TeamPCP: CI/CD Security Tool Supply Chain Compromise

Four-Wave Campaign Weaponizes Trivy, KICS, and LiteLLM Against Cloud Infrastructure

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-25

Key Takeaways

- The threat actor known as TeamPCP orchestrated a four-wave supply chain campaign between March 19–24, 2026, compromising Trivy (Aqua Security), Checkmarx KICS/AST GitHub Actions, and LiteLLM—three widely used tools in cloud-native CI/CD pipelines.
- The campaign is distinctive for using compromised security scanning tools as the initial vector, enabling cascading credential theft that fed subsequent attack waves. Tokens stolen in the Trivy compromise were used to compromise LiteLLM's PyPI publishing pipeline five days later.
- CVE-2026-33634 [24] (CVSS 9.8 Critical) covers the Trivy binary and GitHub Actions compromise; PYSEC-2026-2 [23] covers the malicious LiteLLM PyPI packages (v1.82.7, v1.82.8). The Checkmarx KICS/AST GitHub Actions compromise has no assigned CVE as of this writing.
- All three compromised tools deployed a "TeamPCP Cloud Stealer" payload designed to harvest cloud provider credentials (AWS, GCP, Azure), Kubernetes tokens, SSH keys, and secrets from `.env` files, exfiltrating via typosquatted domains including `scan.aquasecurtiy[.]org` and `checkmarx[.]zone`.
- A self-propagating npm worm (CanisterWorm) using Internet Computer Protocol (ICP) blockchain infrastructure for command-and-control extended the campaign's reach to 141 malicious npm package artifacts across 66+ packages.
- A geopolitically targeted destructive wiper was deployed against Kubernetes clusters detected as operating in Iran, executing recursive filesystem deletion and cluster-wide DaemonSet deployment.
- Organizations that ran Trivy, KICS, or LiteLLM during the exposure windows should treat their entire credential store as compromised and rotate all secrets immediately.

Background

The Threat Actor: TeamPCP

TeamPCP—also tracked under the aliases DeadCatx3, PCPcat, and ShellForce—is a financially and geopolitically motivated threat actor with an established history targeting cloud-native infrastructure. The group conducted a mass compromise of over 60,000 servers in December 2025, exploiting exposed Docker APIs, Kubernetes API servers, Redis instances, and a React2Shell vulnerability (CVE not publicly assigned as of this writing) [1]. Their March 2026 campaign marks a tactical shift: rather than directly targeting vulnerable infrastructure, they subverted the security tooling used to protect it — an approach consistent with broader supply chain attack trends but representing a new capability for this actor.

Based on the operational consistency documented across multiple incident analyses, CSA assesses attribution confidence as high, though no formal intelligence community attribution has been published as of this writing [1][2][5][21]. TeamPCP exhibits consistent operational signatures across attack waves, including the use of RSA-4096/AES-256-CBC hybrid encryption for credential exfiltration, archive naming as `tpcp.tar.gz`, typosquatted domain patterns mimicking legitimate vendor infrastructure, and a YouTube kill-switch mechanism in their backdoor payloads [1][2]. Their motivation appears hybrid—financial gain through credential theft and access brokering, reputation-focused defacement of victim organizations' repositories, and a geopolitical destructive component targeting Iranian infrastructure. Whether the Iran-targeting reflects state direction or independent actor preference has not been established from public evidence.

The Attack Chain: A Cascading Supply Chain Compromise

What distinguishes the March 2026 TeamPCP campaign is the deliberate chaining of compromises across the software supply chain. Each wave used credentials stolen in the previous one, creating a cascade that transformed a single incomplete credential rotation into a multi-vendor incident affecting AI infrastructure, cloud security tooling, and the npm ecosystem simultaneously.

The chain began in late February 2026, when a misconfigured GitHub Actions workflow in the Trivy repository allowed the `hackerbot-claw` bot to exploit a `pull_request_target` vulnerability, stealing a personal access token with write privileges. When Aqua Security disclosed this incident publicly on approximately March 1 and attempted credential rotation, the rotation was incomplete—a residual valid service account token remained active. This residual access became the foothold for the March 19 attack [3][4].

Security Analysis

Wave 1: The Trivy Ecosystem Compromise (March 19, 2026)

At approximately 17:43 UTC on March 19, 2026, attackers used the residual service account token to force-push 76 of 77 version tags in the `aquasecurity/trivy-action` repository and all 7 tags in `aquasecurity/setup-trivy` to malicious commits. A malicious binary—Trivy v0.69.4—was published to the GitHub releases page at 18:22 UTC [3].

The malicious payload executed silently before the legitimate scanning logic, so affected CI/CD pipelines appeared to complete successfully with no visible anomaly [3][4]. The stealer performed memory scraping of `/proc/*/mem` to dump the `Runner.Worker` process, harvesting GitHub PATs, SSH keys, AWS/GCP/Azure IAM credentials, Kubernetes service account tokens, Docker registry credentials, database passwords, and TLS private keys from the runner's in-memory state. It supplemented this with a direct query to the AWS Instance Metadata Service (IMDS) endpoint at `169.254.169.254` for IAM credentials. All harvested material was encrypted using AES-256-CBC with RSA-4096 OAEP key encapsulation, archived as `tpcp.tar.gz`, and exfiltrated to attacker infrastructure at `scan.aquasecurtiy[.]org` (45.148.10.212)—a domain that typosquats Aqua Security's legitimate domain. As a fallback, the malware created public GitHub repositories prefixed `tpcp-docs-<timestamp>` in victim accounts for out-of-band data staging [3][4][21].

Aqua Security detected the compromise at approximately 20:38 UTC and began containment. The malicious v0.69.4 binary was removed by 21:42 UTC, setup-trivy tags were remediated by 21:44 UTC, and trivy-action tags were fully remediated by 05:40 UTC on March 20. Malicious Docker Hub images (v0.69.5, v0.69.6) published on March 22 represented a second attempt by the attacker to maintain a foothold via an alternate artifact registry. Safe versions are: trivy binary v0.69.2 or v0.69.3, trivy-action v0.35.0, and setup-trivy v0.2.6 (post-recreation) [3]. According to Mandiant, as reported by The Register, more than 1,000 SaaS environments were confirmed compromised, with over 10,000 CI/CD pipelines potentially exposed based on GitHub workflow reference counts [5].

Wave 2: The CanisterWorm npm Campaign (March 20–22, 2026)

Beginning the evening of March 20, TeamPCP deployed a self-propagating npm worm using publish tokens stolen during the Trivy wave. The worm's distinguishing feature is autonomous propagation: compromised packages include a `findNpmTokens()` function that locates npm authentication

tokens in the installing developer's environment and autonomously publishes the malware to every package the victim has publish rights to—transforming each victim into a propagation vector without further attacker intervention [6].

The worm's command-and-control infrastructure uses an ICP (Internet Computer Protocol) canister at `tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io`, a smart contract on the Internet Computer blockchain. Unlike traditional C2 domains that can be sinkholed or seized, blockchain-hosted canister endpoints are substantially more resistant to conventional takedown — ICP's on-chain governance layer and DFINITY Foundation's abuse processes represent available but slower remediation pathways compared to DNS-based infrastructure [6][7]. The backdoor polls this endpoint every approximately 50 minutes using a spoofed browser User-Agent string. A kill-switch is embedded in the payload: if the returned URL contains `youtube.com`, execution is skipped, allowing the attacker to keep infected systems dormant and avoid detection [6][7].

The first CanisterWorm wave compromised 47 packages across multiple npm organization scopes including @EmilGroup, @opengov, @airtm, and @pypestream. By the time of widespread detection, 141 malicious package artifacts across 66+ unique npm packages had been identified [6]. The March 22 defacement of 44 Aqua Security internal GitHub repositories with the message "TeamPCP Owns Aqua Security" appears to have been a demonstrative act of access — a message occurring on the same day as the CanisterWorm deployment, though whether the two were coordinated within the campaign is not established from public evidence.

Wave 3: The Checkmarx KICS and AST GitHub Actions Compromise (March 23, 2026)

At 02:53 UTC on March 23, attackers compromised the `cx-plugins-releases` Checkmarx service account. While the precise vector is publicly unconfirmed by Checkmarx, the credential harvesting patterns from Wave 1 provide a plausible causal pathway—KICS and AST GitHub Actions workflows that ran Trivy scans during the Wave 1 exposure window would have had their runner memory scraped [8][18]. Between 12:58 and 16:50 UTC, 35 version tags in `checkmarx/kics-github-action` and corresponding tags in `checkmarx/ast-github-action` were force-pushed to malicious commits containing an identical shell-based stealer payload [8].

The Checkmarx payload—identified by vendors as the "TeamPCP Cloud Stealer"—added Slack and Discord webhook URL harvesting to the credential set already described in Wave 1. Exfiltration occurred to `checkmarx[.]zone` (83.142.209.11), another typosquatted domain. OpenVSX plugins `checkmarx.ast-results` v2.53.0 and `checkmarx.cx-dev-assist` v1.7.0 were also

compromised between 02:53 and 15:41 UTC, extending the attack surface to developer IDE environments [8]. Safe versions are: kics-github-action v2.1.20 or later, ast-github-action v2.3.33 or later, and the VS Code/OpenVSX extensions v2.56.0 and v1.10.0 respectively.

Wave 4: The LiteLLM PyPI Compromise (March 24, 2026)

The final documented wave targeted LiteLLM, a widely used Python library that provides a unified interface to commercial large language model APIs. With approximately 95 million monthly PyPI downloads [9] and present in an estimated 36% of cloud environments scanned by Wiz at the time of the incident [10], LiteLLM represents a high-value target for an attacker seeking broad AI infrastructure access.

The most probable root cause traces to Wave 1: LiteLLM's `PYPI_PUBLISH` token was stored as a `.env` variable in the project's GitHub repository; if LiteLLM's CI/CD pipeline ran Trivy during the Wave 1 exposure window, this token would have been harvested via runner memory scraping. With this token, the attacker bypassed LiteLLM's official CI/CD release workflows entirely, uploading malicious packages directly to PyPI [9][10][20][22].

Malicious versions 1.82.7 and 1.82.8 were published at 10:39 and 10:52 UTC respectively — a 13-minute interval suggesting either active payload iteration or pre-staged sequential deployment. PyPI quarantined both packages at approximately 11:25 UTC, though the effective exposure window extended to approximately 16:00 UTC per LiteLLM's own advisory [11] — a gap of nearly 4.5 hours attributable to cached PyPI mirrors, CDN propagation delays, and packages already installed from cache that continued executing without re-fetching from the registry. The last clean version is 1.82.6.

The v1.82.8 payload escalated significantly beyond v1.82.7. While v1.82.7 injected Base64-encoded malicious code at line 128 of `litellm/proxy/proxy_server.py` —requiring LiteLLM proxy invocation to trigger—v1.82.8 added a `litellm_init.pth` file to the Python `site-packages/` directory. A `.pth` file with executable code is processed on every Python interpreter startup, meaning the backdoor fires without any LiteLLM invocation. Any Python process on the infected host—cron jobs, other applications, administrative scripts—triggers credential exfiltration [9][10]. The three-stage payload in v1.82.8 performs recursive credential harvesting across 8 directories to a depth of 6 levels (SSH keys, `.env` files, AWS/GCP/Azure credentials, Kubernetes secrets, database passwords, authentication logs, and `/etc/passwd` / `/etc/shadow`), followed by Kubernetes lateral movement that deploys privileged pods named `node-setup-{node_name}` to the `kube-system` namespace with `hostPID`, `hostNetwork`, and a full host filesystem mount, and establishes a persistent systemd backdoor polling `checkmarx.zone/raw` on a 50-minute interval [10][11].

A destructive wiper component detected the target system's timezone (`Asia/Tehran`) and locale (`fa_IR`) and, if both conditions were met, executed `rm -rf / --no-preserve-root` on standalone hosts and deployed a Kubernetes DaemonSet with universal tolerations for cluster-wide recursive filesystem deletion followed by a forced reboot [12]. This represents a significant escalation from credential theft to infrastructure destruction, though its geopolitical intent and authorization remain unconfirmed from public evidence.

Recommendations

Immediate Actions

Organizations should assume any CI/CD pipeline that ran `aquasecurity/trivy-action` (versions v0.0.1–v0.34.2), `aquasecurity/setup-trivy` (v0.2.0–v0.2.6 pre-recreation), `checkmarx/kics-github-action` (tags before v2.1.20), or `checkmarx/ast-github-action` (tags before v2.3.33) between March 19–23, 2026 had its runner memory scraped. Any Python environment that installed `litellm` v1.82.7 or v1.82.8 should be treated as fully compromised regardless of whether LiteLLM was invoked. The specific remediation steps are:

- Rotate all secrets accessible to affected CI/CD runners immediately, including cloud provider IAM credentials, GitHub PATs, SSH keys, Kubernetes service account tokens, Docker registry credentials, database passwords, and any secrets stored in environment variables.
- Scan all systems that ran affected LiteLLM versions for `litellm_init.pth` in any Python `site-packages/` directory and remove it. Inspect for `~/.config/sysmon/sysmon.py` and corresponding systemd units named "System Telemetry Service," "pgmon.service," or "sysmon.service."
- Audit Kubernetes clusters accessible from affected environments for unauthorized pods named `node-setup-*` in the `kube-system` namespace, and inspect for DaemonSets with universal tolerations and `hostPID` / `hostNetwork` flags.
- Search GitHub repositories for publicly visible repos prefixed `tpcp-docs-` which may indicate exfiltration staging by the malware in victim accounts.
- Block outbound connections to known TeamPCP C2 infrastructure: `scan.aquasecurtiy[.]org` (45.148.10.212), `checkmarx[.]zone` (83.142.209.11), `models[.]litellm[.]cloud`, and the ICP canister `tdtqy-oyaaa-aaaae-af2dq-cai.raw.icp0.io`.

Short-Term Mitigations

Pin GitHub Actions to full commit SHAs rather than mutable version tags. The core vulnerability enabling Waves 1 and 3 was the use of Git tags (e.g., `uses: aquasecurity/trivy-action@v0.34.0`) which can be force-pushed to point at arbitrary commits. Pinning to a full commit SHA (e.g., `uses: aquasecurity/trivy-action@a20de5420d57c4102486cdd9349b532d685f4b1e`) eliminates this vector entirely, as commit hashes cannot be retroactively altered [3][4]. While this approach requires periodic manual updates as upstream tools release new versions, the security benefit substantially outweighs the maintenance cost.

CI/CD runners should operate under the principle of least-privilege credential access. GitHub Actions runners should not receive PATs or service account tokens with broader permissions than the specific task requires. Secrets should not be stored in `.env` files within repositories—use dedicated secrets managers (AWS Secrets Manager, HashiCorp Vault, GitHub Encrypted Secrets) with short-lived, scoped credentials wherever possible [4][13][19].

Enable OIDC-based short-lived credential issuance for cloud provider access from CI/CD pipelines. AWS, GCP, and Azure all support GitHub Actions OIDC federation, which eliminates the need to store long-lived credentials in CI/CD environments at all. Because the TeamPCP stealer relies on harvesting long-lived credentials from memory, OIDC-federated pipelines substantially reduce blast radius even if a scanner compromise occurs.

Strategic Considerations

The TeamPCP campaign exposes a structural weakness in the security tooling supply chain: the tools organizations rely on to detect vulnerabilities in their software now constitute a high-value attack surface in their own right. Security scanners occupy a privileged position in CI/CD pipelines—they run with broad credentials, access source code and artifact repositories, and are trusted implicitly by the teams that deploy them. An attacker who compromises a widely used scanner can harvest credentials from thousands of organizations simultaneously without triggering the security controls that scanner is ostensibly enforcing.

Organizations should extend their software composition analysis practices to cover their own security tooling, applying the same rigor to Trivy, KICS, and similar tools that they apply to application dependencies. This includes monitoring for unexpected behavior from scanner processes (e.g., outbound network connections, filesystem writes outside scan paths, memory access to unrelated

processes), which behavioral detection tools on CI/CD runners can surface. Subscribing to security advisories from all CI/CD tool vendors and participating in coordinated vulnerability disclosure programs for tools embedded in build pipelines provides earlier warning when the tooling itself is compromised.

The CanisterWorm's use of ICP blockchain infrastructure for C2 represents a demonstrated instance of blockchain-based C2 resilience. Organizations relying on domain blocklists or sinkholing for C2 disruption should evaluate whether their endpoint detection tooling can identify and block connections to `.icp0.io` endpoints or unusual polling patterns consistent with blockchain-hosted C2.

CSA Resource Alignment

This incident connects directly to multiple areas of CSA's established guidance on cloud and AI security. The MAESTRO threat modeling framework's Layer 3 (Agent Execution Environment) and Layer 5 (Infrastructure and Orchestration) are directly relevant: the campaign demonstrates how compromised execution environments can be weaponized to harvest credentials that enable lateral movement through orchestration layers, precisely the threat model MAESTRO addresses for agentic AI systems operating in Kubernetes-hosted environments [14].

CSA's Cloud Controls Matrix (CCM) Domain SEF (Security Incident Management, E-Discovery & Cloud Forensics) and AIS (Application & Interface Security) controls address the organizational response requirements activated by this incident. Specifically, CCM control AIS-04 (Application Security Metrics) and AIS-07 (Third-Party Software) call for security assessments of third-party software components, which should explicitly include CI/CD tooling. CCM control SEF-02 (Incident Management) requires timely notification and documentation procedures; organizations with affected pipelines should review and execute the response procedures described in SEF-02 [15].

CSA's guidance on software supply chain transparency—including SBOM adoption per CycloneDX and SPDX standards—provides a framework for the continuous inventory of CI/CD tooling that would enable organizations to rapidly identify affected components during future campaigns. Applying SBOM practices to internal CI/CD pipelines, not just application software, is a direct extension of this guidance to the threat model this campaign represents [16].

The AI Controls Matrix (AICM), CSA's AI-specific extension of the CCM, addresses the particular risk this campaign poses to AI workloads. LiteLLM's widespread presence in AI inference infrastructure means the supply chain compromise reached directly into environments where large language model APIs are

called and where AI agent orchestration logic runs. The AICM's controls on model and dependency integrity validation, and on runtime monitoring of AI system components, apply directly to the threat of compromised AI support libraries such as LiteLLM [17].

The CSA Zero Trust guidance's principle of continuous verification is operationalized by short-lived OIDC credentials and commit-SHA pinning: neither trust the identity of a secret at rest (which can be stolen) nor the identity of a version tag (which can be redirected), but instead verify the specific artifact at each step of the supply chain.

References

[1] cstromblad.com, "Threat Actor Profile — TeamPCP," March 2026. https://cstromblad.com/posts/threat-actor-profile-teampcp/

[2] CrowdStrike, "From Scanner to Stealer — Inside the trivy-action Supply Chain Compromise," March 2026. https://www.crowdstrike.com/en-us/blog/from-scanner-to-stealer-inside-the-trivy-action-supply-chain-compromise/

[3] GitHub Security Advisory, "GHSA-69fq-xp46-6x23: Trivy ecosystem supply chain temporarily compromised," March 23, 2026. https://github.com/aquasecurity/trivy/security/advisories/GHSA-69fq-xp46-6x23

[4] Aqua Security, "Trivy Supply Chain Attack — What You Need to Know," March 2026. https://www.aquasec.com/blog/trivy-supply-chain-attack-what-you-need-to-know/

[5] The Register, "1K+ cloud environments infected via Trivy attack," March 24, 2026. https://www.theregister.com/2026/03/24/1k_cloud_environments_infected_following/

[6] Aikido Security, "TeamPCP Deploys CanisterWorm on npm Following Trivy Compromise," March 2026. https://www.aikido.dev/blog/teampcp-deploys-worm-npm-trivy-compromise

[7] JFrog Security Research, "New compromised versions detected in CanisterWorm attack," March 2026. https://research.jfrog.com/post/canister-worm/

[8] Checkmarx, "Checkmarx Security Update," March 2026. https://checkmarx.com/blog/checkmarx-security-update/

[9] Endor Labs, "TeamPCP Isn't Done," March 2026. https://www.endorlabs.com/learn/teampcp-isnt-done

[10] Wiz, "Threes a Crowd — TeamPCP Trojanizes LiteLLM in Continuation of Campaign," March 2026. https://www.wiz.io/blog/threes-a-crowd-teampcp-trojanizes-litellm-in-continuation-of-campaign

[11] LiteLLM, "Security Update March 2026," March 24, 2026. https://docs.litellm.ai/blog/security-update-march-2026

[12] BleepingComputer, "TeamPCP Deploys Iran-Targeted Wiper in Kubernetes Attacks," March 2026. https://www.bleepingcomputer.com/news/security/teampcp-deploys-iran-targeted-wiper-in-kubernetes-attacks/

[13] GitGuardian, "Trivy's March Supply Chain Attack Shows Where Secret Exposure Hurts Most," March 2026. https://blog.gitguardian.com/trivys-march-supply-chain-attack-shows-where-secret-exposure-hurts-most/

[14] Cloud Security Alliance, "MAESTRO: Multi-Agent Environment Security Threat Representation and Overview," 2025. https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro

[15] Cloud Security Alliance, "Cloud Controls Matrix v4," 2021. https://cloudsecurityalliance.org/research/cloud-controls-matrix/

[16] Cloud Security Alliance, "Securing the Software Supply Chain: Transparency in the Age of the Software Driven Society," 2024. https://cloudsecurityalliance.org/research/working-groups/security-software-supply-chain/

[17] Cloud Security Alliance, "AI Controls Matrix (AICM)," 2025. https://cloudsecurityalliance.org/artifacts/ai-controls-matrix/

[18] Sysdig, "TeamPCP Expands — Supply Chain Compromise Spreads from Trivy to Checkmarx GitHub Actions," March 2026. https://www.sysdig.com/blog/teampcp-expands-supply-chain-compromise-spreads-from-trivy-to-checkmarx-github-actions

[19] Microsoft Security Blog, "Guidance for Detecting, Investigating, and Defending Against the Trivy Supply Chain Compromise," March 24, 2026. https://www.microsoft.com/en-us/security/blog/2026/03/24/detecting-investigating-defending-against-trivy-supply-chain-compromise/

[20] ReversingLabs, "TeamPCP Software Supply Chain Attack Spreads to LiteLLM," March 2026. https://www.reversinglabs.com/blog/teampcp-supply-chain-attack-spreads

[21] Palo Alto Networks Unit 42, "When Security Scanners Become the Weapon — Breaking Down the Trivy Supply Chain Attack," March 2026. https://www.paloaltonetworks.com/blog/cloud-security/trivy-supply-chain-attack/

[22] Snyk, "How a Poisoned Security Scanner Became the Key to Backdooring LiteLLM," March 2026. https://snyk.io/articles/poisoned-security-scanner-backdooring-litellm/

[23] PyPA Advisory Database, "PYSEC-2026-2: Malicious code in litellm," March 24, 2026. https://github.com/pypa/advisory-database/blob/main/vulns/litellm/PYSEC-2026-2.yaml

[24] Tenable, "CVE-2026-33634," March 23, 2026. https://www.tenable.com/cve/CVE-2026-33634