INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-14 04:22 UTC

China and DPRK Dominate Technology Sector Targeting: 2026 Threat Landscape Signals Escalating Supply Chain and IP Risk

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0452
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, software development companies, mail infrastructure
Discovery Source	Rss:T1 Threatintel

Executive Summary

CrowdStrike's 2026 Technology Threat Landscape Report identifies the technology sector as the most targeted industry globally, with China-nexus actors responsible for more than 58% of state-sponsored intrusions and DPRK-linked FAMOUS CHOLLIMA leading hands-on-keyboard operations at 47% of observed state activity. A confirmed supply chain incident during this period compromised axios npm package versions v1.14.1 and v0.30.4 with an embedded Remote Access Trojan, creating downstream code execution risk across any software project consuming those versions. Organizations face compounding pressure from eCrime actors, with 572 technology companies named on leak sites and initial access broker listings for the sector up approximately 30% year-over-year, signaling sustained, multi-vector targeting of software development pipelines and intellectual property repositories.

Technical Analysis

This item covers a threat landscape report period (April 2025-March 2026) with an embedded, confirmed supply chain incident. The axios npm package, one of the most widely consumed HTTP client libraries in the JavaScript ecosystem, was compromised in two specific versions: v1.14.1 and v0.30.4. Both versions were backdoored with a Remote Access Trojan enabling downstream code execution in dependent projects. No CVE ID is present in the source data for this specific axios compromise; upstream pipeline has validated this item accordingly. Relevant CWE patterns: CWE-494 (Download of Code Without Integrity Check), CWE-829 (Inclusion of Functionality from Untrusted Control Sphere), and CWE-306 (Missing Authentication for Critical Function). MITRE ATT&CK techniques observed across the broader campaign include T1195.001 and

T1195.002 (supply chain compromise of software and dependencies), T1078 (valid accounts), T1199 (trusted relationship abuse), T1072 (software deployment tools), T1059 (command and scripting interpreter), T1566 (phishing), T1110.003 (password spraying), T1176 (browser extensions), T1213 (data from information repositories), T1486 (data encrypted for impact), T1190 (exploit public-facing application), T1587.001 (malware development), T1591 (gather victim organization information). FAMOUS CHOLLIMA is documented for insider threat placement at technology firms and macOS-targeted malware deployment. China-nexus actors focused on persistent access and IP exfiltration from software development environments and GitHub repositories.

Action Checklist

1. Step 1: Containment, Audit all software projects and CI/CD pipelines for dependencies on axios v1.14.1 or v0.30.4; immediately isolate any build environment or deployed service that consumed these versions. Block outbound connections to unknown or suspicious external endpoints from affected systems pending investigation. Do not block all outbound connectivity, as this may disrupt package registry access needed for remediation.
2. Step 2: Detection, Query package-lock.json, yarn.lock, and npm audit outputs across all repositories for axios versions v1.14.1 and v0.30.4. Review process creation and outbound network logs for anomalous connections originating from Node.js processes. Examine endpoint telemetry on macOS developer machines for indicators of FAMOUS CHOLLIMA-associated implants. Cross-reference AU-6 (Audit Record Review, Analysis, and Reporting) log sources including SIEM, EDR, and network flow data for command-and-control patterns consistent with RAT beaconing (CIS 8.2, Collect Audit Logs).
3. Step 3: Eradication, Upgrade axios to the latest verified clean release per the official axios GitHub advisory (issues/10636) and npm security advisory. Purge npm cache and rebuild dependency trees from clean package manifests. Rotate any secrets, tokens, API keys, and credentials accessible from build environments that ran compromised versions. Apply credential rotation (D3-CRO) for all service accounts and developer credentials exposed to affected pipelines. Review and harden package integrity verification processes per CWE-494 mitigation guidance.
4. Step 4: Recovery, Validate remediated builds produce clean npm audit results with no flagged axios versions. Re-scan deployed artifacts and container images rebuilt after the compromise window. Enable subresource integrity checks and package signing verification where supported (NIST SI-4 monitoring controls). Monitor developer endpoints and CI/CD runners for 30 days post-remediation for delayed-execution RAT activity. Confirm audit logging is intact and has not been tampered with per NIST AU-9 (Protection of Audit Information).
5. Step 5: Post-Incident, Conduct a full software supply chain risk review: map all third-party dependencies against verified publisher signatures and integrity hashes (CIS 2.1, Establish and Maintain a Software Inventory; CIS 2.2, Ensure Authorized Software is Currently Supported). Implement automated dependency scanning in CI/CD pipelines to enforce allowlisted package versions. Review insider threat controls given FAMOUS CHOLLIMA's documented tactic of placing personnel at technology firms; strengthen identity verification in hiring and privileged access provisioning (NIST AC-2, Account Management; CIS 5.4, Restrict Administrator Privileges to Dedicated Administrator Accounts). Assess whether IAB-listed assets require immediate credential rotation and access review.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to senior IR leadership and legal counsel immediately if memory forensics or network captures confirm active C2 beaconing from a Node.js process, if source code repositories show unauthorized commits during the compromise window, or if any affected build environment had access to customer PII or regulated data triggering breach notification obligations under GDPR, CCPA, or applicable state law.
Recovery Notes	All production artifacts — Docker images, Lambda packages, or deployed Node.js services — built during the axios v1.14.1/v0.30.4 exposure window must be treated as potentially backdoored and rebuilt from clean dependency manifests before being returned to service. Monitor CI/CD runners and macOS developer endpoints via Sysmon or osquery for 30 days post-remediation, specifically watching for LaunchAgent persistence, scheduled task creation, and outbound connections to newly registered or low-reputation domains consistent with FAMOUS CHOLLIMA C2 infrastructure. Given FAMOUS CHOLLIMA's documented use of delayed-execution implants and insider placement, access reviews for privileged developer accounts and CI/CD service principals should be repeated at 15-day and 30-day intervals.
Forensic Artifacts	Trojanized axios tarball: SHA-512 integrity hash from `package-lock.json` `integrity` field for axios v1.14.1 or v0.30.4, cross-referenced against the npm registry manifest to confirm tampering — this is the direct evidence of supply chain compromise | Node.js process network connections: `netstat -ano` / `lsof -i` output correlating node PIDs to outbound C2 connections initiated at install or import time by the embedded RAT payload in the compromised axios package | npm cache tarballs: Files under `~/.npm/_cacache/content-v2/sha512/` corresponding to the compromised axios versions, preserved as binary evidence of the malicious package before cache purge | macOS persistence artifacts: LaunchAgent plist files under `~/Library/LaunchAgents/` and `~/Library/LaunchDaemons/` on developer machines, which FAMOUS CHOLLIMA macOS implants use to survive reboots following initial access via compromised npm packages | CI/CD pipeline audit logs: GitHub Actions workflow run logs and audit log API entries (`/orgs/{org}/audit-log`) documenting which pipelines executed builds consuming axios v1.14.1 or v0.30.4, establishing the blast radius of downstream code execution across the software supply chain

Per-Action IR Details

Step 1: Containment — Audit all software projects and CI/CD pipelines for dependencies on axios v1.14.1 or v0.30.4; immediately isolate any build environment or deployed service that consumed these versions. Block outbound connections from affected systems pending investigation.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.5 (Implement and Manage a Firewall on End-User Devices)

Compensating: Run `grep -r 'axios' **/package-lock.json | grep -E '1\.14\.1|0\.30\.4'` across all repo checkouts to identify affected projects. Use `ufw deny out` (Linux) or `netsh advfirewall firewall add rule dir=out action=block` (Windows) to block outbound traffic from affected build hosts before network-level controls are applied. For macOS developer machines, use `pfctl` with a deny-all egress ruleset: `echo 'block out all' | sudo pfctl -ef -`.

Evidence: Before isolating any build host or CI/CD runner, capture: (1) live RAM dump using `osxpmem` (macOS) or `winpmem` (Windows) to preserve in-memory RAT artifacts from the axios-embedded trojan; (2) active network connection state via `netstat -ano` (Windows) or `lsof -i -n -P` (macOS/Linux) documenting all Node.js process outbound connections; (3) full process tree snapshot via `ps auxf` (Linux/macOS) or `Get-CimInstance Win32_Process | Select-Object ProcessId,ParentProcessId,Name,CommandLine` (Windows) to identify node processes spawned by

the compromised package. These artifacts are destroyed upon network isolation or process termination.

Step 2: Detection — Query package-lock.json, yarn.lock, and npm audit outputs across all repositories for axios versions v1.14.1 and v0.30.4. Review process creation and outbound network logs for anomalous connections originating from Node.js processes. Examine endpoint telemetry on macOS developer machines for indicators of FAMOUS CHOLLIMA-associated implants. Cross-reference AU-6 (Audit Record Review, Analysis, and Reporting) log sources including SIEM, EDR, and network flow data for command-and-control patterns consistent with RAT beaconing (CIS 8.2 — Collect Audit Logs).

NIST Phase: Detection Analysis

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

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs)

Compensating: Run `npm audit --json | jq '.vulnerabilities | keys[]'` and `grep -rE '"axios":\s*"(1\.14\.1|0\.30\.4)"' **/package-lock.json` across all repos. On macOS developer machines, use `osquery` with query `SELECT pid, name, cmdline, parent FROM processes WHERE name='node';` to identify suspicious Node.js invocations. For network-based C2 detection without a SIEM, use Wireshark or `tcpdump -i any -w axios_capture.pcap 'tcp and (port 443 or port 80)' ` and filter for beaconing intervals from node PIDs. Write a Sigma rule targeting Sysmon Event ID 3 (Network Connection) where Image ends with `node` or `node.exe` and DestinationPort is non-standard.

Evidence: Capture before any process or session termination: (1) npm cache contents at `~/.npm/_cacache/` (macOS/Linux) or `%AppData%\npm-cache` (Windows) to confirm the trojanized axios tarball hash; (2) Sysmon Event ID 1 (Process Create) and Event ID 3 (Network Connection) logs filtered on `node.exe` or `node` as parent/child process; (3) on macOS, `/var/log/unified system log` entries and `~/.bash_history` / `~/.zsh_history` for post-exploitation commands consistent with FAMOUS CHOLLIMA initial access tradecraft; (4) DNS query logs from the affected host's resolver cache (`ipconfig /displaydns` on Windows, `scutil --dns` on macOS) to identify C2 domain lookups initiated by the RAT callback.

Step 3: Eradication — Upgrade axios to the latest verified clean release per the official axios GitHub advisory (issues/10636). Purge npm cache and rebuild dependency trees from clean package manifests. Rotate any secrets, tokens, API keys, and credentials accessible from build environments that ran compromised versions. Apply D3-CRO (Credential Rotation) for all service accounts and developer credentials exposed to affected pipelines. Review and harden package integrity verification processes per CWE-494 mitigation guidance.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

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

Compensating: Run `npm cache clean --force` then `npm install axios@latest --save-exact` and verify the resolved package hash against the axios GitHub advisory (issues/10636) checksum. For credential rotation without a PAM tool, enumerate exposed secrets using `truffleHog --regex --entropy=False` against CI/CD pipeline configs, `.env` files, and GitHub Actions secrets references. Revoke and reissue GitHub personal access tokens, npm publish tokens, and cloud provider API keys (AWS, GCP, Azure) scoped to affected CI/CD runners. Document all rotated credentials with timestamps for post-incident audit trail.

Evidence: Before rotating credentials or purging the npm cache, capture: (1) a full directory listing and hash inventory of `~/.npm/_cacache/` to preserve the trojanized axios tarball (sha1/sha512 from `package-lock.json` integrity field) as forensic evidence of the supply chain artifact; (2) CI/CD environment variable exports (e.g., `printenv` from the runner) to document which secrets were in scope — mask values but preserve variable names; (3) git log of any pipeline configuration changes in the compromise window (`git log --all --since='YYYY-MM-DD' -- .github/workflows/`) to identify whether FAMOUS CHOLLIMA operators modified pipeline definitions to persist access. These are destroyed when cache is purged or environment variables are rotated.

Step 4: Recovery — Validate remediated builds produce clean npm audit results with no flagged axios versions. Re-scan deployed artifacts and container images rebuilt after the compromise window. Enable subresource integrity checks and package signing verification where supported (NIST SI-4 monitoring controls). Monitor developer endpoints and CI/CD runners for 30 days post-remediation for delayed-execution RAT activity. Confirm audit logging is intact and has not been tampered with per NIST AU-9 (Protection of Audit Information).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-9 (Protection Of Audit Information), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Run `npm audit --audit-level=critical` and `npm ls axios` against rebuilt dependency trees to confirm no axios v1.14.1 or v0.30.4 references remain. For container image scanning without an enterprise tool, use `trivy image ` (free, open-source) against all rebuilt images. To verify audit log integrity without a SIEM, compare log file hashes (`sha256sum /var/log/syslog*` or Windows Security Event Log export hashes) against pre-incident baselines stored off-system. Deploy a Sysmon configuration with the SwiftOnSecurity ruleset on CI/CD runners to detect delayed-execution RAT persistence mechanisms (e.g., LaunchAgent plist drops on macOS, scheduled task creation on Windows).

Evidence: Before returning systems to production, verify: (1) no modification timestamps on audit logs during the compromise window — check `/var/log/auth.log`, Windows Security Event Log (Event ID 1102: Log Cleared, Event ID 4719: Audit Policy Changed) for evidence of FAMOUS CHOLLIMA operators suppressing forensic trails; (2) container image layer digests (`docker inspect --format='{{.RootFS.Layers}}' `) to confirm no compromised axios layer persists in the rebuild; (3) macOS LaunchAgent and LaunchDaemon directories (`~/Library/LaunchAgents/`, `/Library/LaunchDaemons/`) for persistence artifacts associated with FAMOUS CHOLLIMA macOS implants, which must be documented before any reimaging of developer endpoints.

Step 5: Post-Incident — Conduct a full software supply chain risk review: map all third-party dependencies against verified publisher signatures and integrity hashes (CIS 2.1 — Establish and Maintain a Software Inventory; CIS 2.2 — Ensure Authorized Software is Currently Supported). Implement automated dependency scanning in CI/CD pipelines to enforce allowlisted package versions. Review insider threat controls given FAMOUS CHOLLIMA's documented tactic of placing personnel at technology firms; strengthen identity verification in hiring and privileged access provisioning (NIST AC-2 — Account Management; CIS 5.4 — Restrict Administrator Privileges to Dedicated Administrator Accounts). Assess whether IAB-listed assets require immediate credential rotation and access review.

NIST Phase: Post Incident

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

Controls: 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 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Use `npm audit` with a custom `.nsprc` allowlist or integrate `socket.dev` CLI (free tier) into CI/CD pipelines to block packages with known supply chain flags. Generate a software bill of materials (SBOM) with `cyclonedx-npm --output-format json` for all production Node.js projects and diff against a known-good baseline. For insider threat controls without an enterprise DLP platform, audit GitHub access logs (`Settings > Security Log` per org) and revoke repository access for any contractor or new-hire accounts provisioned during the FAMOUS CHOLLIMA activity window; require re-verification of identity documents through a second channel before restoring privileged access.

Evidence: For the lessons-learned record, preserve and archive: (1) the complete dependency graph (`npm ls --all --json > dep-tree-snapshot.json`) from each affected project at time of discovery, documenting the axios v1.14.1/v0.30.4 infection vector; (2) HR and access provisioning records for any personnel onboarded into CI/CD privileged roles within 90 days prior to discovery, to support insider threat assessment aligned with FAMOUS CHOLLIMA's documented IT worker placement tactic; (3) a timeline of all GitHub repository access events (`GET

/orgs/{org}/audit-log?phrase=axios&include=git`) during the compromise window to determine whether threat actors pushed dependency changes or exfiltrated source code. These records support both internal review and any required regulatory disclosure.

Detection Guidance

Primary detection surface is dependency inventory. Run 'npm audit' and grep package-lock.json or yarn.lock files across all repositories for axios versions v1.14.1 and v0.30.4. In CI/CD pipeline logs, look for unexpected outbound HTTP/HTTPS requests to unfamiliar endpoints initiated by Node.js or npm install processes. On macOS developer systems, review process execution logs for anomalous child processes spawned by development tools, consistent with FAMOUS CHOLLIMA macOS implant behavior (T1059). In SIEM, correlate EDR telemetry with network flow data for periodic beaconing patterns, fixed-interval outbound connections to single external IPs from developer workstations or build servers. Review GitHub repository access logs for unauthorized pushes, token usage from unrecognized IPs, or newly added collaborators (T1195.001, T1195.002). Behavioral indicators of RAT activity include: unexpected process persistence entries, new scheduled tasks or launch agents on macOS, and exfiltration of .env files or credential stores. D3-SFA (System File Analysis) is applicable for monitoring configuration and credential files on developer endpoints. D3-LAM (Local Account Monitoring) applies to detecting unauthorized local account creation on build infrastructure consistent with insider threat activity.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://registry.npmjs.org/axios/-/axios-1.14.1.tgz`	Compromised axios npm package version v1.14.1 — do not install or use	HIGH
URL	`https://registry.npmjs.org/axios/-/axios-0.30.4.tgz`	Compromised axios npm package version v0.30.4 — do not install or use	HIGH

Framework Mappings

MITRE-ATTACK

- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1078** — Valid Accounts
- **T1199** — Trusted Relationship
- **T1486** — Data Encrypted for Impact
- **T1190** — Exploit Public-Facing Application
- **T1195.002** — Compromise Software Supply Chain
- **T1195** — Supply Chain Compromise
- **T1587.001** — Malware
- **T1072** — Software Deployment Tools
- **T1059** — Command and Scripting Interpreter

- **T1176** — Software Extensions
- **T1213** — Data from Information Repositories
- **T1566** — Phishing
- **T1591** — Gather Victim Org Information
- **T1110.003** — Password Spraying

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SR-2** — Supply Chain Risk Management Plan
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-8** — Spam Protection
- **CM-3** — Configuration Change Control

OWASP-TOP10-2021

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

CIS-V8

- **6.3** — Require MFA for Externally-Exposed Applications
- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **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
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1078	Valid Accounts	Defense-Evasion
T1199	Trusted Relationship	Initial-Access
T1486	Data Encrypted for Impact	Impact
T1190	Exploit Public-Facing Application	Initial-Access
T1195.002	Compromise Software Supply Chain	Initial-Access
T1195	Supply Chain Compromise	Initial-Access
T1587.001	Malware	Resource-Development
T1072	Software Deployment Tools	Execution
T1059	Command and Scripting Interpreter	Execution
T1176	Software Extensions	Persistence
T1213	Data from Information Repositories	Collection
T1566	Phishing	Initial-Access
T1591	Gather Victim Org Information	Reconnaissance
T1110.003	Password Spraying	Credential-Access

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/crowdstrike-2026-technology-...	T3

Source	URL	Tier
Axios NPM Package Compromised: Supply Chain Attack Hits ...	https://www.trendmicro.com/en_us/research/26/c/axios-npm-package-co...	T3
axios npm Compromised: RAT in v1.14.1 & v0.30.4 (2026)	https://phoenix.security/axios-supply-chain-compromise-npm-rat-2026/	T3
Post Mortem: axios npm supply chain compromise #10636 - GitHub	https://github.com/axios/axios/issues/10636	T3
The Axios Compromise: What Happened, What It Means ... - HeroDevs	https://www.herodevs.com/blog-posts/the-axios-compromise-what-happe...	T3

Generated 2026-06-14 04:22 UTC by TJS Security Command Center