CSAI Foundation

Cloud Security Alliance AI Safety Initiative

TeamPCP Supply Chain Cascade: When Security Tools Become Attack Infrastructure

Unofficial AI-assisted Research

2026-04-02

Key Takeaways

- Between March 19 and March 27, 2026, the threat group TeamPCP executed a cascading supply chain attack that compromised four widely deployed security and AI tooling packages — Trivy, Checkmarx KICS, LiteLLM, and the Telnyx Python SDK — in a single coordinated campaign [1][2].
- The attack exploited a defining structural vulnerability: security scanning tools occupy privileged positions in CI/CD pipelines, where they have simultaneous access to cloud credentials, code repositories, and container infrastructure. Compromising a scanner can expose every secret accessible within that pipeline context — often including cloud credentials, source repository tokens, and publishing keys [3].
- LiteLLM's compromise demonstrated a second structural risk: AI gateway libraries aggregate credentials for dozens of LLM providers into a single dependency, creating an exceptionally high-value target. A single poisoned version yielded concurrent access to API keys for OpenAI, Anthropic, Google, Cohere, and others across approximately 95–97 million monthly download instances [5].
- The attack introduced novel persistence via Python `.pth` files — a mechanism that survives package uninstallation — and documented the first operational use of a decentralized blockchain canister as command-and-control infrastructure, significantly complicating conventional takedown efforts [2][5].
- CVE-2026-33634 (CVSS 9.4) was assigned to the Trivy vector; CISA issued a joint advisory on March 27 [6]. Organizations that ran affected package versions should treat all associated credentials as compromised and rotate them, including LLM API keys.

Background

TeamPCP, also tracked by researchers under the aliases PCPcat, ShellForce, and DeadCatx3, is a financially motivated threat group with documented activity stretching back to at least September 2025 [1]. The group originated in cryptocurrency mining and theft, transitioned to ransomware operations, and in early 2026 pivoted to large-scale supply chain compromise as its primary revenue model. On March

30, TeamPCP announced a formal partnership with the Vect ransomware-as-a-service operation, signaling a further monetization shift: stolen credentials harvested through supply chain attacks would now serve as the initial access vector for downstream ransomware deployment [7].

The group's selection of targets was not arbitrary. Aqua Security's Trivy is among the most widely deployed container vulnerability scanners, embedded by default across a broad range of Kubernetes and CI/CD environments. Checkmarx KICS performs infrastructure-as-code security analysis in pipelines across thousands of organizations. LiteLLM functions as a universal AI gateway — an abstraction layer that lets applications route to any of more than 100 LLM providers through a single dependency. Each target occupied a structural chokepoint: tools that by design touch everything a pipeline touches. Compromising these tools did not require social engineering users, exploiting application logic, or defeating authentication mechanisms. It required compromising the tools themselves, then waiting for legitimate builds to pull and execute the malicious code.

The campaign's starting point was an incomplete credential rotation. In late February 2026, TeamPCP obtained credentials for the `aqua-bot` GitHub service account used to maintain Trivy's official repositories [1][3]. Aqua Security detected the intrusion but rotated only some of the affected secrets. The overlooked residual access became the launchpad for everything that followed.

Security Analysis

Wave One: Trivy and the Privilege of the Scanner (March 19)

On March 19, using the residual `aqua-bot` credentials, TeamPCP force-pushed malicious commits to 76 of 77 version tags in `aquasecurity/trivy-action` and all 7 tags in `aquasecurity/setup-trivy` [3][4]. Because git tags in GitHub Actions workflows are mutable by default — meaning a tag like `v0.20` can be silently repointed to a different commit — any pipeline pinned to a version tag rather than a full commit SHA was immediately and transparently compromised. The infected Trivy binary v0.69.4 was simultaneously published to official distribution channels.

The payload architecture reflected deliberate tradecraft. A 150-line bash script executed before legitimate scan logic, querying the AWS Instance Metadata Service at `169.254.169.254` to harvest cloud credentials, then scraping SSH keys, Slack and Discord webhook URLs, and environment variables [1]. A 15-line loader then retrieved a Python second stage (`kube.py`) which deployed CanisterWorm — a self-replicating component capable of enumerating Docker APIs and deploying Kubernetes DaemonSets across victim clusters [2]. All exfiltrated data was encrypted with AES-256-CBC and an

RSA-4096 session key wrap before transmission to the typosquatted domain `scan.aquasecurtiy[.]org` (note the transposed letters), identifiable in network logs by the distinctive HTTP header `X-Filename: tpcp.tar.gz` [1].

The scale of direct exposure was substantial. Researchers estimated that 474 public repositories executed the malicious `trivy-action`, based on quantitative analysis of a 30,000-repository sample [15]. CVE-2026-33634 was assigned a CVSS score of 9.4, reflecting the combination of low attack complexity and high impact across confidentiality, integrity, and availability dimensions [6].

Wave Two: Credential Chaining into Checkmarx KICS (March 23)

The Trivy compromise was not merely an endpoint — it was a staging ground. Among the credentials harvested from Trivy's CI/CD environment were GitHub Personal Access Tokens belonging to Checkmarx, a separate security vendor. On March 23, TeamPCP used those stolen tokens to force-push malicious commits to all 35 version tags of `checkmarx/kics-github-action` and to poison version 2.3.28 of `checkmarx/ast-github-action` [9]. The attack window lasted approximately four hours (12:58 to 16:50 UTC) before Checkmarx detected and remediated the compromise.

The credential-chaining dynamic is among the most operationally significant aspects of the TeamPCP campaign. The group did not need to independently compromise Checkmarx; it harvested the keys to Checkmarx's systems from Trivy's pipeline, then used those keys to pivot laterally across vendor boundaries. This cross-organizational propagation demonstrates how shared CI/CD infrastructure can enable lateral movement across vendor boundaries — a significant escalation in the scope of impact achievable from a single vendor compromise. Additionally, the compromised payload installed a hidden GitHub repository named `docs-tpcp` within victim organizations to serve as an alternate command-and-control channel — providing persistence even after initial cleanup of the malicious action [9].

Checkmarx's VS Code extensions were also affected. Versions 2.53 and 1.7.0 of two extensions on the OpenVSX marketplace, representing more than 36,500 combined installations, were distributed with the same multi-stage payload architecture used in the Trivy attack [9].

Wave Three: LiteLLM and the AI Credential Aggregator Problem (March 24)

The third wave introduced a distinct and severe category of risk. LiteLLM is a Python library that provides a unified interface to over 100 LLM provider APIs, including OpenAI, Anthropic, Google, Cohere, Mistral, and Amazon Bedrock. Organizations integrate LiteLLM to simplify multi-provider AI

workflows; a single LiteLLM deployment may hold API keys for a dozen or more providers simultaneously. This architectural role — credential aggregator for AI infrastructure — made it a uniquely valuable target.

Using publishing credentials obtained during earlier campaign phases, TeamPCP published LiteLLM versions 1.82.7 and 1.82.8 to PyPI [4]. Version 1.82.6 is confirmed clean. The v1.82.7 payload used double Base64 encoding to obfuscate injected code within `litellm/proxy/proxy_server.py` — an obfuscation technique likely to evade signature-based static analysis. The v1.82.8 payload introduced a significantly more dangerous technique: a `.pth` file named `litellm_init.pth` placed in Python's `site-packages` directory.

Python processes files with the `.pth` extension unconditionally at interpreter startup. This means the malware persisted and executed with every Python process launch — including those entirely unrelated to LiteLLM — and critically, it survived uninstallation of the LiteLLM package itself. Organizations that removed the poisoned LiteLLM versions remained fully compromised unless they also manually located and deleted `litellm_init.pth` from their `site-packages` directories [4][5]. Additional persistence was established through `~/.config/sysmon/sysmon.py` and a user systemd unit named `sysmon.service`, designed to masquerade as legitimate system monitoring infrastructure.

SANS Institute estimated that 5,000 to 10,000 environments had direct exposure to the LiteLLM compromise, and approximately 1,705 Python packages were configured to automatically pull affected versions [8][15]. Named downstream users with potential exposure included Netflix, Stripe, Google, CrewAI, DSPy, and MLflow [5].

Wave Four: Telnyx and Steganographic Delivery (March 27)

The final documented wave targeted the Telnyx Python SDK through a delivery mechanism not previously observed in PyPI supply chain campaigns — steganographic embedding within WAV audio files. Versions 4.87.1 and 4.87.2, published on March 27, downloaded WAV audio files — `hangup.wav` on Windows and `ringtone.wav` on Linux — from attacker-controlled infrastructure. The second-stage payload was embedded within the audio frames using an XOR-based steganographic scheme: eight bytes of encrypted payload per audio frame, each immediately followed by its eight-byte XOR key [2]. The RSA public key used matched that from earlier campaign waves, strongly indicating shared infrastructure and consistent with a single operator set.

CanisterWorm and the Blockchain C2 Innovation

Across all four waves, TeamPCP deployed CanisterWorm — a self-replicating worm component with capabilities for Docker API enumeration, Kubernetes DaemonSet deployment, and destructive wiping. The worm's command-and-control channel was hosted on the Internet Computer Protocol (ICP) blockchain at a canister endpoint resistant to conventional domain seizure or DNS-level blocking [2]. Researchers documented this as the first confirmed operational use of decentralized blockchain infrastructure as attacker C2.

On March 23, a wiper payload named `kamikaze.sh` was deployed against Kubernetes clusters; non-containerized hosts received a persistent backdoor instead. A geographically selective element was also observed: systems geolocated to Iran received the wiper, while others received the persistence implant — suggesting either operational targeting criteria or an attempt to complicate attribution [2].

CanisterWorm also included a self-propagating npm component. Stolen npm publishing tokens were automatically weaponized to infect victim-maintained packages without additional attacker intervention, creating cascading upstream compromises across 44 additional packages in the `@emilgroup` and `@opengov` namespaces [4].

Post-Compromise Activity and Attribution

Wiz research documented the group's post-compromise tradecraft in detail [10]. After harvesting initial credentials, the group used TruffleHog to validate stolen credentials with live API calls, then performed IAM enumeration, EC2, Lambda, RDS, and ECS discovery, bulk S3 and AWS Secrets Manager exfiltration, and systematic deletion of GitHub workflow logs to obscure activity. The `gato-x` GitHub Actions exploitation research tool — originally developed for defensive research — was adapted into the attack workflow.

The group self-reported exfiltrating over 300 GB of data and approximately 500,000 credentials via Telegram channels, figures not independently verified [7]. Credential types included AWS, GCP, and Azure access tokens; SSH private keys; Kubernetes secrets and kubeconfig files; LLM provider API keys; Docker registry credentials; database passwords; TLS private keys; cryptocurrency wallets; and CI/CD pipeline secrets.

Recommendations

Immediate Actions

Any organization that ran `trivy-action`, `kics-github-action`, `checkmarx/ast-github-action`, or LiteLLM versions 1.82.7 or 1.82.8 during the exposure window should treat all credentials accessible from those environments as compromised. Rotation priority should include cloud provider access keys, LLM provider API keys, GitHub PATs and machine tokens, PyPI and npm publishing tokens, SSH keys, Kubernetes service account tokens, Docker registry credentials, and database passwords. The scope of rotation must reflect that these tools had broad environmental access, not just the permissions explicitly assigned to them.

For LiteLLM specifically, package uninstallation is insufficient for remediation. Security teams must manually search all Python `site-packages` directories for `litellm_init.pth` and remove it. They should also check for `~/.config/sysmon/sysmon.py` and active `sysmon.service` systemd units, and search Kubernetes clusters for pods matching the pattern `node-setup-*`. Network defenders should look for historical connections to `scan.aquasecurtiy[.]org`, `checkmarx[.]zone`, `models.litellm[.]cloud`, and the ICP canister endpoint `tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0[.]io`, as well as the distinctive HTTP header `X-Filename: tpcp.tar.gz` in proxy and WAF logs [1][2][4].

GitHub organizations should be audited for unexpected repositories named `tpcp-docs` (planted during the Trivy wave) or `docs-tpcp` (planted during the KICS wave), both of which served as alternate C2 staging repositories.

Short-Term Mitigations

The structural vulnerability exploited by TeamPCP in its first two waves — mutable version tags in GitHub Actions — has a direct and well-established mitigation: pin all Actions references to full commit SHAs rather than version tags. A mutable tag can be silently repointed by anyone with write access to the repository; a full commit SHA is cryptographically immutable. Organizations should audit their CI/CD workflows and replace tag-based references (e.g., `uses: aquasecurity/trivy-action@v0.20`) with SHA-pinned references (e.g., `uses: aquasecurity/trivy-action@57a97c7`).

For Python dependencies, hash verification provides equivalent protection against PyPI poisoning. Deploying pip with `--require-hashes` and a compiled `requirements.txt` that specifies expected SHA-256 digests for each package means a compromised or replaced package version will fail to install rather than execute silently. Adoption of PyPI Trusted Publishers – an OIDC-based publishing mechanism that eliminates long-lived publishing tokens – would have significantly raised the barrier for the LiteLLM wave by removing the stolen credential that enabled it.

File integrity monitoring on Python `site-packages` directories, with alerting on unexpected new `.pth` files, would have detected the LiteLLM persistence mechanism. The beacon interval for CanisterWorm was approximately 50 minutes with a 5-minute execution delay; network behavioral analysis with windows shorter than typical hourly thresholds can surface low-frequency beacons of this type.

Strategic Considerations

The TeamPCP campaign exposes a structural design assumption that warrants revisiting at an architectural level: CI/CD pipelines have historically been granted access scopes commensurate with their full operational function, which often means simultaneous access to publishing credentials, cloud infrastructure, source repositories, and LLM provider keys. This aggregation of privileges in a single pipeline execution context means that any compromised tool within the pipeline inherits all of those privileges.

The principle of least privilege applies to pipeline contexts as it does to human users and service accounts. Security scanning steps should execute with read-only access to the artifacts they analyze and no ambient access to cloud credentials or publishing tokens. LLM API keys should be scoped to the specific provider and function required, not exposed as environment variables to the entire application container. Credential separation between security tooling and production publishing workflows represents a structural defense that limits the blast radius of any single supply chain compromise, regardless of the specific technique used.

The ICP blockchain C2 technique also merits strategic attention. Conventional incident response playbooks for supply chain attacks typically include domain seizure and infrastructure takedown as key deconfliction steps. A C2 mechanism hosted on a decentralized, seizure-resistant blockchain invalidates that response option. Organizations and incident response teams should treat decentralized infrastructure as a durable C2 mechanism, requiring that eradication focus on endpoint-side implant removal rather than infrastructure disruption.

The campaign's announced pivot to ransomware deployment through the Vect partnership suggests that future exploitation of harvested credentials should be anticipated even for organizations that completed credential rotation after the initial disclosure. Harvested credential sets may have been sold or retained before rotation occurred.

CSA Resource Alignment

The TeamPCP campaign maps directly to threat scenarios described in CSA's MAESTRO framework for agentic AI systems. MAESTRO Tier 5 (Infrastructure and Scalability) addresses supply chain risks to the underlying infrastructure on which agentic systems depend, including compromised dependencies, poisoned package repositories, and malicious tooling distributed through official channels [11]. The LiteLLM compromise — where an AI gateway library serving as a central dependency for agentic systems was backdoored — closely resembles the Tier 5 attack pattern described in MAESTRO and represents precisely the kind of infrastructure-layer threat the framework was designed to model [16][17].

CSA's AI Controls Matrix (AICM v1.0.3) provides a detailed control framework applicable to the vulnerabilities TeamPCP exploited. The Supply Chain Management and Transparency (STA) domain, controls STA-01 through STA-09, addresses open-source dependency governance, third-party AI component vetting, and Software Bill of Materials requirements [12]. The LiteLLM wave specifically illustrates the risk described in STA-04 and STA-06: organizations that lacked a comprehensive inventory of which systems consumed LiteLLM, and at which version, were unable to assess their exposure scope or prioritize credential rotation. An enforced AI-BOM posture would have made this triage tractable rather than speculative.

CSA's Zero Trust guidance for LLM environments addresses the credential aggregation risk directly, recommending that LLM provider API keys be treated as high-value secrets subject to the same access controls applied to cloud root credentials, with per-request authentication, comprehensive audit logging, and isolation between environments [13]. The TeamPCP campaign validates this framing: LLM API keys are functionally equivalent to cloud access credentials in terms of the privileges and costs they govern.

The mutable tag vulnerability in GitHub Actions aligns with the Supply Chain Management and Transparency (STA) domain of the CSA Cloud Controls Matrix (CCM), which addresses integrity verification requirements for software supply chains. CCM control STA-09 requires cryptographic integrity verification for third-party components prior to integration — a control that SHA pinning in GitHub Actions directly satisfies [14].

References

[1] Unit 42, Palo Alto Networks. "TeamPCP Supply Chain Attacks: Weaponizing the Protectors." Palo Alto Networks, March 2026.

[2] Wiz Security Research. "Threes a Crowd: TeamPCP Trojanizes LiteLLM in Continuation of Campaign." Wiz, March 2026.

[3] Wiz Security Research. "Trivy Compromised by TeamPCP: Supply Chain Attack Analysis." Wiz, March 2026.

[4] Datadog Security Labs. "LiteLLM and Telnyx PyPI Compromise: TeamPCP Supply Chain Campaign." Datadog, March 2026.

[5] Endor Labs. "TeamPCP Isn't Done: Tracking Ongoing Supply Chain Risk." Endor Labs, March 2026.

[6] Help Net Security. "CISA Adds CVE-2026-33634 and CVE-2026-33017 to Known Exploited Vulnerabilities Catalog." Help Net Security (reporting on CISA advisory), March 27, 2026.

[7] Help Net Security. "TeamPCP Threat Group Pivots to Ransomware Following Supply Chain Campaign." Help Net Security, March 30, 2026.

[8] SANS Internet Storm Center. "When the Security Scanner Became the Weapon: Inside the TeamPCP Supply Chain Campaign." SANS Institute, March 2026.

[9] Wiz Security Research. "KICS GitHub Action Compromised: TeamPCP Expands Campaign." Wiz, March 2026.

[10] Wiz Security Research. "Tracking TeamPCP: Investigating Post-Compromise Attacks Seen in the Wild." Wiz, March 2026.

[11] Cloud Security Alliance. "MAESTRO: Multi-Agent Environment, Security, Threats, and Risk Operations." CSA AI Safety Initiative, 2025. *(Primary MAESTRO artifact URL was not reachable at time of publication; see CSA AI working group page for current links.)*

[12] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.0.3." CSA, 2025.

[13] Cloud Security Alliance. "Using Zero Trust to Secure Enterprise Information in LLM Environments." CSA, 2025. *(Direct paper link may vary; see CSA Zero Trust Working Group page.)*

[14] Cloud Security Alliance. "Cloud Controls Matrix v4." CSA, 2023.

[15] GitGuardian. "TeamPCP Snowball Analysis: Quantifying Supply Chain Exposure." GitGuardian, March 2026.

[16] Cloud Security Alliance. "TeamPCP and the Cascading AI/ML Supply Chain Campaign." CSA Lab Space, March 29, 2026.

[17] Cloud Security Alliance. "TeamPCP: Cascading Supply Chain Attack on AI/ML Tooling." CSA Lab Space, March 30, 2026.