INTELLIGENCE BRIEFING
Security Command Center

TLP:CLEAR
2026-06-18 07:15 UTC

China and DPRK Drive 2025-2026 Technology Sector Targeting Wave: Supply Chains, AI Assets, and IT Worker Fraud at the Core

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0505
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Technology sector broadly; npm ecosystem (Axios package v1.14.1 and v0.30.4); GitHub repositories; North American tech organizations; mail infrastructure
Discovery Source	Rss:T1 Threatintel

Executive Summary

CrowdStrike's 2026 Technology Threat Landscape Report documents a coordinated, multi-vector assault on the technology sector spanning April 2025 through March 2026: China-nexus actors drove more than 58% of state-sponsored intrusions, primarily targeting intellectual property and AI capabilities, while DPRK-affiliated operatives placed fraudulent employees inside tech firms and trojanized the widely-deployed Axios npm package (versions 1.14.1 and 0.30.4) with a remote access trojan. Simultaneously, eCrime groups listed 572 technology organizations on ransomware extortion leak sites, a volume exceeding every other sector. Organizations that build software, hold AI research, or employ contract developers face converging risks: espionage, insider access, compromised build dependencies, and extortion, often targeting the same environment.

Technical Analysis

This campaign item aggregates three concurrent threat vectors documented in CrowdStrike's 2026 Technology Threat Landscape Report and corroborated by Trend Micro, Orca Security, and the axios GitHub issue tracker.

Supply Chain, Axios npm Compromise (CWE-829, CWE-494): Axios versions v1.14.1 and v0.30.4 were trojanized with a remote access trojan (RAT) prior to publication on the npm registry. The injected code constitutes inclusion of functionality from an untrusted control sphere (CWE-829) and download of code without integrity check (CWE-494). Downstream projects that resolved ^1.14.x or pinned to v0.30.4 pulled the malicious package automatically. The compromise leveraged Axios's broad JavaScript ecosystem adoption to maximize blast radius. MITRE technique: T1195.002 (Compromise Software Supply Chain).

Credential Access, Password Spraying (CWE-521): SUNRISE PANDA and related China-nexus actors (tracked under the MURKY PANDA campaign identifier) conducted password spraying (T1110.003) against mail and collaboration infrastructure. Weak password requirements (CWE-521) enabled access; post-compromise activity included data exfiltration from information repositories (T1213), consistent with IP and AI asset theft objectives.

Insider Placement, DPRK IT Worker Fraud (T1078): DPRK-affiliated actors placed fraudulent employees at North American tech organizations, establishing valid account access (T1078) from inside the perimeter. This vector bypasses most network-layer controls and provides persistent access to source code repositories, collaboration tools, and internal systems.

GitHub Repository Injection (T1608.001, CWE-829): Malicious GitHub repositories were used as staging or injection points for additional payload delivery, consistent with adversary-controlled infrastructure staging (T1583.003).

eCrime Extortion: 572 technology organizations were named on extortion leak sites during the reporting period, indicating parallel ransomware operations targeting the same sector, separate from nation-state activity but operating against the same asset pool.

No CVE ID is assigned to this campaign item. This campaign aggregates state-sponsored and eCrime activities across multiple vectors; qualitative severity is 'high' based on scope and impact. No CISA KEV entry exists as of the configuration date.

Action Checklist

1. Step 1: Assessment & Containment, Audit all JavaScript/Node.js projects and CI/CD pipelines for dependency on Axios. Immediately identify any build that resolved v1.14.1 or v0.30.4 from the npm registry. Isolate affected build artifacts and any systems where those artifacts were deployed. Do not run or promote builds produced while the malicious versions were available without re-vetting the dependency tree.
2. Step 2: Detection, Query npm lock files (package-lock.json, yarn.lock, pnpm-lock.yaml) across all repositories for axios versions 1.14.1 or 0.30.4. Search SIEM and EDR telemetry for outbound connections initiated by Node.js processes to unexpected external IPs following axios package installation events. Review build system logs (GitHub Actions, Jenkins, CircleCI) for axios install timestamps that fall within the compromise window. Behavioral indicators: unexpected RAT-style beaconing from application servers, new scheduled tasks or cron jobs created post-deployment, lateral movement from application service accounts. Apply AU-6 (Audit Record Review, Analysis, and Reporting) to prioritize log review for affected build pipelines. Map to D3-SFA (System File Analysis) for review of installed package files.
3. Step 3: Eradication, Upgrade Axios to a verified clean version as confirmed in the axios GitHub issue tracker (issue #10636) and Trend Micro and Orca Security advisories. Rebuild all affected artifacts from clean source after dependency pinning is corrected. Rotate all secrets, tokens, and credentials accessible to any system that ran the compromised package. For password spraying exposure: enforce password complexity and account lockout thresholds per NIST AC-7 (Unsuccessful Logon Attempts) and CIS 5.2 (Use Unique Passwords). Audit Active Directory and Entra ID for accounts created or modified during the intrusion window.
4. Step 4: Recovery, Validate remediated builds by confirming Axios package hash against the npm registry integrity field for the clean version. Re-scan production artifacts with a software composition analysis (SCA) tool after upgrade. Monitor application telemetry for 30 days post-remediation for residual

RAT activity. Apply AU-11 (Audit Record Retention) to preserve logs covering the full compromise window for forensic use. Verify MFA enforcement on all externally-exposed collaboration and mail systems per CIS 6.3 (Require MFA for Externally-Exposed Applications) and CIS 6.4 (Require MFA for Remote Network Access).

5. Step 5: Post-Incident Review, Conduct a control gap review against CIS 7.1 (Vulnerability Management Process) and CIS 7.4 (Automated Application Patch Management), assess whether dependency scanning was in place before the compromise. Evaluate whether DPRK IT worker fraud risk is addressed in your third-party and contractor vetting process; consult CISA and FBI guidance on DPRK IT worker fraud (published 2024-2025) for hiring control recommendations. Implement or tune software composition analysis in CI/CD pipelines to flag unsigned or integrity-mismatched packages before build promotion. Review code repository access logs against known employee roster per CIS 5.3 (Disable Dormant Accounts) and AC-6 (Least Privilege) to identify anomalous access patterns consistent with insider placement.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to CISO, legal, and external IR retainer if: any confirmed C2 beaconing from production systems running axios v1.14.1 or v0.30.4 is observed; any Entra ID or Active Directory account cannot be reconciled to a verified employee or contractor (potential DPRK insider placement); or if intellectual property repositories, AI model weights, or customer PII/PHI were accessible from any system that executed the compromised axios build, triggering breach notification obligations under applicable state, federal, or sector-specific regulations.
Recovery Notes	Promote only builds that were fully rebuilt from clean source after dependency pinning was corrected and whose axios package hash is verified against the npm registry integrity field for the confirmed clean version — do not trust cached artifacts from the compromise window. Sustain active monitoring for 30 days post-remediation using Sysmon NetworkConnect events or auditd connect syscall rules scoped to Node.js processes, as the Axios-embedded RAT may have installed secondary persistence (cron, scheduled tasks, or systemd units) that survives package removal and could re-establish C2 independently. Confirm that all DPRK-attributed contractor or IT worker accounts identified during the AD/Entra ID audit have been fully offboarded, including revocation of SSH keys, GitHub tokens, and any OAuth application consents they granted during their tenure.

Forensic Artifacts	Trojanized axios package tarballs: SHA-256 hash of the cached npm tarballs for axios v1.14.1 and v0.30.4 from `~/.npm/_cacache` (Linux/macOS) or `%AppData%\npm-cache` (Windows) and the unpacked `node_modules/axios/` directory — primary malware artifact tying the host to the supply chain compromise | Sysmon Event ID 3 (NetworkConnect) logs scoped to node.exe / node processes: documents C2 beaconing destinations, ports, and timing intervals characteristic of the Axios-embedded RAT's check-in behavior following package installation | CI/CD build pipeline logs (GitHub Actions, Jenkins, CircleCI): records the exact timestamp when axios v1.14.1 or v0.30.4 was resolved and installed, establishing which downstream artifact versions are tainted and the full exposure window | Entra ID Sign-in logs and Unified Audit Log (UAL) entries for contractor and third-party accounts: captures authentication anomalies, MFA registration events, and OAuth consent grants by potentially DPRK-placed IT workers, constituting insider threat forensic evidence | Linux cron and Windows Task Scheduler event logs (Windows Event IDs 4698 and 4702) from application servers post-axios deployment: documents persistence mechanisms installed by the RAT after initial execution, which may survive package removal and re-establish C2 independently

Per-Action IR Details

Step 1: Containment — Audit all JavaScript/Node.js projects and CI/CD pipelines for dependency on Axios. Immediately identify any build that resolved v1.14.1 or v0.30.4 from the npm registry. Isolate affected build artifacts and any systems where those artifacts were deployed. Do not run or promote builds produced while the malicious versions were available without re-vetting the dependency tree.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: isolate affected systems and artifacts to prevent further spread of the Axios-embedded RAT before eradication actions alter live state

Controls: NIST AC-4 (Information Flow Enforcement) — restrict outbound network flows from systems running compromised Axios builds until clean artifacts are confirmed, CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — enumerate all Node.js application hosts and build pipelines that may have pulled axios v1.14.1 or v0.30.4, CIS 2.1 (Establish and Maintain a Software Inventory) — identify axios as a licensed/tracked dependency and flag the two malicious versions across all project manifests

Compensating: Run `grep -r '"axios": "1.14.1\|0.30.4"' /path/to/repos --include='package*.json'` and `find . -name 'package-lock.json' -exec grep -l 'axios' {} \;` across all local repository clones. For CI/CD systems without centralized tooling, pull GitHub Actions workflow run logs via `gh run list --limit 200` and grep for axios install events. Block outbound traffic from application servers using host-based iptables rules (`iptables -A OUTPUT -p tcp --dport 443 -m owner --uid-owner nodejs -j DROP`) as an emergency measure until artifact vetting is complete.

Evidence: BEFORE isolating any application server or disabling any running Node.js process: capture full RAM image using WinPmem (Windows) or `avml` (Linux) to preserve in-memory RAT state, active C2 socket descriptors, and any injected code not present on disk. Capture `netstat -ano` / `ss -tulnp` output and map PIDs to Node.js processes. Record `lsof -p ` to document open file handles and network connections held by the compromised axios module. Document running process tree with `ps auxf` (Linux) or `Get-CimInstance Win32_Process | Select-Object ProcessId,ParentProcessId,CommandLine` (Windows) before any isolation action.

Step 2: Detection — Query npm lock files (package-lock.json, yarn.lock, pnpm-lock.yaml) across all repositories for axios versions 1.14.1 or 0.30.4. Search SIEM and EDR telemetry for outbound connections initiated by Node.js processes to unexpected external IPs following axios package installation events. Review build system logs (GitHub Actions, Jenkins, CircleCI) for axios install timestamps that fall within the compromise window. Behavioral indicators: unexpected RAT-style beaconing from application servers, new scheduled tasks or cron jobs created post-deployment, lateral movement from application service accounts. Apply AU-6 (Audit Record Review, Analysis, and Reporting) to prioritize log review for affected build pipelines. Map to D3-SFA (System File Analysis) for review of installed package files.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: correlate npm install telemetry, build pipeline logs, and network beaconing patterns to determine scope of axios RAT deployment and identify C2 communication channels

Controls: NIST AU-6 (Audit Record Review, Analysis, and Reporting) — review build pipeline logs and application server network logs for axios install timestamps and post-install outbound connection anomalies, NIST AU-2 (Event Logging) — ensure Node.js process-level network events and npm install activity are captured in audit log scope, NIST AU-3 (Content Of Audit Records) — verify that captured logs contain source IP, destination IP, process name, and timestamp sufficient to reconstruct axios RAT beaconing timeline

Compensating: Without SIEM: use `jq '.dependencies.axios.version' package-lock.json` to batch-check all repos. Install Sysmon with a config that captures NetworkConnect events (Event ID 3) filtered to processes named `node.exe` or `node`; export with `Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object {$_.Id -eq 3 -and $_.Message -like '*node*'}`. On Linux, use `auditd` rules: `auditctl -a always,exit -F arch=b64 -S connect -F comm=node -k axios_c2`. For cron persistence dropped by the RAT, run `for user in $(cut -f1 -d: /etc/passwd); do crontab -u $user -l 2>/dev/null; done` and compare against a pre-compromise baseline. Use YARA to scan the axios package directory (`~/.npm` cache and `node_modules/axios/lib/`) for embedded RAT payloads — write a rule matching known C2 callback strings from Trend Micro and Orca Security advisories.

Evidence: This step is analytical and does not alter live state, but preserve volatile artifacts before any follow-on containment actions triggered by findings. Key evidence sources: (1) npm cache directory (`~/.npm/_cacache` or `%AppData%\npm-cache`) — the cached axios tarball for v1.14.1/v0.30.4 is the primary malware artifact; hash it with SHA-256 immediately. (2) `node_modules/axios/` directory tree on all affected hosts — diff against the clean axios package manifest from the official axios GitHub repository. (3) Sysmon Event ID 3 (NetworkConnect) and Event ID 11 (FileCreate) logs from application servers covering the compromise window. (4) GitHub Actions workflow run artifacts and logs for any job that executed `npm install` or `npm ci` during the compromise window — downloadable via GitHub API before retention expiry. (5) `/var/log/cron` and Windows Task Scheduler event logs (Event IDs 4698, 4702) for persistence mechanisms installed by the RAT post-deployment.

Step 3: Eradication — Upgrade Axios to a verified clean version as confirmed in the axios GitHub issue tracker (issue #10636) and Trend Micro and Orca Security advisories. Rebuild all affected artifacts from clean source after dependency pinning is corrected. Rotate all secrets, tokens, and credentials accessible to any system that ran the compromised package. For password spraying exposure: enforce password complexity and account lockout thresholds per NIST AC-7 (Unsuccessful Logon Attempts) and CIS 5.2 (Use Unique Passwords). Audit Active Directory and Entra ID for accounts created or modified during the intrusion window.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: remove the trojanized axios package from all environments, invalidate all credentials exposed to the RAT's access scope, and verify the threat actor has no remaining foothold via AD/Entra ID accounts created during the intrusion window

Controls: NIST AC-7 (Unsuccessful Logon Attempts) — enforce lockout thresholds to close residual password-spraying attack surface identified in the campaign, NIST AC-2 (Account Management) — audit and disable any accounts created or modified in Active Directory or Entra ID during the compromise window consistent with DPRK IT worker insider placement, CIS 5.2 (Use Unique Passwords) — enforce unique, complex credentials for all service accounts and developer accounts that had access to systems running compromised axios builds, CIS 7.4 (Perform Automated Application Patch Management) — enforce pinned, verified dependency versions in CI/CD pipelines to prevent re-introduction of malicious axios versions

Compensating: Verify the clean axios package integrity before deployment: `npm view axios@ dist.integrity` and compare to the npm registry integrity field. Pin the dependency in package.json with an exact version (`"axios": "="`) and commit a `package-lock.json` generated from a clean network-isolated build environment. For credential rotation on a small team: use `az ad user list --query '[].{UPN:userPrincipalName,Created:createdDateTime}' --output table` to enumerate Entra ID accounts and cross-reference against HR roster for DPRK-placed insider accounts. For AD: `Get-ADUser -Filter * -Properties WhenCreated,WhenChanged | Where-Object {$_.WhenCreated -gt ''}` to surface accounts created during the compromise window.

Evidence: BEFORE rotating any credential, token, or secret: export a complete credential inventory snapshot — list all OAuth tokens, API keys, and service account credentials accessible from affected systems using your secrets manager audit log (AWS CloudTrail `GetSecretValue` events, HashiCorp Vault audit log, or Azure Key Vault diagnostic logs). BEFORE reimaging or wiping any affected build host: acquire a full disk image and memory capture. BEFORE disabling any suspected DPRK-placed insider account: preserve a read-only export of the account's full audit trail from Entra ID Sign-in logs and Unified Audit Log (UAL) — specifically authentication history, MFA registration events, and any OAuth application consents granted — as this constitutes insider threat forensic evidence. Capture `secretsdump.py` or `ntdsutil` snapshot of NTDS.DIT only if AD compromise is confirmed and only after memory acquisition.

Step 4: Recovery — Validate remediated builds by confirming Axios package hash against the npm registry integrity field for the clean version. Re-scan production artifacts with a software composition analysis (SCA) tool after upgrade. Monitor application telemetry for 30 days post-remediation for residual RAT activity. Apply AU-11 (Audit Record Retention) to preserve logs covering the full compromise window for forensic use. Verify MFA enforcement on all externally-exposed collaboration and mail systems per CIS 6.3 (Require MFA for Externally-Exposed Applications) and CIS 6.4 (Require MFA for Remote Network Access).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore production systems from verified clean builds, confirm integrity of the axios dependency chain, and sustain monitoring for 30 days to detect RAT re-activation or residual DPRK/China-nexus persistence mechanisms

Controls: NIST AU-11 (Audit Record Retention) — retain all logs covering the full axios compromise window (April 2025 through detection date) to support forensic reconstruction and any regulatory notification obligations, CIS 6.3 (Require MFA for Externally-Exposed Applications) — enforce MFA on all externally-exposed collaboration and mail systems targeted by the campaign's mail infrastructure compromise vector, CIS 6.4 (Require MFA for Remote Network Access) — enforce MFA for all remote access paths, specifically targeting developer VPN and remote build system access used by potentially fraudulent DPRK-placed contractors, CIS 7.4 (Perform Automated Application Patch Management) — validate that the remediated axios version is propagated across all production artifact registries via automated patch management

Compensating: Validate clean axios package hash without SCA tooling: `shasum -a 512 node_modules/axios/dist/axios.min.js` and compare against the value published in the axios GitHub release for the clean version. For 30-day residual RAT monitoring without EDR: deploy a Sigma rule targeting Node.js processes initiating outbound connections to non-RFC1918 addresses on non-standard ports — convert to auditd rules or Windows Event Log queries using `sigma convert`. Archive all relevant logs to immutable storage immediately: `aws s3 cp /var/log/syslog s3:///incident-$(date +%Y%m%d)/ --sse` or equivalent. For MFA verification on mail systems, use Microsoft Secure Score or Google Workspace Admin console MFA enforcement reports to confirm coverage without additional tooling.

Evidence: Preserve pre-recovery baseline: before promoting any remediated build to production, snapshot current application server network telemetry (active connections, listening ports, running process list) so post-recovery monitoring has a clean baseline to diff against. Retain all npm audit logs, CI/CD build logs, and application server logs covering the full compromise window in write-once storage — these are required for forensic chain of custody and potential regulatory breach notification. For residual RAT detection during the 30-day monitoring window, the primary indicators are: periodic outbound HTTP/S beaconing from Node.js processes at regular intervals (characteristic of RAT check-in), new cron entries or scheduled tasks not present in pre-compromise configuration baselines, and unexpected DNS queries from application servers to newly-registered domains.

Step 5: Post-Incident — Conduct a control gap review against CIS 7.1 (Vulnerability Management Process) and CIS 7.4 (Automated Application Patch Management) — assess whether dependency scanning was in place before the compromise. Evaluate whether DPRK IT worker fraud risk is addressed in your third-party and contractor vetting process; consult CISA and FBI guidance on DPRK IT worker fraud (published 2024–2025) for hiring control recommendations. Implement or tune software composition analysis in CI/CD pipelines to flag unsigned or integrity-mismatched packages before build promotion. Review code repository

access logs against known employee roster per CIS 5.3 (Disable Dormant Accounts) and AC-6 (Least Privilege) to identify anomalous access patterns consistent with insider placement.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: conduct lessons-learned review to close dependency scanning and contractor vetting gaps exploited by DPRK IT worker fraud, and update detection capabilities to identify trojanized npm packages and insider-threat access patterns before the next campaign cycle

Controls: CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — assess whether a formal vulnerability management process covering open-source dependency integrity would have detected the trojanized axios package before deployment, CIS 7.4 (Perform Automated Application Patch Management) — evaluate whether automated application patch management with SCA integration would have flagged axios v1.14.1 or v0.30.4 as integrity-mismatched before build promotion, CIS 5.3 (Disable Dormant Accounts) — review code repository access for accounts inactive or anomalous relative to the known employee roster, specifically targeting access patterns consistent with DPRK IT worker insider placement, NIST AC-6 (Least Privilege) — audit repository and CI/CD pipeline permissions to ensure contractor and third-party accounts had only the minimum access required, limiting the blast radius of insider-placed DPRK operatives, CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts) — verify that any contractor or IT worker accounts identified as potentially DPRK-placed did not hold administrative privileges on build systems or production repositories

Compensating: For SCA without enterprise tooling: integrate `npm audit` and `npm audit signatures` (verifies package registry signatures) as a mandatory CI/CD gate — add `npm audit --audit-level=high && npm audit signatures` as a required step in GitHub Actions workflows before any build promotion. For DPRK IT worker insider detection without DLP: export GitHub repository access logs via `gh api /repos///events --paginate` and cross-reference committer email addresses and GitHub usernames against the HR system roster; flag accounts with no corresponding HR record. Reference CISA Advisory AA23-230A and FBI PIN 20240515-001 for specific DPRK IT worker behavioral indicators to include in contractor onboarding screening checklists.

Evidence: For the post-incident review, the primary forensic artifacts to analyze are: (1) GitHub and GitLab access logs covering the intrusion window — specifically commits, branch protections bypassed, and secrets or environment variable access by contractor accounts; (2) Entra ID / Active Directory account creation and modification logs for the full campaign window (April 2025 through detection) to assess DPRK insider footprint; (3) CI/CD pipeline build history showing which builds consumed axios v1.14.1 or v0.30.4 and which downstream systems received those artifacts; (4) npm audit log from the package registry for both malicious versions to establish the full exposure timeline. These artifacts should be archived as part of the incident record before any systems are decommissioned or logs rotated.

Detection Guidance

npm Supply Chain (Axios RAT):

- Search all repository lock files for `axios` at versions `1.14.1` or `0.30.4`. Tools: `npm ls axios`, `grep -r 'axios' package-lock.json`, or SCA platform queries.

- In build logs, identify timestamps when these versions were installed; cross-reference with deployment records to determine affected runtime environments.

- On hosts running Node.js applications built with compromised versions, hunt for: unexpected outbound TCP connections from `node` processes to non-application IPs, new persistence mechanisms (cron, systemd units, scheduled tasks) created post-deployment, and file writes to temp directories by the application process.

- EDR query pattern (generic): process_name = 'node' AND network_connection_initiated = true AND destination_ip NOT IN [known_application_endpoints].

Password Spraying (Mail/Collaboration Infrastructure):

- Review Entra ID / Azure AD sign-in logs and on-premises authentication logs for: single-source IPs attempting authentication against many accounts with low failure-per-account ratios (classic spray pattern), authentication attempts outside business hours from foreign ASNs, and successful logins followed immediately by mail rule creation or forwarding rule additions.

- SIEM logic: COUNT(failed_logins) BY source_ip WHERE unique_target_accounts > 10 AND failures_per_account < 3 WITHIN 1 hour.

DPRK IT Worker Insider:

- Behavioral indicators: contractor accounts accessing source code repositories outside assigned project scope, bulk repository clone operations, unusual data egress to personal cloud storage (Google Drive, Dropbox, GitHub personal accounts), VPN or remote access originating from unexpected geolocations.

- Apply D3-LAM (Local Account Monitoring) to track newly created contractor accounts and their access patterns against peer baselines.

- Review CIS 5.1 (Inventory of Accounts) to ensure all contractor accounts are documented; flag undocumented accounts for immediate review.

GitHub Repository Injection:

- Audit CI/CD pipeline configurations for references to third-party GitHub repositories not on an approved vendor list.

- Flag any recently added or modified dependency source references in workflow files (`.github/workflows/*.yml`).

Map all detections against MITRE ATT&CK T1195.002, T1110.003, T1078, T1213, T1608.001, and T1583.003 for coverage gap analysis.

Indicators of Compromise

Type	Value	Context	Confidence
HASH	`axios@1.14.1 (npm)`	Trojanized Axios npm package version containing embedded RAT; confirmed via Trend Micro, Orca Security, and axios GitHub issue #10636	HIGH
HASH	`axios@0.30.4 (npm)`	Trojanized Axios npm package version containing embedded RAT; confirmed via same sources	HIGH
URL	`https://github.com/axios/axios/issues/10636`	Official axios project post-mortem confirming the supply chain compromise and affected versions	HIGH

Framework Mappings

MITRE-ATTACK

- **T1110.003** — Password Spraying

- **T1608.001** — Upload Malware
- **T1195.002** — Compromise Software Supply Chain
- **T1566** — Phishing
- **T1657** — Financial Theft
- **T1213** — Data from Information Repositories
- **T1078** — Valid Accounts
- **T1583.003** — Virtual Private Server
- **T1133** — External Remote Services
- **T1071** — Application Layer Protocol

NIST-800-53R5

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

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **15.1** — Establish and Maintain an Inventory of Service Providers

NIST-CSF-2

- **RS.MI-01** — Incidents are contained
- **GV.SC-01** — Cybersecurity supply chain risk management program

HIPAA-SECURITY

- **164.308(a)(7)(ii)(A)** — Data Backup Plan

ISO-27001-2022

- **A.5.29** — Information security during disruption
- **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
T1110.003	Password Spraying	Credential-Access
T1608.001	Upload Malware	Resource-Development
T1195.002	Compromise Software Supply Chain	Initial-Access
T1566	Phishing	Initial-Access
T1657	Financial Theft	Impact
T1213	Data from Information Repositories	Collection
T1078	Valid Accounts	Defense-Evasion
T1583.003	Virtual Private Server	Resource-Development
T1133	External Remote Services	Persistence
T1071	Application Layer Protocol	Command-And-Control

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

Source	URL	Tier
Post Mortem: axios npm supply chain compromise #10636 - GitHub	https://github.com/axios/axios/issues/10636	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 2026-06-18 07:15 UTC by TJS Security Command Center