INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-18 14:12 UTC

Technology Sector Under Siege: China Dominates State-Sponsored Intrusions While DPRK and eCrime Groups Escalate Supply Chain and Extortion Operations

THREAT CAMPAIGN | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0508
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	axios npm package (v1.14.1, v0.30.4), GitHub repositories, North American tech organizations, software development companies, mail infrastructure (East/Southeast Asia)
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 adversaries responsible for more than 58% of state-sponsored intrusions and DPRK-linked actors executing a high-impact supply chain attack against the axios npm package, used approximately 100 million times per week, by publishing malicious versions containing a Remote Access Trojan. Simultaneously, eCrime groups listed 572 technology organizations on extortion leak sites during the reporting period. Organizations building or shipping software face simultaneous exposure to intellectual property theft, supply chain poisoning, and ransomware extortion, with the axios compromise representing an immediate, high-severity threat to any development pipeline consuming the affected package versions.

Technical Analysis

DPRK-linked threat actors compromised the axios npm package by publishing two malicious versions, v1.14.1 and v0.30.4, containing an embedded Remote Access Trojan (RAT). Axios reports approximately 100 million weekly downloads, making this one of the highest-reach open-source supply chain attacks in recent memory. The compromise is corroborated by independent analysis from StepSecurity, Huntress, and Phoenix Security. No CVE has been assigned as of the session configuration date (2026-03-04); the attack maps to CWE-494 (Download of Code Without Integrity Check) and CWE-829 (Inclusion of Functionality from Untrusted Control

Sphere) for the supply chain vector, with CWE-287 (Improper Authentication) and CWE-522 (Insufficiently Protected Credentials) covering the broader campaign's authentication and credential exploitation. MITRE ATT&CK techniques include T1195.002 (Compromise Software Supply Chain), T1072 (Software Deployment Tools), T1059 (Command and Scripting Interpreter), T1078 (Valid Accounts), T1567 (Exfiltration Over Web Service), T1133 (External Remote Services), T1110.003 (Password Spraying), T1566 (Phishing), T1213 (Data from Information Repositories), T1588.006 (Vulnerabilities), T1486 (Data Encrypted for Impact), T1195.001 (Compromise Software Dependencies and Development Tools), and T1657 (Financial Theft). China-nexus adversaries focused on IP theft and software development infrastructure. eCrime actors conducted extortion operations, listing 572 tech organizations on leak sites. No patch version has been identified in the source material beyond removing or avoiding the compromised versions.

Action Checklist

1. Step 1: Containment (DevSecOps/Platform Engineers), Audit all package.json, package-lock.json, and yarn.lock files across every repository and build pipeline immediately. Identify any dependency, direct or transitive, on axios v1.14.1 or v0.30.4 and block installation of those versions at your artifact registry or package manager level. Isolate any build agents or developer workstations that resolved either version during the exposure window.
2. Step 2: Detection (Security Operations/SOC), Query runtime process logs, EDR telemetry, and network logs on any system that consumed the compromised axios versions for anomalous outbound connections, unexpected process spawning from Node.js processes, and RAT beacon patterns. Review npm audit logs and CI/CD pipeline build logs for installs of v1.14.1 or v0.30.4. Check SIEM for T1567 (unusual data exfiltration) and T1059 (script interpreter activity) indicators on affected build hosts. Reference StepSecurity, Huntress, and Phoenix Security advisories for published IOCs specific to the RAT payload.
3. Step 3: Eradication (DevSecOps), Pin axios to a verified clean version in all manifests and lock files. Run a full dependency integrity check using npm audit and verify package checksums against the npm registry's known-good hashes for clean versions. Remove compromised versions from any internal artifact mirrors or caches. Rotate any credentials, API keys, or tokens accessible to processes that ran on affected systems (NIST AC-2, CIS 5.2, D3-CRO: Credential Rotation). Rebuild all container images and deployment artifacts from clean base states.
4. Step 4: Recovery (Platform/DevOps), Re-deploy applications from freshly built, verified artifacts. Monitor production environments for 72 hours post-remediation for residual RAT activity using EDR and network behavioral baselines. Validate that artifact registries enforce version pinning and checksum verification going forward. Confirm CI/CD pipelines now fail builds on unresolved package integrity warnings (NIST AU-6, CIS 8.2).
5. Step 5: Post-Incident (Security Leadership/Engineering), Review software composition analysis (SCA) tooling coverage across all pipelines; if SCA was absent or not blocking, enforce it now. Evaluate whether your pipeline enforces package signing or integrity verification; CWE-494 and CWE-829 indicate a systemic gap in dependency trust. Map findings to NIST SP 800-53 SI-7 (Software, Firmware, and Information Integrity). Conduct a third-party dependency inventory review against CIS 2.1 (Establish and Maintain a Software Inventory) and CIS 2.2 (Ensure Authorized Software is Currently Supported). Brief development leadership on T1195.002 supply chain compromise risk and establish a recurring review cadence for high-download open-source dependencies.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to executive leadership, legal, and external IR retainer immediately if network logs or EDR telemetry confirm any outbound C2 connection from a host that resolved axios v1.14.1 or v0.30.4, if source code repositories or CI/CD secrets were accessible to processes running the RAT payload, or if the organization is subject to SOC 2, PCI DSS, or state breach notification laws and exfiltration of credentials or customer data cannot be ruled out within the initial triage window.
Recovery Notes	Re-deploy only from container images and deployment artifacts built entirely after the eradication window, with `npm ci --ignore-scripts` enforced and lock file integrity verified against the npm registry — do not promote any artifact whose build log cannot confirm axios resolved to a clean, pinned version. Monitor production Node.js process network activity against the RAT C2 IOC lists from StepSecurity, Huntress, and Phoenix Security advisories for a minimum of 72 hours post-redeployment, extending to 7 days if any system had confirmed exposure exceeding 24 hours. Verify that artifact registries block axios v1.14.1 and v0.30.4 at the ingress level and that CI/CD pipelines are configured to treat package integrity warnings as build-breaking failures before declaring recovery complete.
Forensic Artifacts	npm install logs at ~/.npm/_logs/ (Linux/Mac) or %APPDATA%\npm-cache_logs\ (Windows) — contain timestamped records of axios v1.14.1 or v0.30.4 resolution events, establishing which systems were exposed and when | Malicious axios tarballs preserved from ~/.npm/_cacache/content-v2/sha512// before cache purge — primary forensic sample of the DPRK-linked RAT payload embedded in the trojanized package, enabling hash comparison against IOCs and malware analysis | CI/CD pipeline build logs (GitHub Actions runner logs, Jenkins console output, GitLab CI job traces) — document every `npm install` invocation that resolved the compromised axios versions across the build fleet, scoping the blast radius to specific jobs, repos, and timestamps | Sysmon Event ID 1 (Process Create) and Event ID 3 (Network Connection) records where parent image is node.exe or node — capture child process spawning and outbound connection attempts initiated by the RAT payload executing within the Node.js runtime on affected build hosts | Docker image layer history (`docker history --no-trunc`) for all container images built during the exposure window — identifies which image layers incorporated the malicious axios tarball and which downstream production deployments inherited the compromise

Per-Action IR Details

Step 1: Containment — Audit all package.json, package-lock.json, and yarn.lock files across every repository and build pipeline immediately. Identify any dependency — direct or transitive — on axios v1.14.1 or v0.30.4 and block installation of those versions at your artifact registry or package manager level. Isolate any build agents or developer workstations that resolved either version during the exposure window.

NIST Phase: Containment

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

Controls: NIST AC-4 (Information Flow Enforcement), CIS 2.3 (Address Unauthorized Software)

Compensating: Run `grep -r 'axios' --include='package*.json' --include='yarn.lock' .` recursively across all repos to surface direct references; follow with `npm ls axios --depth=Infinity 2>/dev/null | grep '1\.14\.1\|0\.30\.4'` on each build agent to catch transitive pulls. In Verdaccio or Nexus OSS, add an explicit deny rule for `axios@1.14.1` and `axios@0.30.4` in the blocklist configuration. Network-isolate flagged build agents at the switch/VLAN level if host-level isolation is unavailable.

Evidence: Before isolating any build agent or workstation, capture: (1) full RAM image using WinPmem (Windows) or LiME (Linux) to preserve the in-memory RAT payload loaded by the malicious axios module; (2) active network connections via `netstat -ano` (Windows) or `ss -tunap` (Linux) to document any live C2 beacons established by the RAT before the host is cut off; (3) running process list with parent-child relationships via `Get-CimInstance Win32_Process | Select Name,ProcessId,ParentProcessId,CommandLine` (Windows) or `ps auxf` (Linux) to capture Node.js child processes spawned by the trojanized package; (4) snapshot of `~/.npm/_cacache` and any internal Verdaccio/Nexus artifact cache directories to preserve the malicious tarballs before they are evicted.

Step 2: Detection — Query runtime process logs, EDR telemetry, and network logs on any system that consumed the compromised axios versions for anomalous outbound connections, unexpected process spawning from Node.js processes, and RAT beacon patterns. Review npm audit logs and CI/CD pipeline build logs for installs of v1.14.1 or v0.30.4. Check SIEM for T1567 (unusual data exfiltration) and T1059 (script interpreter activity) indicators on affected build hosts. Reference StepSecurity, Huntress, and Phoenix Security advisories for published IOCs specific to the RAT payload.

NIST Phase: Detection Analysis

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

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

Compensating: Deploy Sysmon with a configuration that logs Event ID 1 (Process Create) and Event ID 3 (Network Connection) on all Node.js-capable hosts; filter for `node.exe` or `node` as parent process spawning `cmd.exe`, `sh`, `bash`, `powershell.exe`, or making outbound TCP connections to non-CDN IPs. Use `grep -E 'axios.*(1\.14\.1|0\.30\.4)' ~/.npm/_logs/*.log /var/log/npm-debug.log` to identify install timestamps. Apply published Sigma rules from the Huntress and StepSecurity advisories against locally collected Sysmon EVTX files using `sigma convert` + PowerShell or Chainsaw. For network detection, run Wireshark or `tcpdump -i any -w axios_capture.pcap` on build hosts and filter for beacon-interval outbound connections matching IOC IP/domain lists from the advisories.

Evidence: This is an analysis step, not a destructive action, so volatile state on hosts not yet isolated should still be preserved before any subsequent containment. Specific artifacts to collect and analyze: (1) npm install logs at `~/.npm/_logs/` (Linux/Mac) or `%APPDATA%\npm-cache\_logs\` (Windows) containing install timestamps and resolved version strings for axios; (2) CI/CD pipeline build logs (GitHub Actions runner logs, Jenkins build console output, GitLab CI job trace) for `npm install` lines resolving `axios@1.14.1` or `axios@0.30.4`; (3) Sysmon Event ID 3 records where `Image` matches `node.exe` or `node` and `DestinationIp` is not a known npm registry or CDN address — cross-reference against IOC IP lists published by StepSecurity and Huntress; (4) Node.js process `--inspect` or environment variable dumps if the RAT established a persistent foothold by modifying `node_modules/axios/lib/` files, recoverable from `/proc//environ` on Linux before process termination.

Step 3: Eradication — Pin axios to a verified clean version in all manifests and lock files. Run a full dependency integrity check using npm audit and verify package checksums against the npm registry's known-good hashes for clean versions. Remove compromised versions from any internal artifact mirrors or caches. Rotate any credentials, API keys, or tokens accessible to processes that ran on affected systems (NIST AC-2, CIS 5.2, D3-CRO: Credential Rotation). Rebuild all container images and deployment artifacts from clean base states.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST SI-2 — note: SI-2 (Flaw Remediation) is not present verbatim in the session knowledge base; verify against SP 800-53 Rev. 5 before citing in formal documentation, CIS 5.2 (Use Unique Passwords), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management)

Compensating: Verify package integrity manually: `npm pack axios@ --dry-run` and compare the `shasum` output against the integrity hash in your lock file using `npm ls --json | jq '.dependencies.axios.resolved'`. For credential rotation without a secrets manager, enumerate all `.env` files, CI/CD secret stores (GitHub Actions secrets, GitLab CI variables, Jenkins credentials store), and hardcoded config files on affected build hosts using `grep -rE

'(API_KEY|SECRET|TOKEN|PASSWORD)=' --include='*.env' --include='*.json' .` and treat every matched value as compromised. Purge Verdaccio or Nexus OSS cache by deleting the axios `1.14.1` and `0.30.4` directories under the storage path and restarting the registry service.

Evidence: Before rotating credentials and before rebuilding container images, capture: (1) a full list of environment variables accessible to Node.js processes on affected hosts via `/proc//environ` (Linux) or `Get-Process node | ForEach { [System.Diagnostics.Process]::GetProcessById($_.Id).StartInfo.EnvironmentVariables }` (Windows) — the RAT may have exfiltrated these at runtime; (2) Docker layer history via `docker history --no-trunc` for any image built during the exposure window to document which layer pulled the malicious axios tarball; (3) the malicious axios tarball itself from `~/.npm/_cacache/content-v2/sha512/` before cache purge — this is primary forensic evidence of the DPRK-linked RAT payload and must be preserved with chain of custody.

Step 4: Recovery — Re-deploy applications from freshly built, verified artifacts. Monitor production environments for 72 hours post-remediation for residual RAT activity using EDR and network behavioral baselines. Validate that artifact registries enforce version pinning and checksum verification going forward. Confirm CI/CD pipelines now fail builds on unresolved package integrity warnings (NIST AU-6, CIS 8.2).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Enforce integrity verification in CI/CD pipelines without enterprise tooling by adding `npm ci --ignore-scripts` (which uses the lock file exclusively and disables lifecycle scripts that the RAT may exploit) and `npm audit --audit-level=high` as required steps that exit non-zero on failure, blocking the build. For 72-hour post-remediation monitoring without EDR, deploy osquery with a pack querying `SELECT * FROM process_open_sockets WHERE remote_address NOT IN ()` on a 60-second interval to catch residual beacon activity from any RAT persistence mechanism that survived the rebuild.

Evidence: During the 72-hour monitoring window, continuously collect: (1) network flow data filtered for outbound connections from Node.js application processes to IPs and domains matching the RAT C2 IOCs published by StepSecurity and Huntress — residual beaconing would indicate the RAT achieved persistence beyond the package (e.g., via a planted `~/.npmrc` postinstall hook or a modified global npm script); (2) file integrity monitoring alerts on `node_modules/` directories in production deployments using `inotifywait` (Linux) or PowerShell `FileSystemWatcher` to detect any unauthorized modification to the clean axios package files post-deployment.

Step 5: Post-Incident — Review software composition analysis (SCA) tooling coverage across all pipelines; if SCA was absent or not blocking, enforce it now. Evaluate whether your pipeline enforces package signing or integrity verification — CWE-494 and CWE-829 indicate a systemic gap in dependency trust. Map findings to NIST SI-7 (Software, Firmware, and Information Integrity) — note: SI-7 is not in the session knowledge base, so verify the control ID against SP 800-53 Rev. 5 directly before citing it in formal documentation. Conduct a third-party dependency inventory review against CIS 2.1 (Establish and Maintain a Software Inventory) and CIS 2.2 (Ensure Authorized Software is Currently Supported). Brief development leadership on T1195.002 supply chain compromise risk and establish a recurring review cadence for high-download open-source dependencies.

NIST Phase: Post Incident

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

Controls: 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), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: For teams without commercial SCA tooling, integrate `npm audit` and `retire.js` as mandatory CI pipeline steps; add `npx better-npm-audit --level high` for richer output. Establish a free OSS dependency watchlist using deps.dev (Google's Open Source Insights) or socket.dev's free tier to monitor axios and other top-100 npm dependencies by download volume for new malicious version publications. Create a monthly cron job running `npm

outdated --json` across all repos and routing output to a shared channel to surface dependency drift against the approved version baseline.

Evidence: For the lessons-learned record and any required regulatory breach notification, preserve: (1) the complete timeline of axios v1.14.1 / v0.30.4 install events reconstructed from CI/CD build logs and npm install logs, establishing the precise exposure window for each affected system; (2) the output of `npm ls --json` snapshots taken from affected hosts during containment, documenting which applications had the malicious transitive dependency at the time of discovery; (3) any network flow or proxy logs showing outbound data transfers from affected build hosts to external IPs during the exposure window — this is material evidence for determining whether credential or source code exfiltration occurred and is required for breach notification threshold assessment.

Detection Guidance

Primary detection focus is on systems that installed or executed code from axios v1.14.1 or v0.30.4. Query package manager logs, CI/CD build outputs, and artifact pull records for references to those exact version strings. On endpoints and build agents where either version was installed, investigate: (1) anomalous outbound TCP connections from Node.js processes to non-standard ports or unfamiliar external IPs, consistent with RAT C2 beaconing (T1567, T1133); (2) unexpected child processes spawned by Node.js or npm, particularly interpreters or shell commands (T1059); (3) file system writes to temp directories or unusual paths by Node.js processes. Cross-reference IOCs published by StepSecurity (https://www.stepsecurity.io/blog/axios-compromised-on-npm-malicious-versions-drop-remote-access-trojan), Huntress (https://www.huntress.com/blog/supply-chain-compromise-axios-npm-package), and Phoenix Security (https://phoenix.security/axios-supply-chain-compromise-npm-rat-2026/). For the broader China-nexus campaign: monitor for T1078 (valid account misuse), T1110.003 (password spraying against VPNs and external services), and T1213 (unusual access to code repositories or documentation stores). CIS 8.2 (Collect Audit Logs) and NIST AU-6 (Audit Record Review, Analysis, and Reporting) provide the logging baseline required for this detection work.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.npmjs.com/package/axios/v/1.14.1`	Malicious axios npm package version containing embedded RAT — do not install; block at artifact registry	HIGH
URL	`https://www.npmjs.com/package/axios/v/0.30.4`	Malicious axios npm package version containing embedded RAT — do not install; block at artifact registry	HIGH

Framework Mappings

MITRE-ATTACK

- **T1567** — Exfiltration Over Web Service
- **T1133** — External Remote Services
- **T1195.002** — Compromise Software Supply Chain

- **T1078** — Valid Accounts
- **T1072** — Software Deployment Tools
- **T1059** — Command and Scripting Interpreter
- **T1566** — Phishing
- **T1110.003** — Password Spraying
- **T1213** — Data from Information Repositories
- **T1588.006** — Vulnerabilities
- **T1486** — Data Encrypted for Impact
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **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
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-8** — Spam Protection
- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **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
- **A04:2021** — Insecure Design

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
- **5.2** — Use Unique Passwords
- **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
- **164.308(a)(5)(ii)(D)** — Password Management

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
T1567	Exfiltration Over Web Service	Exfiltration
T1133	External Remote Services	Persistence
T1195.002	Compromise Software Supply Chain	Initial-Access
T1078	Valid Accounts	Defense-Evasion
T1072	Software Deployment Tools	Execution
T1059	Command and Scripting Interpreter	Execution
T1566	Phishing	Initial-Access
T1110.003	Password Spraying	Credential-Access
T1213	Data from Information Repositories	Collection
T1588.006	Vulnerabilities	Resource-Development
T1486	Data Encrypted for Impact	Impact

Technique ID	Technique Name	Tactic
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1657	Financial Theft	Impact

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/crowdstrike-2026-technology-...	T3
axios Compromised on npm - Malicious Versions Drop Remote ...	https://www.stepsecurity.io/blog/axios-compromised-on-npm-malicious...	T3
axios npm Compromised: RAT in v1.14.1 & v0.30.4 (2026)	https://phoenix.security/axios-supply-chain-compromise-npm-rat-2026/	T3
Supply Chain Compromise of axios npm Package - Huntress	https://www.huntress.com/blog/supply-chain-compromise-axios-npm-pac...	T3
Axios npm package targeted by supply chain attack - Facebook	https://www.facebook.com/groups/techtitansgroup/posts/1624708708856...	T3

Generated 2026-06-18 14:12 UTC by TJS Security Command Center