INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-05 05:56 UTC

UNC1069 Targets npm Ecosystem: North Korean Actors Weaponize ClickFix Playbook Against High-Impact Open Source Maintainers

THREAT CAMPAIGN | **CRITICAL** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0149
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.5
Affected Products	Axios npm package versions 1.14.1 and 0.30.4; downstream Node.js ecosystem dependents; macOS, Windows, Linux
Published	2026-04-04T16:30:42
Discovery Source	Rss

Executive Summary

North Korean threat actor UNC1069 compromised the Axios npm package, a JavaScript HTTP library with billions of weekly downloads, by social engineering a maintainer into installing malware via a fake Microsoft Teams prompt. Two malicious versions (1.14.1 and 0.30.4) were published to the npm registry and remained live for approximately three hours, sufficient time for any organization running automated dependency pulls to potentially introduce the trojanized package into their environment. Downstream compromise requires execution of the embedded RAT payload, which has not been publicly confirmed at scale.

Technical Analysis

UNC1069 (North Korea-nexus, tracked by Google Threat Intelligence) executed a supply chain compromise against the Axios npm package using a ClickFix-style social engineering lure. The attack chain: (1) A maintainer was targeted via spearphishing (T1566.002) with a fake Microsoft Teams update error prompt, leading to execution of a RAT installer (T1204.002, T1219). (2) With maintainer credentials and likely session tokens compromised (T1539, T1078), the actor published two trojanized versions to the npm registry: 1.14.1 and 0.30.4. (3) Both versions introduced a malicious dependency, 'plain-crypto-js', which functioned as a dropper (CWE-829, CWE-494) for cross-platform malware targeting macOS, Windows, and Linux. (4) The malicious packages bypassed origin validation controls (CWE-346) and represent a supply chain compromise of the software build pipeline (T1195.002, CWE-1395, CWE-693). No CVE had been assigned as of campaign

publication; this is a supply chain campaign, not a library vulnerability. Affected versions are exclusively 1.14.1 and 0.30.4. The malicious packages were removed from the registry after approximately three hours. Microsoft has published vendor mitigation guidance. Source quality score for this item is 0.73; the Microsoft blog (Tier 1) is the authoritative source for mitigation specifics.

Action Checklist

1. Step 1: Containment. Immediately audit all Node.js application dependency lock files (package-lock.json, yarn.lock, pnpm-lock.yaml) across your environment for Axios versions 1.14.1 or 0.30.4. Block installation of these versions at your artifact proxy or registry mirror (Nexus, Artifactory, Verdaccio). Quarantine any build pipeline, container image, or deployed artifact that resolved either version during the exposure window. Reference: Microsoft Security Blog (2026-04-01).
2. Step 2: Detection. Search SIEM and EDR telemetry for the string 'plain-crypto-js' in process execution logs, npm install output logs, and network traffic. On macOS and Linux, query for unexpected shell interpreter spawns (T1059.004) originating from Node.js processes. On Windows, look for PowerShell or cmd.exe child processes spawned by node.exe. Check npm audit logs and CI/CD pipeline logs for installs of Axios 1.14.1 or 0.30.4 occurring between the package publish timestamp and removal. Review endpoint telemetry for outbound connections to unknown hosts initiated by Node.js processes.
3. Step 3: Eradication. Downgrade Axios to the last known-good version. Verify the current stable release at npmjs.com/package/axios before pinning; as of campaign date 1.7.9 was stable but confirm at time of remediation. Remove the 'plain-crypto-js' package from all dependency trees. Rebuild all container images and application artifacts from clean base states. Rotate npm publish credentials and MFA tokens for any maintainer accounts in your organization. Invalidate any session tokens that may have been exposed on compromised developer workstations.
4. Step 4: Recovery. After redeployment, verify lock files reflect the pinned clean Axios version. Run 'npm audit' against all application dependency trees and confirm no residual references to 'plain-crypto-js' or the malicious versions. Monitor outbound network traffic from previously affected systems for 30 days for signs of RAT callback activity. Review EDR alerts on developer workstations that may have received the ClickFix lure for persistence mechanisms (scheduled tasks, launch agents, cron jobs).
5. Step 5: Post-Incident. This campaign exposes three control gaps: (a) absence of package integrity verification in CI/CD pipelines (implement Sigstore or npm provenance attestation per SLSA framework guidance at https://slsa.dev); (b) lack of registry mirroring with allowlist controls, allowing direct pulls from the public npm registry; (c) insufficient phishing-resistant MFA on npm publish accounts (enforce hardware key or passkey authentication for all package publish credentials). Map findings to NIST CSF Protect function controls PR.DS-6 (integrity checking) and PR.AC-1 (identity management). Schedule a supply chain risk review against NIST SP 800-161 Revision 1 (2022); see https://csrc.nist.gov/publications/detail/sp/800-161/rev-1.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, legal, and executive leadership immediately if any evidence confirms that Axios 1.14.1 or 0.30.4 was installed in a production environment serving external users or processing PII, PHI, or payment data, as this may trigger breach notification obligations under GDPR, CCPA, or HIPAA; also escalate if the compromised maintainer workstation shows evidence of UNC1069 RAT persistence beyond the initial ClickFix lure, indicating active hands-on-keyboard intrusion rather than an automated supply chain hit.
Recovery Notes	After redeploying all applications with Axios pinned to 1.7.9 and lock files verified, maintain elevated monitoring on outbound network connections from all Node.js application hosts and developer workstations for a minimum of 30 days, specifically watching for periodic beacon-style HTTPS connections to external hosts that were not present in pre-incident baselines — UNC1069 RATs associated with North Korean supply chain campaigns have historically used scheduled callback intervals. Verify that all npm publish credentials and session tokens rotated in Step 3 show no subsequent unauthorized use by monitoring npm's access log for your organization's packages. Conduct a tabletop exercise within 14 days specifically simulating the ClickFix social engineering vector against your developer population to measure susceptibility and validate that your phishing-resistant MFA rollout covers all package publish accounts.
Forensic Artifacts	npm cache tarballs (~/.npm/_cacache/content-v2/ on Linux/macOS, %APPDATA%\npm-cache_cacache\content-v2\ on Windows): The malicious axios@1.14.1 and axios@0.30.4 package tarballs and the 'plain-crypto-js' dependency tarball may persist in the npm content-addressable cache even after registry removal — extract, hash with sha256sum, and preserve as primary malware samples for IOC development. | package-lock.json and yarn.lock integrity hashes from affected repositories: These lock files record the exact sha512 integrity hash that npm verified against the downloaded tarball — the hash for axios@1.14.1 or 0.30.4 in any lock file definitively proves the malicious version was resolved and installed, serving as chain-of-custody evidence for the exposure scope determination. | Compromised maintainer workstation browser profile (Chrome/Firefox/Safari download history, cache, and session cookies): The ClickFix lure was delivered via a fake Microsoft Teams prompt — the delivery URL, the downloaded payload (likely a .dmg, .pkg, or .exe disguised as a Teams installer), and any harvested npm session cookies would be recoverable from the browser profile and OS download history at ~/Downloads/ or %USERPROFILE%\Downloads\. | macOS LaunchAgents and Windows Run registry keys post-ClickFix: UNC1069's RAT persistence after the ClickFix lure execution would manifest as a new .plist file in ~/Library/LaunchAgents/ (macOS) or a new HKCU\Software\Microsoft\Windows\CurrentVersion\Run entry (Windows) — collect these with their creation timestamps and the binary or script they reference, as these are the primary persistence indicators for North Korean ClickFix-delivered implants. | CI/CD pipeline execution logs with timestamps bracketing the npm registry exposure window: Build logs from GitHub Actions, Jenkins, GitLab CI, or equivalent systems that show npm install invocations between the package publish time and removal time are the definitive record of which pipelines resolved the malicious versions — these logs must be archived in write-protected storage immediately to prevent tampering and to support the downstream exposure scope analysis across all application artifacts produced during that window.

Per-Action IR Details

Step 1: Containment — Immediately audit all Node.js application dependency lock files (package-lock.json, yarn.lock, pnpm-lock.yaml) across your environment for Axios versions 1.14.1 or 0.30.4. Block installation of these versions at your artifact proxy or registry mirror (Nexus, Artifactory, Verdaccio). Quarantine any build pipeline, container image, or deployed artifact that resolved either version during the exposure window.

Reference: Microsoft Security Blog (2026-04-01).

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-2 (Baseline Configuration), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without a registry proxy, run the following across all repos: grep -r '"axios": "1\.14\.1\|0\.30\.4"' . --include='package*.json' and grep -r 'axios@1\.14\.1\|axios@0\.30\.4' . --include='*.lock'. For container environments: docker inspect --format='{{json .Config.Labels}}' $(docker ps -q) and scan image layers with Trivy (free, OSS) using trivy image --severity CRITICAL . Add a deny rule to your /etc/hosts or internal DNS for the UNC1069 C2 infrastructure if IOCs become available. Pin Axios to 1.7.9 immediately in all package.json files using npm install axios@1.7.9 --save-exact.

Evidence: Before quarantining any artifact or pipeline runner, capture: (1) full package-lock.json and yarn.lock snapshots from affected repos — these record the exact resolved version and integrity hash (sha512) of every installed package including Axios 1.14.1 or 0.30.4 and the malicious 'plain-crypto-js' transitive dependency; (2) CI/CD pipeline execution logs timestamped during the npm registry exposure window showing the npm install invocation and resolved dependency tree; (3) Docker image layer digests for any container built during the window (docker image inspect outputs the layer SHA256s needed to reconstruct what was installed); (4) npm cache contents on build runners (~/.npm/_cacache) which may retain the malicious package tarballs even after removal from the registry.

Step 2: Detection — Search SIEM and EDR telemetry for the string 'plain-crypto-js' in process execution logs, npm install output logs, and network traffic. On macOS and Linux, query for unexpected shell interpreter spawns (T1059.004) originating from Node.js processes. On Windows, look for PowerShell or cmd.exe child processes spawned by node.exe. Check npm audit logs and CI/CD pipeline logs for installs of Axios 1.14.1 or 0.30.4 occurring between the package publish timestamp and removal. Review endpoint telemetry for outbound connections to unknown hosts initiated by Node.js processes.

NIST Phase: Detection Analysis

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

Controls: NIST IR-4 (Incident Handling), NIST IR-5 (Incident Monitoring), 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), MITRE ATT&CK T1059.004 (Command and Scripting Interpreter: Unix Shell), MITRE ATT&CK T1195.002 (Supply Chain Compromise: Compromise Software Supply Chain), MITRE ATT&CK T1566.002 (Phishing: Spearphishing Link — ClickFix lure delivery)

Compensating: Without SIEM/EDR, use Sysmon (Windows) with the SwiftOnSecurity Sysmon config to capture Event ID 1 (Process Creation) filtering ParentImage containing 'node.exe' and Image containing 'cmd.exe', 'powershell.exe', or 'wscript.exe'. On macOS/Linux, run: sudo auditctl -a always,exit -F arch=b64 -S execve -k node_child_exec and review /var/log/audit/audit.log for execve calls with ppid matching a node process. Search npm install logs directly: find ~/.npm -name '*.log' -newer | xargs grep -l 'plain-crypto-js'. Use Wireshark or tcpdump to capture 24 hours of egress from developer workstations and CI runners: tcpdump -i eth0 -w capture.pcap 'src port 443 and src host '. Write a YARA rule scanning ~/.node_modules and global npm cache for the 'plain-crypto-js' package.json name field.

Evidence: Before clearing logs or reimaging, preserve: (1) On Windows, export Windows Security Event Log filtering Event ID 4688 (Process Creation) where ParentProcessName contains 'node.exe' — UNC1069's ClickFix payload would spawn a shell from within the Node.js process after the malicious Axios postinstall script executes; (2) On macOS, collect the Unified Log stream for the exposure window: log collect --start '2026-04-01 00:00:00' --output node_unified.logarchive filtering for process 'node' with child spawns; (3) npm debug log at ~/.npm/_logs/ — these verbose logs record every package installed, its resolved version, and the postinstall script execution for 'plain-crypto-js'; (4) Network flow logs or pcap showing outbound connections from node.exe or the CI runner to non-RFC1918 addresses, particularly immediately following the npm install invocation — UNC1069 RAT callback traffic would originate from this process context; (5) The actual malicious package tarball if still present in npm cache (~/.npm/_cacache/content-v2/) — extract and hash for IOC submission.

Step 3: Eradication — Downgrade Axios to the last known-good version (1.7.9 is the current stable as of the campaign; verify against the official npm registry at npmjs.com/package/axios before pinning). Remove the 'plain-crypto-js' package from all dependency trees. Rebuild all container images and application artifacts from clean base states. Rotate npm publish credentials and MFA tokens for any maintainer accounts in your organization. Invalidate any session tokens that may have been exposed on compromised developer workstations.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-3 (Malicious Code Protection), NIST SI-7 (Software, Firmware, and Information Integrity), NIST IA-5 (Authenticator Management), CIS 4.6 (Securely Manage Enterprise Assets and Software), CIS 5.2 (Use Unique Passwords), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.4 (Perform Automated Application Patch Management), MITRE ATT&CK T1195.002 (Supply Chain Compromise: Compromise Software Supply Chain)

Compensating: Without an enterprise secret rotation platform, use the following procedure: (1) Immediately revoke the compromised maintainer's npm token via npmjs.com account settings → Access Tokens → delete all active tokens, then issue a new granular automation token scoped only to required packages; (2) Run npm ls plain-crypto-js --all in every application repo root to enumerate all dependency tree references before removal; (3) After pinning axios@1.7.9, run npm ci (not npm install) to force a clean install from the updated lock file — npm ci deletes node_modules first, preventing residual malicious files; (4) For container rebuilds without CI/CD automation, build with --no-cache flag: docker build --no-cache -t : . to prevent layer cache reuse from the exposure window; (5) On developer workstations suspected of receiving the ClickFix lure, run ClamAV: clamscan -r --bell -i /Users/ (macOS) or clamscan -r --bell -i /home/ (Linux) after updating signatures with freshclam.

Evidence: Before rotating credentials, forensically image or snapshot the compromised maintainer's workstation — this machine received the ClickFix lure (fake Microsoft Teams prompt) and is the UNC1069 initial access point. Capture: (1) browser history and downloads folder for the fake Teams installer or ClickFix payload delivery URL; (2) macOS ~/Library/LaunchAgents/ and ~/Library/Application Support/ for persistence mechanisms planted by the RAT after the ClickFix lure executed; (3) Windows %APPDATA%\Roaming\, %LOCALAPPDATA%\Temp\, and HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry key for UNC1069 persistence artifacts; (4) The npm token file at ~/.npmrc on the maintainer's workstation — this file holds the plaintext publish token that UNC1069 harvested to push the malicious Axios versions; (5) Memory dump of the node.exe or the ClickFix payload process if still running, captured with WinPmem (Windows) or osxpmem (macOS) before any reboot.

Step 4: Recovery — After redeployment, verify lock files reflect the pinned clean Axios version. Run 'npm audit' against all application dependency trees and confirm no residual references to 'plain-crypto-js' or the malicious versions. Monitor outbound network traffic from previously affected systems for 30 days for signs of RAT callback activity. Review EDR alerts on developer workstations that may have received the ClickFix lure for persistence mechanisms (scheduled tasks, launch agents, cron jobs).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST SI-2 (Flaw Remediation), NIST SI-4 (System Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST CP-10 (System Recovery and Reconstitution), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), MITRE ATT&CK T1053.003 (Scheduled Task/Job: Cron — persistence post-ClickFix), MITRE ATT&CK T1543.001 (Create or Modify System Process: Launch Agent — macOS persistence)

Compensating: Without EDR for 30-day RAT callback monitoring, implement the following: (1) Use osquery to continuously audit persistence locations on developer workstations — run osqueryi 'SELECT * FROM launchd WHERE path LIKE "/Users/%/Library/LaunchAgents/%";' on macOS and osqueryi 'SELECT * FROM scheduled_tasks;' on Windows on a daily cron/scheduled task; (2) Deploy a Sigma rule to parse syslog or auditd for cron job modifications: monitor /var/spool/cron/crontabs/ (Linux) and /var/at/tabs/ (macOS) for changes post-incident; (3) For outbound RAT callback detection without NDR, configure host-based firewall logging on all previously affected nodes and pipe logs to

a central syslog server — filter for node.exe or the ClickFix payload process name making outbound TCP connections to ports 443, 8443, or non-standard ports; (4) Run npm audit --json > audit_results.json in each repo and parse with jq '.vulnerabilities | keys[]' to programmatically confirm absence of plain-crypto-js.

Evidence: During the 30-day monitoring window, collect and preserve: (1) netflow or firewall logs showing any outbound beacon traffic from previously affected build runners or developer workstations — UNC1069 RAT callbacks from North Korean infrastructure often use HTTPS to blend with normal traffic but may show beaconing intervals (fixed-interval connections to the same external IP); (2) Scheduled task / cron / LaunchAgent audit snapshots taken at recovery baseline versus current state — any new entries added after the ClickFix lure execution date that were not present before the incident indicate persistent implant activity; (3) npm audit output (JSON format) from each application repo immediately post-recovery, retained as the verified clean baseline for comparison if a future alert fires.

Step 5: Post-Incident — This campaign exposes three control gaps: (a) absence of package integrity verification in CI/CD pipelines (implement Sigstore or npm provenance attestation per SLSA framework guidance); (b) lack of registry mirroring with allowlist controls, allowing direct pulls from the public npm registry; (c) insufficient phishing-resistant MFA on npm publish accounts (enforce hardware key or passkey authentication for all package publish credentials). Map findings to NIST CSF Protect function controls PR.DS-6 (integrity checking) and PR.AC-1 (identity management). Schedule a supply chain risk review against NIST SP 800-161r1.

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 SA-12 (Supply Chain Protection), NIST IA-5 (Authenticator Management), NIST SI-2 (Flaw Remediation), NIST AU-6 (Audit Record Review, Analysis, and Reporting), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: For teams without budget for Sigstore or enterprise provenance tooling: (1) Implement npm package-lock.json integrity enforcement in CI — add a pre-install script that verifies the sha512 integrity hash in package-lock.json matches the expected value for axios@1.7.9 before allowing any install to proceed; (2) Configure Verdaccio (free, OSS npm proxy) as a registry mirror with an explicit package allowlist — any package not in the allowlist, including a future poisoned version, will be blocked at install time; (3) Enforce npm publish via hardware key by enabling npm's built-in WebAuthn/passkey support for publish operations (available in npm CLI 9+) — this would have blocked UNC1069 from publishing the malicious Axios versions even with harvested session tokens; (4) Add a YARA rule to your CI pipeline that scans every downloaded npm tarball for known malicious strings from this campaign (e.g., 'plain-crypto-js') before installation using yara .

Evidence: For the post-incident review, compile and retain: (1) A complete timeline of the UNC1069 campaign correlated against your environment's CI/CD pipeline execution logs — document exactly which build jobs ran during the three-hour exposure window, which ones resolved Axios 1.14.1 or 0.30.4, and which downstream artifacts were produced; (2) The maintainer workstation forensic image and memory capture from Step 3 — review these in the lessons-learned phase to understand the full ClickFix lure chain and whether your current phishing-resistant MFA controls would have interrupted UNC1069's initial access; (3) A software bill of materials (SBOM) generated post-recovery using cyclonedx-npm or syft for each affected application, establishing the verified clean dependency baseline and serving as the supply chain attestation record going forward.

Detection Guidance

Primary IOC: presence of 'plain-crypto-js' in any dependency tree, lock file, or node_modules directory. Query package-lock.json files across repositories for the string 'plain-crypto-js'. In SIEM, search npm install logs and CI/CD pipeline stdout for 'axios@1.14.1' or 'axios@0.30.4'. On endpoints, query EDR for file creation events involving 'plain-crypto-js' under any node_modules path. Behavioral indicators include: Node.js processes

spawning shell interpreters without expected application context (T1059.004); unexpected outbound TCP connections from node.exe or node processes to non-application destinations; credential store access (T1555) or cookie theft behavior (T1539) originating from developer workstation sessions around the exposure window. For threat hunting, based on observed ClickFix TTP patterns, search for browser-to-clipboard or browser-to-terminal paste events on developer machines, and look for execution of unsigned binaries or scripts downloaded immediately following a Teams-themed web page visit. MITRE techniques to prioritize in detection rules: T1195.002, T1204.002, T1219, T1059.004.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`npm package: plain-crypto-js`	Malicious dependency introduced by trojanized Axios versions 1.14.1 and 0.30.4; functions as dropper for cross-platform RAT	HIGH
URL	`npm package: axios@1.14.1`	Trojanized Axios version published by UNC1069 following maintainer account compromise; confirmed malicious	HIGH
URL	`npm package: axios@0.30.4`	Trojanized Axios version published by UNC1069 following maintainer account compromise; confirmed malicious	HIGH

Framework Mappings

MITRE-ATTACK

- **T1219** — Remote Access Tools
- **T1566.002** — Spearphishing Link
- **T1059** — Command and Scripting Interpreter
- **T1553** — Subvert Trust Controls
- **T1195.002** — Compromise Software Supply Chain
- **T1078** — Valid Accounts
- **T1539** — Steal Web Session Cookie
- **T1555** — Credentials from Password Stores
- **T1588.002** — Tool
- **T1204.002** — Malicious File
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1059.004** — Unix Shell
- **T1566** — Phishing

NIST-800-53R5

- **AT-2** — Literacy Training and Awareness

- **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
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CA-7** — Continuous Monitoring
- **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
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **15.1** — Establish and Maintain an Inventory of Service Providers

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

ISO-27001-2022

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1219	Remote Access Tools	Command-And-Control
T1566.002	Spearphishing Link	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1553	Subvert Trust Controls	Defense-Evasion
T1195.002	Compromise Software Supply Chain	Initial-Access
T1078	Valid Accounts	Defense-Evasion
T1539	Steal Web Session Cookie	Credential-Access
T1555	Credentials from Password Stores	Credential-Access
T1588.002	Tool	Resource-Development
T1204.002	Malicious File	Execution
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1059.004	Unix Shell	Execution
T1566	Phishing	Initial-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/axios-npm-hack-used-...	T3
Mitigating the Axios npm supply chain compromise - Microsoft	https://www.microsoft.com/en-us/security/blog/2026/04/01/mitigating...	T1
New axios 1.14.1 and 0.30.4 on npm are likely malicious - Reddit	https://www.reddit.com/r/msp/comments/1s8e1af/new_axios_1141_and_03...	T3
axios Compromised on npm - Malicious Versions Drop Remote ...	https://www.stepsecurity.io/blog/axios-compromised-on-npm-malicious...	T3
Axios NPM Package Compromised: Supply Chain Attack Hits ...	https://www.trendmicro.com/en_us/research/26/c/axios-npm-package-co...	T3

DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. 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 2026-04-05 05:56 UTC by TJS Security Command Center