INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-12 18:29 UTC

Dual Supply Chain Attacks Compromise Trivy, Axios, and LiteLLM Open-Source Tools

THREAT CAMPAIGN | CRITICAL | CVSS 9.0

SCC Item ID	SCC-CAM-2026-0169
Type	Threat Campaign
Severity	CRITICAL
CVSS Base Score	9.0
Affected Products	Trivy (vulnerability scanner), Axios (JavaScript HTTP client), LiteLLM (AI/LLM proxy library), specific compromised versions not confirmed in available data
Published	2026-04-11
Discovery Source	Gemini

Executive Summary

Two separate threat actors, including a group tracked as TeamPCP, compromised three widely used open-source developer tools in March 2026: Trivy (a container and code vulnerability scanner), Axios (a JavaScript HTTP client), and LiteLLM (an AI/LLM proxy library). Attackers injected malicious code into published package releases by exploiting compromised maintainer accounts or poisoned release pipelines, targeting CI/CD environments to steal API keys, tokens, and credentials from downstream organizations. Organizations that consumed affected versions of these packages during the compromise window should treat their pipeline secrets and downstream systems as potentially exposed.

Technical Analysis

Two distinct threat actors conducted coordinated supply chain attacks against Trivy, Axios, and LiteLLM in March 2026. TeamPCP, attributed by Arctic Wolf, targeted Trivy, Checkmarx KICS, and LiteLLM. A separate actor targeted Axios. Attack vectors included maintainer account compromise (T1078), credential theft (T1552), and poisoned software update/release paths (T1195.001, T1195.002). Malicious payloads were embedded in published package releases and designed to execute automatically within CI/CD pipelines (T1059), exfiltrating secrets via outbound command-and-control channels (T1071). The weaponization of Trivy is particularly significant: organizations relying on Trivy scan output for security decisions were directly targeted, undermining trust in scanner results produced during the compromise window. Relevant CWEs: CWE-1357 (improper protection of infrastructure components), CWE-506 (embedded malicious code), CWE-255 (credentials management errors), CWE-494 (download of code without integrity check). Specific compromised version ranges have not been confirmed in available open-source reporting as of this writing. Severity assessed as

critical based on attack scope and credential exposure risk. No CVE identifier assigned. No CISA KEV listing at time of writing. Confidence: MEDIUM, full technical payload details and confirmed IOCs are not yet publicly available. Sources: Arctic Wolf (TeamPCP attribution), Legit Security (Trivy weaponization analysis), The Register (2026-04-11).

Action Checklist

1. Step 1: Containment. Immediately audit your package manifests, lockfiles, and CI/CD pipeline dependencies for Trivy, Axios, and LiteLLM. Identify which versions were pulled during March 2026. If any pipeline ran with a potentially compromised version, treat all secrets, API keys, and tokens accessible to that pipeline as compromised and rotate them now. Isolate affected build environments pending investigation.
2. Step 2: Detection. Review CI/CD pipeline logs (GitHub Actions, Jenkins, GitLab CI, etc.) for unexpected outbound network connections, unusual process execution, or secret access events during and after March 2026 package pulls. Query SIEM for outbound traffic from build agents to unfamiliar external hosts. Review secret manager access logs (AWS Secrets Manager, HashiCorp Vault, Azure Key Vault) for anomalous reads coinciding with pipeline runs. Check for unexpected npm, pip, or container registry pulls against your SBOM. Note: specific IOC values (IPs, domains, hashes) are not confirmed in available open-source data; monitor Arctic Wolf and Legit Security advisories for updates.
3. Step 3: Eradication. Pin all three packages to versions verified as clean by the respective maintainers or your internal SBOM review. For Trivy, verify against the official Aqua Security release page and GitHub release signatures. For Axios, verify against the official npm registry and GitHub release history. For LiteLLM, verify against the official PyPI release history. Rebuild all container images and pipeline artifacts from verified clean base versions. Enable package integrity verification (checksums, signed releases) in your dependency management tooling.
4. Step 4: Recovery. After rotating secrets and rebuilding pipelines from clean versions, validate that no exfiltrated credentials remain active in downstream systems. Audit access logs across cloud providers, SaaS platforms, and internal services for anomalous access using the rotated credentials. Re-run security scans of pipeline artifacts using a clean, verified scanner instance. Re-establish baseline behavioral monitoring for CI/CD environments.
5. Step 5: Post-Incident. This attack exposed the gap between trusting scanner output and verifying scanner integrity. Implement a software supply chain security control baseline: enforce SLSA framework levels for critical pipeline dependencies, require signed releases, pin dependency versions with hash verification, and scan SBOMs for dependency provenance. Review whether your security tooling itself (scanners, linters, security libraries) is subject to the same dependency integrity controls as production software. Consider adopting a dedicated supply chain security tool (e.g., Sigstore, in-toto) for pipeline artifact verification.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE

Escalation Criteria	Escalate to CISO, legal counsel, and breach notification counsel immediately if CloudTrail, Azure Monitor, or GCP audit logs confirm that exfiltrated API keys or tokens were used to access systems storing PII, PHI, or payment card data, or if any cloud IAM privilege escalation events are detected post-compromise, as these conditions trigger regulatory notification obligations under GDPR Article 33, HIPAA Breach Notification Rule, or PCI DSS Requirement 12.10.4.
Recovery Notes	After rotating all secrets and rebuilding pipelines from hash-verified clean package versions, maintain elevated monitoring on cloud provider IAM and API gateway logs for a minimum of 30 days, as TeamPCP-harvested credentials may have been staged for delayed use or sold to secondary threat actors. Re-validate that all downstream systems (internal APIs, SaaS integrations, cloud service accounts) that authenticated using any credential accessible to affected pipelines show no anomalous access patterns using the original credential values. Do not stand down monitoring until all rotated credential IDs are confirmed inactive across every integrated system and a clean SBOM baseline has been formally accepted for all rebuilt pipeline artifacts.
Forensic Artifacts	npm registry integrity hashes: the 'integrity' field (SHA-512, base64-encoded) in package-lock.json for the Axios entry pulled during March 2026 — compare against the authoritative value at 'https://registry.npmjs.org/axios/' to confirm whether a tampered tarball was fetched | PyPI download records: the exact .whl or .tar.gz filename and SHA-256 hash for the LiteLLM version installed in March 2026, recoverable from pip's HTTP cache (~/.cache/pip/http/) or pipeline stdout logs — submit to PyPI security and Legit Security for cross-reference against known-bad hashes once IOC data is published | CI/CD runner process execution records: ephemeral runner logs or auditd/syslog entries capturing child processes spawned by node (Axios), python (LiteLLM), or the trivy binary during March 2026 pipeline runs — specifically any execve calls to curl, wget, python -c, or base64 that indicate in-process payload execution characteristic of malicious postinstall scripts | Secret manager access audit trails: AWS CloudTrail 'GetSecretValue' events, HashiCorp Vault audit log 'read' operations on secret paths, or Azure Key Vault 'SecretGet' events timestamped within seconds of the compromised package installation step in pipeline logs — the temporal correlation between package install and secret read is the primary forensic indicator of credential harvesting by the injected malicious code | Container image layer manifests: 'docker history --no-trunc ' and 'docker inspect ' output for all images built during March 2026, preserving the exact layer digests and RUN commands that installed Trivy, Axios, or LiteLLM — these establish the forensic chain of custody linking specific image builds to the compromised package versions and support blast radius scoping across all deployment targets

Per-Action IR Details

Step 1: Containment — Immediately audit your package manifests, lockfiles, and CI/CD pipeline dependencies for Trivy, Axios, and LiteLLM. Identify which versions were pulled during March 2026. If any pipeline ran with a potentially compromised version, treat all secrets, API keys, and tokens accessible to that pipeline as compromised and rotate them now. Isolate affected build environments pending investigation.

NIST Phase: Containment

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

Controls: NIST IR-4 (Incident Handling), NIST CM-3 (Configuration Change Control), NIST IA-5 (Authenticator Management), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Run 'cat package-lock.json | grep -A2 axios', 'cat Pipfile.lock | grep litellm', and 'grep -r trivy .github/workflows/' across all repos to enumerate exact pulled versions. For container-based build agents, run 'docker

history ' to identify the layer that installed the compromised package. Revoke and rotate secrets immediately using CLI: 'aws secretsmanager rotate-secret --secret-id ' or equivalent Vault CLI 'vault lease revoke -prefix '. Snapshot the build agent filesystem (e.g., 'tar czf /tmp/build-agent-evidence.tar.gz /home/runner /tmp /var/log') before teardown.

Evidence: Before isolating build environments, preserve: (1) package-lock.json, Pipfile.lock, poetry.lock, and go.sum files showing the exact resolved version hashes for Axios, LiteLLM, and Trivy pulled in March 2026; (2) CI/CD runner ephemeral filesystem snapshots or Docker layer manifests capturing the installed package state; (3) pipeline execution logs from GitHub Actions (~/.github/runner/_diag/), Jenkins (JENKINS_HOME/jobs//builds//log), or GitLab CI (/var/log/gitlab-runner/) covering the March 2026 window; (4) secret manager audit trails from AWS CloudTrail (event: GetSecretValue), HashiCorp Vault audit log (/var/log/vault/audit.log, operation: read), or Azure Monitor (SecretGet operations) timestamped to pipeline run windows.

Step 2: Detection — Review CI/CD pipeline logs (GitHub Actions, Jenkins, GitLab CI, etc.) for unexpected outbound network connections, unusual process execution, or secret access events during and after March 2026 package pulls. Query SIEM for outbound traffic from build agents to unfamiliar external hosts. Review secret manager access logs (AWS Secrets Manager, HashiCorp Vault, Azure Key Vault) for anomalous reads coinciding with pipeline runs. Check for unexpected npm, pip, or container registry pulls against your SBOM. Note: specific IOC values (IPs, domains, hashes) are not confirmed in available open-source data; monitor Arctic Wolf and Legit Security advisories for updates.

NIST Phase: Detection Analysis

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

Controls: NIST IR-5 (Incident Monitoring), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), NIST SI-4 (System Monitoring), CIS 8.2 (Collect Audit Logs), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without SIEM, use osquery to query process execution on self-hosted runners: 'SELECT pid, name, cmdline, parent, start_time FROM processes WHERE parent IN (SELECT pid FROM processes WHERE name IN ("node", "python", "trivy"))' — this surfaces child processes spawned by compromised Axios (node), LiteLLM (python), or Trivy binaries. For network detection without EDR, deploy Wireshark or tcpdump on the build agent subnet with a capture filter targeting established connections from the runner IP range to non-RFC1918 addresses during pipeline execution windows: 'tcpdump -i eth0 -w /tmp/pipeline-cap.pcap "src net and not dst net (10.0.0.0/8 or 172.16.0.0/12 or 192.168.0.0/16)"'. Parse GitHub Actions workflow logs for unexpected 'Run' steps or curl/wget invocations using: 'grep -E "(curl|wget|Invoke-WebRequest|nc |ncat)" .github/workflows/*.yml' and compare against approved workflow definitions. Apply the Sigma rule 'process_creation_susp_network_connection_via_certutil' pattern logic manually against Jenkins build logs for analogous exfiltration indicators.

Evidence: Capture before analysis: (1) GitHub Actions workflow run logs via API ('GET /repos/{owner}/{repo}/actions/runs?created=>2026-03-01') to retrieve all runs that installed Axios, Trivy, or LiteLLM, including runner IP and timing; (2) VPC Flow Logs or NSG flow logs filtered to build agent subnet for March 2026, specifically egress connections on ports 443, 80, and non-standard ports from runner IPs; (3) AWS CloudTrail logs filtered for 'GetSecretValue', 'GetParameter' (SSM), and 'AssumeRole' events originating from build agent IAM roles during pipeline run timestamps — these would reflect TeamPCP's credential harvest from environment variables or mounted secret stores; (4) npm audit logs (~/.npm/_logs/) and pip install stdout captured in pipeline logs showing the exact package content hash fetched from registry at install time; (5) container registry pull logs (Docker Hub, ECR, GCR) for the base images used during March 2026 builds to identify if poisoned Trivy was pulled as a scanner sidecar.

Step 3: Eradication — Pin all three packages to versions verified as clean by the respective maintainers or your internal SBOM review. For Trivy, verify against the official Aqua Security release page and GitHub release signatures. For Axios, verify against the official npm registry and GitHub release history. For LiteLLM, verify against the official PyPI release history. Rebuild all container images and pipeline artifacts from verified clean base versions. Enable package integrity verification (checksums, signed releases) in your dependency management tooling.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication

Controls: NIST SI-2 (Flaw Remediation), NIST SI-7 (Software, Firmware, and Information Integrity), NIST CM-3 (Configuration Change Control), CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Verify Trivy release integrity using cosign: 'cosign verify-blob --certificate trivy__checksums.txt.pem --signature trivy__checksums.txt.sig trivy__checksums.txt' against Aqua Security's published signing certificate. For Axios, compare the npm tarball SHA-512 integrity field in package-lock.json against the value published in the official npm registry API ('curl https://registry.npmjs.org/axios/' and inspect the 'dist.integrity' field). For LiteLLM, run 'pip download litellm== --no-deps -d /tmp/litellm-verify && sha256sum /tmp/litellm-verify/*.whl' and compare against PyPI's published hash ('curl https://pypi.org/pypi/litellm//json | jq .urls[].digests'). Rebuild affected Docker images using 'docker build --no-cache' to force re-pull from verified registry sources, and validate with 'docker scan' or a locally verified Trivy binary from a clean, air-gapped install.

Evidence: Before rebuilding, preserve as eradication evidence: (1) the exact SHA-256 or SHA-512 hash of the compromised package tarballs pulled from npm (Axios) and PyPI (LiteLLM) as recorded in lockfiles, to submit to Legit Security, Arctic Wolf, or CISA for IOC confirmation; (2) container image digests ('docker inspect --format={{.Id}} ') for all images built with the compromised packages, to support future threat hunting if IOC data is later published; (3) a snapshot of the poisoned pipeline workflow YAML files (GitHub Actions .yml, Jenkinsfile, .gitlab-ci.yml) before any remediation changes, preserving the dependency resolution chain for forensic record.

Step 4: Recovery — After rotating secrets and rebuilding pipelines from clean versions, validate that no exfiltrated credentials remain active in downstream systems. Audit access logs across cloud providers, SaaS platforms, and internal services for anomalous access using the rotated credentials. Re-run security scans of pipeline artifacts using a clean, verified scanner instance. Re-establish baseline behavioral monitoring for CI/CD environments.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST IR-4 (Incident Handling), NIST IA-5 (Authenticator Management), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-2 (Flaw Remediation), CIS 5.2 (Use Unique Passwords), CIS 6.3 (Require MFA for Externally-Exposed Applications)

Compensating: For credential validation without enterprise PAM tooling, query GitHub's token meta-endpoint to confirm old tokens are revoked: 'curl -H "Authorization: token " https://api.github.com/user' — a 401 response confirms revocation. For AWS, run 'aws iam list-access-keys --user-name ' to confirm old key IDs are in 'Inactive' status and audit recent usage with 'aws cloudtrail lookup-events --lookup-attributes AttributeKey=Username,AttributeValue= --start-time 2026-03-01'. Re-scan rebuilt pipeline artifacts with a Trivy binary installed from a clean, hash-verified tarball on an isolated workstation: 'trivy image --ignore-unfixed ' — do NOT use a Trivy instance pulled from any pipeline that ran during March 2026. Establish a lightweight behavioral baseline for build agents using auditd rules targeting execve syscalls for node, python, and curl: 'auditctl -a always,exit -F arch=b64 -S execve -F uid= -k pipeline-exec'.

Evidence: During recovery validation, collect: (1) cloud provider access logs (AWS CloudTrail, GCP Cloud Audit Logs, Azure Activity Log) for all API calls made using credentials that were accessible to pipelines running the compromised packages — look specifically for CreateRole, PutBucketPolicy, CreateUser, or similar privilege escalation events that would indicate TeamPCP leveraged harvested credentials beyond initial exfiltration; (2) SaaS application access logs (GitHub audit log via 'GET /orgs/{org}/audit-log', Slack admin audit, Jira audit trail) for sessions authenticated with tokens that were exposed to the compromised pipeline environment during March 2026; (3) confirmation records (timestamps, API responses) documenting successful revocation of each rotated secret, to support post-incident reporting under NIST IR-6 (Incident Reporting) obligations.

Step 5: Post-Incident — This attack exposed the gap between trusting scanner output and verifying scanner integrity. Implement a software supply chain security control baseline: enforce SLSA framework levels for critical pipeline dependencies, require signed releases, pin dependency versions with hash verification, and scan SBOMs for dependency provenance. Review whether your security tooling itself (scanners, linters, security libraries) is subject to the same dependency integrity controls as production software. Adopt a dedicated supply chain security tool (e.g., Sigstore, in-toto) for pipeline artifact verification.

NIST Phase: Post Incident

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

Controls: NIST IR-4 (Incident Handling), NIST IR-8 (Incident Response Plan), NIST SI-7 (Software, Firmware, and Information Integrity), NIST SA-12 (Supply Chain Protection), CIS 7.2 (Establish and Maintain a Remediation Process), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Implement Sigstore's cosign for signing and verifying all pipeline-produced container images at no cost: add 'cosign sign --key cosign.key ' as a required pipeline step and 'cosign verify --key cosign.pub ' as a required gate before deployment. Generate SBOMs for all pipeline artifacts using Syft ('syft -o spdx-json > sbom.json') and scan them with Grype ('grype sbom:sbom.json') — both are free OSS tools — establishing a policy that security tooling (Trivy, linters, SAST tools) must pass the same SBOM and hash verification gates as production dependencies. Document the lessons-learned finding that Trivy itself was weaponized in this campaign, and add a standing playbook entry requiring that any scanner used in CI/CD must itself be verified via signed release before use, checked against Aqua Security's published cosign signatures on every pipeline initialization.

Evidence: For the post-incident lessons-learned record, preserve: (1) the complete SBOM snapshots (SPDX or CycloneDX format) from all affected pipelines covering the March 2026 window, establishing the dependency graph that allowed TeamPCP's poisoned packages to reach production build environments; (2) a comparative diff of pipeline workflow definitions before and after remediation, documenting exactly which dependency pinning and integrity verification controls were absent and have now been added; (3) the full incident timeline (first compromised package pull → detection → containment → eradication) with timestamps, to support regulatory notification assessment and to feed threat intelligence back to CISA, the npm security team, and PyPI's security disclosure process per NIST IR-6 (Incident Reporting) and NIST DE.AE-08 incident declaration criteria.

Detection Guidance

Confirmed IOCs (specific IPs, domains, file hashes) are not available in open-source reporting as of this writing; treat this guidance as behavioral and structural detection rather than indicator-based. Query CI/CD pipeline logs for outbound connections from build agents to external hosts during pipeline runs, particularly any connections not matching known registry or artifact endpoints. Review npm audit logs, pip install logs, and container pull history for Trivy, Axios, and LiteLLM pulls dated to March 2026. In your SIEM, correlate pipeline execution timestamps with secret manager access events (AWS CloudTrail: GetSecretValue, Vault audit logs, Azure Monitor). Flag any pipeline run that accessed secrets followed by an outbound connection to an unrecognized host. For Trivy specifically, review scan result integrity: if scan results were produced during the compromise window, treat them as potentially tampered and re-run with a verified clean version. Monitor Arctic Wolf's published IOC list for TeamPCP indicators as their investigation matures.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`[not confirmed in available open-source data]`	Specific C2 domains used by TeamPCP or the Axios attacker have not been publicly released as of this writing. Monitor Arctic Wolf's advisory for updates.	LOW

Framework Mappings

MITRE-ATTACK

- **T1078** — Valid Accounts
- **T1552** — Unsecured Credentials
- **T1195.001** — Compromise Software Dependencies and Development Tools
- **T1071** — Application Layer Protocol
- **T1059** — Command and Scripting Interpreter
- **T1195.002** — Compromise Software Supply Chain

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-7** — Software, Firmware, and Information Integrity
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **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

HIPAA-SECURITY

- **164.308(a)(5)(ii)(D)** — Password Management
- **164.312(d)** — Person or Entity Authentication

CIS-V8

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

SOC2-TSC

- **CC6.1** — Logical access security software, infrastructure, and architectures
- **CC9.2** — Manages risks associated with vendors and business partners

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1078	Valid Accounts	Defense-Evasion
T1552	Unsecured Credentials	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1071	Application Layer Protocol	Command-And-Control
T1059	Command and Scripting Interpreter	Execution
T1195.002	Compromise Software Supply Chain	Initial-Access

Sources

Source	URL	Tier
Two different attackers poisoned popular open source tools	https://www.theregister.com/2026/04/11/trivy_axios_supply_chain_att...	T3
When Your Scanner Becomes the Weapon: From Trivy to ...	https://www.legitsecurity.com/blog/when-your-scanner-becomes-the-we...	T3
Open Source Projects Compromised: Axios, LiteLLM, Trivy ...	https://www.linkedin.com/posts/michakaufman_three-major-open-source...	T3
TeamPCP Supply Chain Attack Campaign Targets Trivy ...	https://arcticwolf.com/resources/blog/teampcp-supply-chain-attack-c...	T3
5 major supply chain attacks in 2 weeks (Trivy, LiteLLM ...	https://www.reddit.com/r/CRACompliance/comments/1sfo94y/5_major_sup...	T3

Generated 2026-04-12 18:29 UTC by TJS Security Command Center