INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-14 06:11 UTC

Technology Sector Under Sustained Nation-State and Criminal Siege: China, DPRK, and eCrime Actors Converge on AI, IP, and Supply Chains

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0456
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Axios npm package (v1.14.1, v0.30.4), GitHub repositories, macOS platforms, North American technology organizations, mail infrastructure targets of SUNRISE PANDA
Discovery Source	Rss:T1 Threatintel

Executive Summary

CrowdStrike's 2026 Technology Threat Landscape Report documents a sustained, multi-vector siege on the technology sector by China-nexus adversaries, DPRK-affiliated actors, and organized eCrime groups operating simultaneously across identity, software supply chain, and cloud environments. DPRK actor STARDUST CHOLLIMA compromised the Axios npm package, downloaded approximately 100 million times per week, injecting a remote access trojan into versions v1.14.1 and v0.30.4, exposing any organization whose software build pipeline consumed those versions. Per CrowdStrike's 2026 report, China-nexus actors account for more than 58% of state-sponsored intrusions against the sector, while eCrime groups named 572 technology companies on extortion leak sites in a single reporting year, creating compound risk spanning IP theft, ransomware exposure, insider threat from embedded DPRK IT workers, and downstream liability for organizations that ship compromised software to their own customers.

Technical Analysis

The Axios npm supply chain compromise (CWE-494: Download of Code Without Integrity Check; CWE-346: Origin Validation Error; CWE-287: Improper Authentication) involves STARDUST CHOLLIMA injecting a remote access trojan into Axios versions v1.14.1 and v0.30.4. Axios is one of the most widely consumed JavaScript HTTP client libraries, with approximately 100 million weekly downloads, making downstream blast radius exceptionally broad. The RAT enables persistent remote command execution on build systems and developer endpoints that installed the poisoned versions during the compromise window. MITRE ATT&CK coverage includes T1195.001 (Compromise Software Dependencies and Development Tools) and T1195.002

(Compromise Software Supply Chain) as the initial access vector, with T1059 (Command and Scripting Interpreter), T1071 (Application Layer Protocol), T1027 (Obfuscated Files or Information), T1090 (Proxy), and T1133 (External Remote Services) supporting post-compromise operations and C2 callbacks. SUNRISE PANDA conducted separate intrusions against mail infrastructure (T1213, T1505), while Glassworm targeted GitHub repositories and an OpenClaw-lure infostealer campaign targeted macOS platforms. DPRK IT worker insertion (T1136, T1078) provides persistent insider access independent of technical exploitation. No CVE ID is assigned to the Axios compromise in the provided source data. Organizations should audit any pipeline that consumed axios@1.14.1 or axios@0.30.4 as potentially compromised build environments.

Action Checklist

1. Step 1: Containment. Immediately audit all package-lock.json, yarn.lock, and pnpm-lock.yaml files across every code repository and CI/CD pipeline for Axios versions v1.14.1 or v0.30.4; block installation of those specific versions at the package registry or artifact proxy level (Artifactory, Nexus, or equivalent) to prevent further consumption. Isolate any build agent or developer workstation confirmed to have installed either version during the compromise window.
2. Step 2: Detection. Search SIEM and EDR telemetry for processes spawned by npm install or node during the window when Axios v1.14.1 or v0.30.4 was available; look for unexpected outbound network connections from build servers or CI runners, particularly to uncommon destinations using T1071-style application-layer protocols. Query dependency manifests for 'axios' at the two affected version strings. Review AU-2 (Event Logging) and AU-6 (Audit Record Review) to confirm build pipeline activity was captured. Cross-reference CIS 8.2 (Collect Audit Logs) compliance: if build-agent logging was not enabled, treat pipeline as potentially compromised and escalate.
3. Step 3: Eradication. Upgrade Axios to a clean, verified version (consult the official Axios GitHub releases page and CrowdStrike advisory at https://www.crowdstrike.com/en-us/blog/stardust-chollima-likely-compromises-axios-npm-package/ for the confirmed safe version floor). Rebuild all artifacts produced during or after consumption of the poisoned versions from a verified clean state. Rotate all secrets, tokens, and credentials accessible to compromised build environments (NIST IA-4, CIS 5.1: Credential Rotation). Re-image build agents confirmed to have run the compromised package per NIST IA-2 (Credential Hardening) principles.
4. Step 4: Recovery. Validate rebuilt artifacts using file integrity verification (NIST SI-7: Software, Firmware, and Information Integrity) and compare dependency hashes against known-good registries. Restore pipeline operations only from verified clean build agents. Enable continuous local account monitoring on build infrastructure and developer endpoints for 30 days post-remediation (NIST AU-6: Audit Review) to detect any residual RAT activity. Confirm NIST AU-9 (Protection of Audit Information) controls are in place to prevent log tampering by any residual implant.
5. Step 5: Post-Incident. Conduct a software supply chain risk assessment against NIST SI-7 (Software, Firmware, and Information Integrity) principles mapped to CIS 2.1 (Establish and Maintain a Software Inventory) and CIS 2.2 (Ensure Authorized Software is Currently Supported). Implement mandatory sub-resource integrity checks or pinned, hash-verified dependency lockfiles as a standing control. Establish a third-party IT worker vetting and access control process aligned to NIST AC-2 (Account Management) and NIST AC-6 (Least Privilege) to address the DPRK IT worker insertion threat vector. Schedule a full supply chain security review against MITRE T1195 technique family coverage gaps identified by this incident.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO, legal counsel, and external IR retainer immediately if: any secrets, tokens, or credentials from compromised build environments are confirmed exfiltrated or used post-compromise; if downstream customers or partners consumed artifacts built with the poisoned axios versions (triggering potential breach notification obligations); if DPRK IT worker insertion is confirmed at any tier of the organization; or if the team lacks memory forensics capability to rule out persistent RAT implantation on build infrastructure.
Recovery Notes	Restore pipeline operations exclusively from build agents provisioned from a verified clean IaC baseline after all credential rotation is confirmed complete and validated — do not bring back any agent that ran axios v1.14.1 or v0.30.4 without a full re-image and baseline integrity check. Monitor rebuilt build agents and developer endpoints for 30 days using Sysmon or auditd, specifically alerting on `node.exe` or `npm` initiating outbound connections to non-registry destinations, new local account creation, and modifications to shell profile or npm global package directories that could indicate RAT re-establishment. If any downstream artifacts were distributed to customers or deployed to production environments, initiate parallel artifact recall and customer notification workflows per your incident response and disclosure policy.
Forensic Artifacts	npm cache directories containing the poisoned axios tarball: `~/.npm/_cacache/` (Linux/macOS) or `%AppData%\npm-cache\` (Windows) — the STARDUST CHOLLIMA-modified package will be present here on any agent that ran `npm install axios@1.14.1` or `npm install axios@0.30.4` during the compromise window, and can be extracted for malware analysis and YARA signature development. | Sysmon Event ID 3 (Network Connection) and Event ID 1 (Process Creation) logs on Windows build agents, specifically parent-child chains showing `npm.cmd` or `node.exe` spawning unexpected child processes or initiating outbound TCP connections to non-npm-registry IP ranges — these are the primary host-based indicators of the axios RAT executing its initial C2 beacon. | CI/CD pipeline execution logs (GitHub Actions `workflow_run` logs, Jenkins `build.xml`, GitLab CI job artifacts) timestamped to the period when the poisoned axios versions were resolvable from the npm registry — these establish which builds consumed the trojanized package and which downstream artifacts must be treated as potentially backdoored. | DNS resolution logs from build network segments for the compromise window — the axios RAT injected by STARDUST CHOLLIMA will have generated DNS queries for C2 infrastructure domains from `node.exe` processes, which will appear anomalous against a baseline of expected npm registry and CDN domains in build environment DNS telemetry. | GitHub repository audit logs and commit history diffs for `package-lock.json` files across affected repositories — these establish the change-control timeline showing when axios v1.14.1 or v0.30.4 was first introduced as a locked dependency, identify which developer or automated account made the change, and support determination of whether the poisoned version was introduced via a direct commit, a dependency update bot (Dependabot, Renovate), or a compromised maintainer account.

Per-Action IR Details

Step 1: Containment — Immediately audit all package-lock.json, yarn.lock, and pnpm-lock.yaml files across every code repository and CI/CD pipeline for axios versions v1.14.1 or v0.30.4; block installation of those specific versions at the package registry or artifact proxy level (Artifactory, Nexus, or equivalent) to prevent further consumption. Isolate any build agent or developer workstation confirmed to have installed either version during the compromise window.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: isolate affected systems to prevent further spread while preserving evidence; prioritize systems critical to ongoing operations (build infrastructure) for immediate isolation.

Controls: NIST AC-4 (Information Flow Enforcement), CIS 2.3 (Address Unauthorized Software), CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: Run `grep -r '"axios"' . --include='package-lock.json' --include='yarn.lock' --include='pnpm-lock.yaml' | grep -E '1\.14\.1|0\.30\.4'` across all repo roots from a central bastion or via a git-hosted CI job. Block the two version strings in Nexus OSS (free tier) under Repository Management > Routing Rules, or in a local npm proxy by adding a deny rule for `axios@1.14.1` and `axios@0.30.4`. For workstation isolation without EDR, use host-based firewall rules: `netsh advfirewall firewall add rule name='ISOLATE' dir=out action=block` (Windows) or `iptables -I OUTPUT -j DROP` (Linux), leaving only management SSH/RDP open.

Evidence: Before isolating any build agent or workstation, capture volatile state: (1) full RAM image using WinPmem (Windows) or `avml` (Linux) to preserve in-memory RAT artifacts from the axios trojan payload; (2) active network connections via `netstat -ano` (Windows) or `ss -tunap` (Linux) — STARDUST CHOLLIMA implants establish C2 over application-layer protocols and active connections will be lost on isolation; (3) running process list with parent-child relationships via `Get-WmiObject Win32_Process | Select-Object ProcessId,ParentProcessId,Name,CommandLine` (Windows) or `ps auxf` (Linux); (4) list of all npm packages currently installed in the build agent's node_modules (`npm list --depth=0 --json > npm_snapshot.json`); (5) environment variables (`set > env_snapshot.txt` / `env > env_snapshot.txt`) to capture any secrets or tokens accessible to the compromised build process before rotation.

Step 2: Detection — Search SIEM and EDR telemetry for processes spawned by npm install or node during the window when axios v1.14.1 or v0.30.4 was available; look for unexpected outbound network connections from build servers or CI runners, particularly to uncommon destinations using T1071-style application-layer protocols. Query dependency manifests for 'axios' at the two affected version strings. Review AU-2 (Event Logging) and AU-6 (Audit Record Review) to confirm build pipeline activity was captured. Cross-reference CIS 8.2 (Collect Audit Logs) compliance: if build-agent logging was not enabled, treat pipeline as potentially compromised and escalate.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: correlate indicators across log sources, establish a timeline of the compromise window, and determine scope of exposure; DE.AE-03 (correlate information from multiple sources) and DE.AE-04 (estimate impact and scope) apply directly.

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

Compensating: Without SIEM, deploy Sysmon on Windows build agents using SwiftOnSecurity's base config augmented with Event ID 3 (Network Connection) filtering on processes `node.exe` and `npm.cmd` — export with `wevtutil qe Microsoft-Windows-Sysmon/Operational /f:text > sysmon_build.txt`. On Linux CI runners, enable auditd rules: `auditctl -a always,exit -F arch=b64 -S connect -F exe=/usr/bin/node -k axios_c2`. For dependency scanning without EDR, run `npm audit --json` and pipe through `jq '.vulnerabilities | to_entries[] | select(.value.name=="axios")'`. Use a Sigma rule targeting `node` or `npm` spawning `cmd.exe`, `sh`, `curl`, or `wget` as child processes — convert to native query with `sigma convert -t splunk` or run manually against exported Sysmon XML logs.

Evidence: This is a detection/analysis phase step that reads existing evidence rather than destroying it; however, if active C2 is suspected, capture before querying: (1) Sysmon Event ID 3 logs from build agents covering the axios compromise window — specifically connections from `node.exe` to non-npm-registry IPs; (2) CI/CD pipeline execution logs (GitHub Actions workflow logs, Jenkins build console output, GitLab CI job traces) timestamped to the window when v1.14.1 or v0.30.4 was resolvable; (3) npm registry access logs or Artifactory/Nexus request logs showing which agents pulled `axios@1.14.1` or `axios@0.30.4` and when; (4) DNS query logs from build network segments for domains resolved by `node.exe` processes — STARDUST CHOLLIMA RAT callbacks will appear here as anomalous non-npm destinations; (5) GitHub repository audit logs (available via GitHub Enterprise audit log API or `gh api /orgs/{org}/audit-log`) for any unauthorized commits or workflow file modifications that could indicate supply chain pivot beyond the npm package.

Step 3: Eradication — Upgrade axios to a clean, verified version (consult the official axios GitHub releases page and CrowdStrike advisory at https://www.crowdstrike.com/en-us/blog/stardust-chollima-likely-compromises-axios-npm-package/ for the confirmed safe version floor). Rebuild all artifacts produced during or after consumption of the poisoned versions from a verified clean state. Rotate all secrets, tokens, and credentials accessible to compromised build environments (D3-CRO: Credential Rotation). Re-image build agents confirmed to have run the compromised package per D3-CH (Credential Hardening) principles.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: remove all components of the incident from the environment, including malicious code, compromised accounts, and vulnerabilities that enabled the attack; for supply chain compromise, eradication requires artifact rebuild from verified clean source, not merely package removal.

Controls: NIST SI-2 (Flaw Remediation), NIST AC-2 (Account Management), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Verify the axios replacement version's integrity without an enterprise tool: download the tarball from npm (`npm pack axios@`) and compare its SHA-512 hash against the value published in the npm registry manifest (`npm view axios@ dist.shasum`). For credential rotation without a PAM tool, use GitHub's built-in secret scanning push protection to block re-introduction of rotated secrets, and enumerate all secrets in CI using `gh secret list` per affected repository. For build agent re-imaging on bare-metal or VM without enterprise tooling, boot from a verified OS ISO, validate ISO SHA-256 against the distribution's signed checksum file, and provision from an IaC template (Ansible playbook or Packer template) stored in a separate, isolated source-of-truth repository that was confirmed clean before the compromise window.

Evidence: Before re-imaging build agents or rotating credentials — both of which destroy or invalidate live state — ensure the following volatile captures from Step 1 are complete and preserved to write-once storage: (1) full RAM image preserving any in-memory RAT payload injected by the axios trojan, including decoded C2 configuration that may be resident only in heap memory; (2) a forensic disk image of the compromised build agent (`dd if=/dev/sda | gzip > build_agent_disk.img.gz` or Windows equivalent via FTK Imager) before reimage — the RAT may have written persistence mechanisms to `~/.npm/_cacache`, node_modules `.bin` directories, or shell profile files (`.bashrc`, `.profile`); (3) export all CI/CD environment variable secrets (names only, not values) from GitHub Actions / GitLab / Jenkins before rotation to document the full credential blast radius; (4) preserve the poisoned axios package tarball itself (`~/.npm/_cacache` on Linux or `%AppData%\npm-cache` on Windows) as a malware sample for YARA rule development and submission to threat intel sharing platforms.

Step 4: Recovery — Validate rebuilt artifacts using file integrity verification (D3-FMBV: File Magic Byte Verification; D3-SFA: System File Analysis) and compare dependency hashes against known-good registries. Restore pipeline operations only from verified clean build agents. Enable D3-LAM (Local Account Monitoring) on build infrastructure and developer endpoints for 30 days post-remediation to detect any residual RAT activity. Confirm AU-9 (Protection of Audit Information) controls are in place to prevent log tampering by any residual implant.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore systems to normal operation with confidence that the threat has been fully eradicated; verify integrity of restored systems before returning to production, and monitor for signs of reinfection or residual implant activity.

Controls: NIST AU-9 (Protection Of Audit Information), NIST SI-7 (Software, Firmware, and Information Integrity), CIS 2.1 (Establish and Maintain a Software Inventory), CIS 7.3 (Perform Automated Operating System Patch Management)

Compensating: For artifact integrity verification without enterprise tooling: run `npm ci --ignore-scripts` (the `--ignore-scripts` flag prevents postinstall hooks — a key STARDUST CHOLLIMA RAT delivery vector) on rebuilt projects and compare `package-lock.json` SHA-512 hashes against the npm registry via `npm view @ dist.integrity`. Use AIDE (Advanced Intrusion Detection Environment, free) on Linux build agents to baseline filesystem state immediately after clean provisioning and alert on drift. For local account monitoring, deploy Sysmon Event ID 4624/4625 forwarding (Windows) or `auditd` `USER_LOGIN` rules (Linux) to a centralized syslog server (rsyslog or

syslog-ng, both free), and create an alerting threshold for any new local account creation (`net user` / `useradd` events) on build infrastructure.

Evidence: Recovery phase steps modify system state by restoring from backup and enabling monitoring agents; before cutting over rebuilt build agents to production pipelines, verify: (1) run `shasum -a 512` on every rebuilt artifact's dependency tree and diff against a pre-incident known-good lockfile snapshot stored in a separate repository or artifact store; (2) confirm no persistence mechanisms remain by checking axios RAT-typical persistence locations — cron entries (`crontab -l`), systemd unit files (`systemctl list-units --type=service`), npm global packages (`npm list -g --depth=0`), and shell profile modifications (`diff ~/.bashrc ~/.bashrc.orig`); (3) validate that audit log forwarding to the centralized syslog destination is functioning by injecting a test event and confirming receipt before relying on AU-9 protections; (4) document the clean baseline state (filesystem hashes, installed package manifest, running services) as the recovery reference point for the 30-day monitoring window.

Step 5: Post-Incident — Conduct a software supply chain risk assessment against NIST SI-7 (Software, Firmware, and Information Integrity) principles mapped to CIS 2.1 (Establish and Maintain a Software Inventory) and CIS 2.2 (Ensure Authorized Software is Currently Supported). Implement mandatory sub-resource integrity checks or pinned, hash-verified dependency lockfiles as a standing control. Establish a third-party IT worker vetting and access control process aligned to AC-2 (Account Management) and AC-6 (Least Privilege) to address the DPRK IT worker insertion threat vector. Schedule a full supply chain security review against MITRE T1195 technique family coverage gaps identified by this incident.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: conduct lessons-learned review, update detection and response capabilities based on evidence gathered, and implement structural improvements to prevent recurrence; supply chain compromise by a nation-state actor warrants a formal after-action report and policy revision cycle.

Controls: NIST SI-7 (Software, Firmware, and Information Integrity), NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), 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: Implement hash-pinned lockfiles immediately at zero cost: enforce `npm ci` (which validates `package-lock.json` integrity hashes) in all CI pipelines by adding a pre-build gate script that fails if `npm install` is used instead. Use the free `socket.dev` CLI (`npx socket`) or `npm audit` to scan for supply chain risk indicators in all third-party packages — `socket` specifically flags packages with install scripts, typosquatting, and maintainer account takeover indicators relevant to STARDUST CHOLLIMA TTPs. For DPRK IT worker vetting without a commercial identity verification service, implement a mandatory video-verified onboarding checklist cross-referencing government-issued ID against OFAC SDN list lookups (free via OFAC API) and require hardware-bound MFA tokens (YubiKey) before granting any repository or CI/CD access.

Evidence: Post-incident activity does not destroy live evidence; however, the lessons-learned process requires preserving the complete incident artifact package: (1) the full timeline reconstructed from CI/CD logs, Sysmon/auditd captures, and npm registry pull logs documenting first and last consumption of axios v1.14.1 and v0.30.4; (2) the preserved malware sample from the npm cache (see Step 3) for YARA rule creation targeting STARDUST CHOLLIMA RAT signatures — submit to internal threat intel library and optionally to VirusTotal or an ISAC; (3) a diff of all `package-lock.json`, `yarn.lock`, and `pnpm-lock.yaml` files between the last known-good commit and the first commit introducing the poisoned axios version, to document the exact change-control gap; (4) access provisioning records for any contractor or third-party developer accounts with repository or CI/CD access during the compromise window, to assess DPRK IT worker insertion risk surface; (5) a complete inventory of all artifacts (container images, compiled binaries, deployment packages) produced from the compromised build environment, to support downstream customer or partner notification obligations.

Detection Guidance

Primary detection focus is the Axios supply chain compromise. Query dependency manifests across all repositories for 'axios' at versions '1.14.1' or '0.30.4', flag any match as a priority investigation. In CI/CD pipeline

logs, search for npm install or yarn install events that resolved Axios to either affected version during the compromise window; correlate timestamps against outbound network events from the same build agent. On developer endpoints and build servers, hunt for unexpected child processes spawned by node.exe or npm processes, particularly those making outbound connections to non-standard destinations (T1133: External Remote Services; T1090: Proxy). EDR behavioral detections should flag encoded command execution (T1027), scripting interpreter activity from build contexts (T1059), and proxy-routed C2 traffic (T1090). For the broader campaign, monitor for: anomalous privileged account creation (T1136) consistent with DPRK IT worker insertion; mail server access anomalies and unusual data repository access (T1213) consistent with SUNRISE PANDA TTPs; macOS process execution chains from document-lure files consistent with OpenClaw infostealer activity. MITRE ATT&CK techniques T1195.001 and T1195.002 should be mapped to detection rules in SIEM for dependency-related alerts. Review AU-6 (Audit Record Review, Analysis, and Reporting) cadence; increase to daily for build infrastructure during elevated threat periods. CIS 8.2 compliance (Collect Audit Logs) should be verified across all CI/CD and source control systems before concluding the environment is clean.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.npmjs.com/package/axios/v/1.14.1`	Compromised axios npm package version — do not install; remove from all dependency lockfiles immediately	HIGH
URL	`https://www.npmjs.com/package/axios/v/0.30.4`	Compromised axios npm package version — do not install; remove from all dependency lockfiles immediately	HIGH

Framework Mappings

MITRE-ATTACK

- **T1133** — External Remote Services
- **T1098** — Account Manipulation
- **T1027** — Obfuscated Files or Information
- **T1588.001** — Malware
- **T1136** — Create Account
- **T1213** — Data from Information Repositories
- **T1566** — Phishing
- **T1059** — Command and Scripting Interpreter
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1110.003** — Password Spraying
- **T1071** — Application Layer Protocol
- **T1078** — Valid Accounts
- **T1505** — Server Software Component
- **T1195.002** — Compromise Software Supply Chain

- **T1090** — Proxy
- **T1657** — Financial Theft

NIST-800-53R5

- **AC-17** — Remote Access
- **AC-20** — Use of External Systems
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous 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
- **CM-3** — Configuration Change Control
- **IA-8** — Identification and Authentication (Non-Organizational Users)
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A07:2021** — Identification and Authentication Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.3** — Require MFA for Externally-Exposed Applications
- **6.4** — Require MFA for Remote Network Access
- **6.5** — Require MFA for Administrative Access
- **15.1** — Establish and Maintain an Inventory of Service Providers

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(d)** — Person or Entity Authentication

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
T1133	External Remote Services	Persistence
T1098	Account Manipulation	Persistence
T1027	Obfuscated Files or Information	Defense-Evasion
T1588.001	Malware	Resource-Development
T1136	Create Account	Persistence
T1213	Data from Information Repositories	Collection
T1566	Phishing	Initial-Access
T1059	Command and Scripting Interpreter	Execution
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1110.003	Password Spraying	Credential-Access
T1071	Application Layer Protocol	Command-And-Control
T1078	Valid Accounts	Defense-Evasion
T1505	Server Software Component	Persistence
T1195.002	Compromise Software Supply Chain	Initial-Access
T1090	Proxy	Command-And-Control
T1657	Financial Theft	Impact

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/crowdstrike-2026-technology-...	T3
Axios NPM Package Compromised: Supply Chain Attack Hits ...	https://www.trendmicro.com/en_us/research/26/c/axios-npm-package-co...	T3
STARDUST CHOLLIMA Likely Compromises Axios npm Package	https://www.crowdstrike.com/en-us/blog/stardust-chollima-likely-com...	T3
axios npm Compromised: RAT in v1.14.1 & v0.30.4 (2026)	https://phoenix.security/axios-supply-chain-compromise-npm-rat-2026/	T3
Axios Supply Chain Attack: Analysis & Fix - Orca Security	https://orca.security/resources/blog/axios-npm-supply-chain-attack-...	T3

Generated 2026-06-14 06:11 UTC by TJS Security Command Center