INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-15 13:49 UTC

China-Nexus and DPRK Actors Dominate Tech Sector Targeting; eCrime Extortion Reaches Record Volume (CrowdStrike 2026 Report)

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0464
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Technology sector broadly; GitHub repositories, Axios npm package (v1.14.1, v0.30.4), macOS platforms, mail infrastructure, private code repositories at unnamed software development company
Discovery Source	Rss:T1 Threatintel

Executive Summary

CrowdStrike's 2026 Technology Threat Landscape Report documents a converging dual threat against the technology sector: China-nexus actors drove more than 58% of state-sponsored intrusions targeting intellectual property and development infrastructure, while DPRK-affiliated actors conducted supply chain attacks including the confirmed compromise of the widely used Axios npm package. Simultaneously, eCrime groups named 572 technology organizations on extortion leak sites during the reporting period, a record volume. Technology companies now face concurrent nation-state espionage and financially motivated extortion requiring parallel, distinct defensive programs rather than a single unified response.

Technical Analysis

CrowdStrike's report (coverage: April 2025 - March 2026) identifies three dominant attack patterns. First, China-nexus adversaries (58%+ of state-sponsored tech intrusions per CrowdStrike) targeted code repositories and development infrastructure for persistent access and IP exfiltration, using valid account abuse (T1078) and external remote services (T1133). Second, DPRK-affiliated actors executed supply chain injection: the Axios npm package was compromised via a hijacked maintainer account, with malicious versions v1.14.1 and v0.30.4 delivering a remote access trojan (RAT). This attack aligns with CWE-494 (Download of Code Without Integrity Check) and MITRE T1195.002 (Compromise Software Supply Chain). The maintainer account takeover reflects CWE-306 (Missing Authentication for Critical Function) and T1586.002 (Compromise Accounts: Email Accounts). Password-based initial access (CWE-521, T1110.003) was a cross-campaign vector. Third, eCrime

actors used data exfiltration to extortion sites (T1567, T1567.001) and ransomware deployment (T1486) to pressure 572 named technology organizations. No CVE ID is assigned to the Axios compromise; the attack vector was account hijacking, not a software vulnerability. Affected package versions: axios v1.14.1 and v0.30.4 on npm. Safe versions: any release not matching those two strings; teams should verify package integrity via lockfile hash and npm audit.

Action Checklist

1. Containment, Immediately audit all package.json and lockfile dependencies across development and CI/CD pipelines for axios versions v1.14.1 and v0.30.4; quarantine any build artifact that included those versions and block their installation via npm registry allow-lists or private mirror policies.
2. Detection, Query endpoint logs and EDR telemetry for processes spawned by Node.js or npm install pipelines that exhibit outbound C2 behavior; review npm audit logs and CI/CD pipeline execution history for installs of axios v1.14.1 or v0.30.4; check SIEM for anomalous outbound connections from build servers (NIST AU-6, CIS 8.2); hunt for T1071 (Application Layer Protocol) beaconing from build or developer workstations.
3. Eradication, Downgrade or upgrade axios to a verified clean version confirmed by npm integrity hash; rotate all credentials (tokens, secrets, API keys) stored in repositories or CI/CD systems that may have been accessed by a compromised build environment (NIST IA-4, Credential Rotation); audit npm maintainer accounts across all internally published packages and enforce MFA on all registry accounts (CIS 6.3, NIST IA-2).
4. Recovery, Re-run all builds from source using verified dependency versions in an isolated environment; validate artifact integrity using lockfile checksums before redeploying to production; monitor post-remediation for RAT persistence indicators including scheduled tasks, modified startup configs (NIST SI-7, System Init Config Analysis), and unexpected outbound network sessions (NIST AU-6, CIS 8.2).
5. Post-Incident, Conduct a full software supply chain inventory to identify all third-party packages with maintainer single-points-of-failure; implement CWE-494 mitigations including subresource integrity checks and dependency pinning; enforce CIS 7.1 (vulnerability management process) and CIS 2.1 (software inventory) for all third-party libraries; establish a process for monitoring npm maintainer account changes on critical dependencies; document DPRK insider threat and account hijacking as live threat vectors in your threat model.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to CISO, legal counsel, and relevant sector ISAC (IT-ISAC) if any evidence confirms that exfiltrated CI/CD secrets (npm tokens, cloud provider keys, GitHub PATs) were actively used post-compromise, if production artifacts containing the malicious axios versions were deployed to customer-facing systems, or if forensic analysis identifies C2 callback activity consistent with DPRK-attributed infrastructure — any of these conditions triggers potential breach notification obligations and law enforcement referral.

Recovery Notes	All production systems and container images built using axios v1.14.1 or v0.30.4 must be treated as fully compromised until rebuilt from a verified clean dependency tree and redeployed from a clean CI/CD environment with rotated credentials; rebuild pipelines must use `npm ci` with network-isolated build containers to prevent postinstall script execution. Monitor all rebuilt systems for a minimum of 30 days post-remediation using Sysmon and network flow logs, specifically watching for low-and-slow beaconing patterns (sub-60-minute intervals) to non-RFC1918 addresses from Node.js processes, which is consistent with DPRK RAT persistence behavior documented in this campaign. Retain all forensic artifacts — npm cache tarballs, pipeline logs, memory captures — for a minimum of 12 months in accordance with NIST AU-11 (Audit Record Retention) and coordinate with legal counsel on preservation obligations before any destruction.
Forensic Artifacts	npm cache tarballs at `~/.npm/_cacache/` (Linux/macOS) or `%AppData%\npm-cache` (Windows) — the cached axios v1.14.1 or v0.30.4 tarball is the primary evidence artifact containing the DPRK-authored malicious postinstall script payload before cache eviction destroys it | CI/CD pipeline environment variable state at time of compromise — GitHub Actions secrets, GitLab CI variables, Jenkins credential store, or CircleCI context configs that were in scope during builds using the compromised axios versions, establishing the full credential exfiltration surface | npm debug logs at `~/.npm/_logs/*.log` recording the exact install invocation, resolved version, integrity hash verification result, and postinstall script execution for axios v1.14.1 or v0.30.4, with timestamps correlating to C2 beacon activity | Node.js process memory image (`gcore` / ProcDump output) from any build server where a Node.js process was active during or after the compromised install — in-memory artifacts of the malicious axios postinstall script, including decoded C2 URLs, exfiltration payloads, and injected shellcode, are only recoverable before process termination | Outbound network flow records (NetFlow, pcap, or firewall session logs) from build servers and developer workstations covering the 30-day window bracketing the first known install of the compromised axios version — DPRK implants in this campaign used application-layer protocol beaconing over HTTPS to blend with legitimate npm registry traffic, making flow timing analysis (periodic intervals, consistent byte counts) the primary network-layer detection signal

Per-Action IR Details

Containment — Immediately audit all package.json and lockfile dependencies across development and CI/CD pipelines for axios versions v1.14.1 and v0.30.4; quarantine any build artifact that included those versions and block their installation via npm registry allow-lists or private mirror policies.

NIST Phase: Containment

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

Controls: NIST CM-3 (Configuration Change Control), NIST SI-3 (Malicious Code Protection), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Run `grep -r 'axios' --include='package*.json' --include='package-lock.json' .` recursively across all repos to enumerate affected manifests. On Verdaccio or Nexus private mirrors, add axios@1.14.1 and axios@0.30.4 to a block list. Without a private mirror, create a pre-install npm hook script (`preinstall` in package.json scripts) that exits 1 if these exact versions are detected. Archive quarantined build artifacts to an isolated directory before deletion to preserve forensic state.

Evidence: Before quarantining any build artifact or blocking package installation, capture: (1) the npm cache directory (`~/.npm/_cacache/` on Linux/macOS, `%AppData%\npm-cache` on Windows) — the cached tarball of the compromised axios version may contain the malicious payload and is volatile once cache is cleared; (2) running process list from all build servers (`ps aux` / `Get-Process`) to identify any Node.js processes spawned from a compromised install; (3) active outbound network connections from build servers (`netstat -ano` or `ss -tnp`) before

network isolation is applied; (4) CI/CD pipeline execution logs (GitHub Actions workflow run logs, Jenkins build console output, GitLab CI job traces) capturing the exact install timestamps and resolved dependency trees for axios v1.14.1 or v0.30.4.

Detection — Query endpoint logs and EDR telemetry for processes spawned by Node.js or npm install pipelines that exhibit outbound C2 behavior; review npm audit logs and CI/CD pipeline execution history for installs of axios v1.14.1 or v0.30.4; check SIEM for anomalous outbound connections from build servers (NIST AU-6, CIS 8.2); hunt for T1071 (Application Layer Protocol) beaconing from build or developer workstations.

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-2 (Event Logging), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Without SIEM/EDR: (1) Deploy Sysmon with a config that captures Event ID 3 (Network Connection) and Event ID 1 (Process Create) — filter for `node.exe` or `node` as parent process making outbound connections to non-RFC1918 addresses. (2) Run `npm audit --json > audit_results.json` in every repo to flag known-malicious versions. (3) Parse npm debug logs (`~/.npm/_logs/` or `%AppData%\npm\_logs\`) for install events matching axios v1.14.1 or v0.30.4. (4) Use Wireshark or `tcpdump -i any -w build_server_capture.pcap` on build servers during a pipeline run to capture outbound traffic for offline C2 beacon analysis. (5) Search shell history (`~/.bash_history`, `~/.zsh_history`) on developer workstations for npm install commands referencing the compromised versions.

Evidence: This is a detection/analysis step that does not itself alter live state; however, if live Node.js processes are running on build servers, capture before any remediation: (1) full memory image of the Node.js process (`gcore ` on Linux or ProcDump on Windows) to recover in-memory payload injected by the malicious axios postinstall script; (2) `netstat -ano` / `ss -tnp` output correlating PID to the suspicious outbound connection; (3) `/proc//fd/` symlinks on Linux to enumerate open file handles from the compromised process; (4) npm debug log at `~/.npm/_logs/` recording the exact install invocation with timestamp, resolved version, and integrity hash mismatch (if any) for axios v1.14.1 or v0.30.4; (5) CI/CD environment variable dump if the pipeline was running at time of detection — DPRK supply chain implants targeting CI/CD commonly exfiltrate `CI_JOB_TOKEN`, `NPM_TOKEN`, `GITHUB_TOKEN`, and AWS/GCP credential env vars injected into the build context.

Eradication — Downgrade or upgrade axios to a verified clean version confirmed by npm integrity hash; rotate all credentials (tokens, secrets, API keys) stored in repositories or CI/CD systems that may have been accessed by a compromised build environment (D3-CRO — Credential Rotation); audit npm maintainer accounts across all internally published packages and enforce MFA on all registry accounts (D3-MFA, CIS 6.3, NIST IA controls).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST AC-2 (Account Management), NIST AC-6 (Least Privilege), NIST IA-5 (Authenticator Management), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access)

Compensating: Credential rotation without enterprise PAM: (1) Use `git log --all --full-history -p -- '**/.env' '**/*.env' '**/secrets*'` to surface any credentials committed to repo history, then revoke each via the issuing platform's API (GitHub PAT revocation, AWS IAM key deletion, npm token revocation via `npm token revoke `). (2) Audit npmjs.com maintainer accounts for all internally published packages via `npm owner ls ` — remove unrecognized maintainers immediately. (3) Enable npmjs.com 2FA enforcement using the `--auth-type=legacy` CLI flag with a TOTP app; for registry accounts without native MFA, enforce login via a hardware key using `npm login --auth-type=webauthn` where supported. (4) Run `truffleHog --regex --entropy=True` or `gitleaks detect` against all repositories to enumerate secrets that may have been exfiltrated during the compromised build.

Evidence: Before rotating any credential or revoking any token — actions that permanently alter authentication state and may destroy evidence of attacker reuse — capture: (1) current active session tokens and API key last-used timestamps from the issuing platform (GitHub Security log, AWS CloudTrail `GetCallerIdentity` events, npm audit log) to establish whether the attacker has already leveraged exfiltrated credentials; (2) CI/CD system environment variable

configuration (masked values aside, capture variable names and scopes) to document the full credential exposure surface; (3) git reflog and `git log --stat` for any repository the compromised build environment had write access to, preserving evidence of unauthorized commits or tag manipulation consistent with DPRK supply chain persistence tactics; (4) npm publish history (`npm info time`) for all internally maintained packages to detect unauthorized version publications made using exfiltrated npm tokens during the window of compromise.

Recovery — Re-run all builds from source using verified dependency versions in an isolated environment; validate artifact integrity using lockfile checksums before redeploying to production; monitor post-remediation for RAT persistence indicators including scheduled tasks, modified startup configs (D3-SICA — System Init Config Analysis), and unexpected outbound network sessions (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), NIST CM-3 (Configuration Change Control), CIS 8.2 (Collect Audit Logs), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Without enterprise artifact signing infrastructure: (1) Validate package integrity manually — `npm ci` (not `npm install`) enforces lockfile-only installs; cross-reference the `integrity` SHA-512 hash in `package-lock.json` against the official npmjs.com registry manifest via `npm view axios@ dist.integrity`. (2) Run builds in a fresh Docker container (`docker run --rm --network none node: npm ci`) with network disabled to prevent any residual malicious postinstall script from reaching C2. (3) Monitor post-remediation persistence using Sysmon Event ID 13 (RegistryValueSet) for `HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` and `HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run` modifications, and audit `/etc/cron.d/`, `/etc/cron.daily/`, and user crontabs (`crontab -l -u `) on Linux build servers for entries added during the compromise window. (4) Use `osquery` with the `scheduled_tasks` and `startup_items` tables to baseline and diff startup configuration state before and after remediation.

Evidence: Before redeploying any artifact to production — an action that introduces potentially contaminated code into production and makes post-incident forensic attribution significantly harder — capture: (1) a filesystem hash baseline of the build server using `sha256sum -r /opt/app/ > baseline_hashes.txt` or `Get-FileHash -Recurse` to document pre-remediation state for legal hold purposes; (2) cron/scheduled task configuration snapshots from all build servers (`crontab -l`, `schtasks /query /fo LIST /v > tasks_before_remediation.txt`) to establish whether a RAT postinstall hook persisted a scheduled callback consistent with DPRK implant behavior; (3) full list of outbound firewall connections from the build environment over the 30-day window preceding discovery, correlated against known DPRK C2 infrastructure indicators published by CISA and CrowdStrike; (4) Docker layer history (`docker history `) for any container images built using the compromised axios versions to determine if malicious layers were baked into distributed images.

Post-Incident — Conduct a full software supply chain inventory to identify all third-party packages with maintainer single-points-of-failure; implement CWE-494 mitigations including subresource integrity checks and dependency pinning; enforce CIS 7.1 (vulnerability management process) and CIS 2.1 (software inventory) for all third-party libraries; establish a process for monitoring npm maintainer account changes on critical dependencies; document DPRK insider threat and account hijacking as live threat vectors in your threat model.

NIST Phase: Post Incident

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

Controls: NIST AU-11 (Audit Record Retention), 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 a 2-person team with no enterprise tooling: (1) Generate a full SBOM using `cyclonedx-npm --output sbom.json` or `syft . -o cyclonedx-json` to enumerate every transitive dependency with its maintainer and last-publish timestamp — flag any package with a single maintainer who published within 30 days of the Axios compromise window. (2) Implement dependency pinning via `npm shrinkwrap` or enforcing `package-lock.json` commits with a pre-commit hook that rejects lockfile drift. (3) Subscribe to npm package change notifications using

`npm-watch` or the socket.dev free tier, which flags maintainer account changes, new publish events, and permission grants for packages in your dependency tree — directly relevant to the DPRK account-hijacking vector used against axios. (4) Add DPRK IT worker and supply chain compromise indicators from CISA Advisory AA23-347A and CrowdStrike's 2026 report to your threat model as named actor TTPs, not generic supply chain risk.

Evidence: Post-incident documentation phase does not alter live system state; however, before closing the incident, ensure the following are preserved for legal hold and lessons-learned: (1) complete timeline of axios v1.14.1 and v0.30.4 install events extracted from npm debug logs and CI/CD pipeline records, with first-install and last-install timestamps establishing the full exposure window; (2) preserved copies of the malicious package tarballs from npm cache if not already quarantined — these are primary forensic evidence of the DPRK-authored payload and may be requested by law enforcement or sector ISACs (e.g., IT-ISAC); (3) inventory of all production artifacts (container images, compiled binaries, deployed Lambda functions) built during the compromise window that must be tracked for downstream incident notification obligations; (4) documented maintainer account audit results for all internally published npm packages, establishing the pre- and post-incident authorization baseline for future anomaly detection.

Detection Guidance

Priority detection focus areas: (1) Supply chain compromise, search CI/CD and build server logs for npm install events referencing axios@1.14.1 or axios@0.30.4; cross-reference lockfile hashes against known-good values published by the axios project. (2) RAT activity, hunt for anomalous outbound TCP sessions originating from Node.js processes or developer workstations, particularly to non-standard ports or newly registered domains (T1071); check NIST AU-6 alerts for new local accounts or privilege changes on build servers post-install. (3) Credential compromise, review authentication logs for unusual access patterns to code repositories, npm registry, or CI/CD platforms, including off-hours logins and access from unexpected geolocations (NIST AU-6, T1078); alert on T1110.003 (Password Spraying) patterns against developer identity providers. (4) Exfiltration precursors, monitor for large outbound data transfers from private repository hosts or developer endpoints (T1567, T1567.001); correlate with file access logs on code repositories (NIST AU-2, AU-3). (5) Insider/DPRK patterns, flag anomalous contractor or new-hire access to sensitive code repositories; apply NIST SI-7 (System File Monitoring) to detect modification of build scripts or pipeline configuration files. Behavioral indicator: a build environment that installs a package and immediately initiates an outbound connection to an external host is high-confidence RAT activity.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`axios@1.14.1 (npm)`	Malicious axios npm package version confirmed to deliver RAT via hijacked maintainer account; verify against lockfile integrity hash from official axios project	HIGH
HASH	`axios@0.30.4 (npm)`	Second malicious axios npm package version confirmed in same supply chain compromise event	HIGH

Framework Mappings

MITRE-ATTACK

- **T1567** — Exfiltration Over Web Service
- **T1567.001** — Exfiltration to Code Repository
- **T1486** — Data Encrypted for Impact
- **T1059** — Command and Scripting Interpreter
- **T1078** — Valid Accounts
- **T1195.002** — Compromise Software Supply Chain
- **T1071** — Application Layer Protocol
- **T1608.003** — Install Digital Certificate
- **T1586.002** — Email Accounts
- **T1588.001** — Malware
- **T1566** — Phishing
- **T1133** — External Remote Services
- **T1110.003** — Password Spraying

NIST-800-53R5

- **CP-9** — System Backup
- **CP-10** — System Recovery and Reconstitution
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **AC-17** — Remote Access
- **AC-20** — Use of External Systems
- **CM-3** — Configuration Change Control
- **SR-2** — Supply Chain Risk Management Plan

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
- **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
T1567	Exfiltration Over Web Service	Exfiltration
T1567.001	Exfiltration to Code Repository	Exfiltration
T1486	Data Encrypted for Impact	Impact
T1059	Command and Scripting Interpreter	Execution
T1078	Valid Accounts	Defense-Evasion
T1195.002	Compromise Software Supply Chain	Initial-Access
T1071	Application Layer Protocol	Command-And-Control
T1608.003	Install Digital Certificate	Resource-Development
T1586.002	Email Accounts	Resource-Development
T1588.001	Malware	Resource-Development
T1566	Phishing	Initial-Access
T1133	External Remote Services	Persistence
T1110.003	Password Spraying	Credential-Access

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 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
Axios compromised: hijacked maintainer account pushes malicious ...	https://www.endorlabs.com/learn/npm-axios-compromise	T3

Generated 2026-06-15 13:49 UTC by TJS Security Command Center