeScript via source maps in v2.1.88 AND the trojanized Axios RAT — as a compounding supply chain risk scenario, with timeline reconstructed from npm registry publish timestamps and git commit history; (2) Dependency graph snapshot showing the transitive path from Claude Code → Axios to document how a transitive dependency compromise propagates through the supply chain — export via 'npm ls --json > full_dep_graph.json'; (3) Updated threat model documenting npm maintainer account hijacking (the Axios attack vector) as an explicit threat scenario for all third-party packages in your CI/CD dependency graph, mapped to MITRE ATT&CK T1195.001 (Supply Chain Compromise: Compromise Software Dependencies and Development Tools); (4) Evidence of SCA tool deployment and first scan results confirming Axios and Claude Code are pinned to verified clean versions, retained for audit purposes under NIST AU-11 (Audit Record Retention).

## Detection Guidance

**Axios Version Check (all affected environments):**

Run: npm list axios --depth=0

Compare the resolved version and its integrity hash in package-lock.json against official npm registry records. The reported malicious release was published during 2026-03-31 00:21-03:29 UTC (per T3 reporting); any Axios version resolved during that window warrants investigation.

**npm Audit Log Review:**

Check ~/.npm/_logs/ for install activity during the exposure window. CI/CD logs should show the resolved Axios version and its download timestamp.

**Host-Based Indicators:**

Look for node.js or npm processes spawning unexpected child processes (cmd.exe, /bin/sh, PowerShell), consistent with T1059 RAT execution. Look for new persistence entries: Windows scheduled tasks or registry run keys created by node processes; Unix cron entries or systemd unit files added post-install.

**Network Indicators:**

Monitor for anomalous outbound connections from developer workstations or CI/CD runners to unfamiliar IPs or domains over HTTP/S (T1071.001 C2 beaconing). Look for regular, low-volume beaconing intervals (common

RAT behavior). DNS queries for domains not in your known-good baseline from node processes are a high-priority signal.

**Source Map Exposure, Scoping Check:**

Run: ls -lh node_modules/@anthropic-ai/claude-code/ and check for .map files or a /src directory. The presence of TypeScript source maps in v2.1.88 is confirmed; this is an exposure indicator, not active compromise.

**IOC Note:** Specific RAT C2 infrastructure IOCs (IPs, domains, file hashes for the malicious Axios payload) have not been confirmed in available T3 sources at this time. Request IOC feed from Endor Labs or monitor npm security advisories and Axios GitHub releases as they are published for confirmed hashes and C2 infrastructure. Subscribe to Anthropic and Axios security advisories for real-time updates.

## Indicators of Compromise

| Type | Value | Context | Confidence |
|---|---|---|---|
| PACKAGE | `@anthropic-ai/claude-code v2.1.88` | npm package containing inadvertently embedded TypeScript source maps (~60 MB, ~512,000 lines). Confirms source architecture exposure. No RAT in this artifact — intellectual property and security posture risk only. | HIGH |
| PACKAGE | `axios (npm) — version published 2026-03-31 00:21–03:29 UTC` | Trojanized Axios release containing cross-platform RAT, delivered via hijacked npm maintainer account. Specific malicious version number and file hash not confirmed in available sources at time of writing. Treat any Axios version resolved during this UTC window as suspect. | MEDIUM |
| BEHAVIORAL | `node.js process spawning shell interpreter (cmd.exe, powershell.exe, /bin/bash, /bin/sh) outside expected build context` | Node.js process leveraged via trojanized Axios dependency injected into supply chain to spawn shell interpreters for remote command execution and system compromise on developer workstations and CI/CD infrastructure. | MEDIUM |
| BEHAVIORAL | `Anomalous outbound HTTP/S beaconing from node.js or CI/CD runner processes to non-baseline destinations` | Consistent with RAT C2 communication (T1071.001). Investigate any regular-interval low-volume connections established post-Axios update. | MEDIUM |
| FILE | `Presence of .map files or /src TypeScript directory within node_modules/@anthropic-ai/claude-code/` | Confirms Claude Code v2.1.88 source map exposure. Not an active compromise indicator — relevant for scoping the intellectual property and architecture disclosure risk. | HIGH |

## Framework Mappings

### MITRE-ATTACK

- **T1566** — Phishing
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1546** — Event Triggered Execution
- **T1071.001** — Web Protocols
- **T1554** — Compromise Host Software Binary
- **T1056** — Input Capture
- **T1059** — Command and Scripting Interpreter
- **T1552** — Unsecured Credentials
- **T1608.001** — Upload Malware
- **T1526** — Cloud Service Discovery

### NIST-800-53R5

- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **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
- **15.1** — Establish and Maintain an Inventory of Service Providers

### 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

### SOC2-TSC

- **CC9.2** — Manages risks associated with vendors and business partners

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1566 | Phishing | Initial-Access |
| T1195.001 | Compromise Software Dependencies and Development Tools | Initial-Access |
| T1546 | Event Triggered Execution | Privilege-Escalation |
| T1071.001 | Web Protocols | Command-And-Control |
| T1554 | Compromise Host Software Binary | Persistence |
| T1056 | Input Capture | Collection |
| T1059 | Command and Scripting Interpreter | Execution |
| T1552 | Unsecured Credentials | Credential-Access |
| T1608.001 | Upload Malware | Resource-Development |
| T1526 | Cloud Service Discovery | Discovery |

## Sources

| Source | URL | Tier |
|---|---|---|
| **Security News** | https://thehackernews.com/2026/04/claude-code-tleaked-via-npm-packa... | T3 |
| **Security vulnerabilitty found in axios. Is claude code affected? #41547** | https://github.com/anthropics/claude-code/issues/41547 | T3 |
| **Axios compromised: hijacked maintainer account pushes malicious ...** | https://www.endorlabs.com/learn/npm-axios-compromise | T3 |
| **Axios npm package hit in supply chain attack that deployed a cross ...** | https://www.tomshardware.com/tech-industry/cyber-security/axios-npm... | T3 |
| **Anthropic Re-Leaks Claude Code Source via 60 MB npm Map File** | https://www.reddit.com/r/cybersecurity/comments/1s8nll9/anthropic_r... | T3 |

Generated 2026-04-01 06:05 UTC by TJS Security Command Center