CSAI Foundation

Cloud Security Alliance AI Safety Initiative

Weaponized Scanners: TeamPCP's Cloud-Native Kill Chain

How a Trivy CI/CD Compromise Cascaded to AI Framework Backdoors, Source Code Theft, and Geopolitical Wiper Malware

Unofficial AI-assisted Research

2026-04-01

Executive Summary

On March 19, 2026, a threat group known as TeamPCP hijacked the GitHub Actions pipelines for Aqua Security's Trivy vulnerability scanner — one of the most widely deployed open-source container security scanners, referenced in more than 10,000 GitHub Actions workflow files [2]. By force-pushing malicious code to 76 of 77 release tags and publishing a backdoored binary to official distribution channels, TeamPCP turned a trusted security instrument into a credential harvesting implant, quietly collecting AWS, GCP, and Azure credentials, SSH keys, Kubernetes service account tokens, and Git authentication material from CI/CD runners at potentially thousands of organizations — though the full scope of successful exfiltration remains difficult to quantify [1][2].

What followed illustrated, with uncommon clarity, how a single high-value compromise propagates through interconnected cloud-native infrastructure. Using credentials stolen via the Trivy pipeline, TeamPCP backdoored LiteLLM — a widely adopted AI model routing library with approximately 95 million monthly downloads — inserting malicious code into versions 1.82.7 and 1.82.8 [3]. A separate credential thread yielded unauthorized access to Cisco's internal development environment, resulting in the theft of customer-facing source code [4]. Meanwhile, a self-propagating worm called CanisterWorm, seeded through the stolen credential network into 47 npm packages, delivered a Kubernetes DaemonSet-based wiper payload that destroys cluster node filesystems — and includes a geopolitical trigger that fires automatically when it detects an Iranian timezone configuration [5][1].

This report examines the TeamPCP kill chain in technical detail, analyzes the mechanisms of each attack stage, and draws out the implications for cloud security practitioners, DevSecOps teams, and AI platform engineers. The campaign represents a significant escalation in the scale and impact of CI/CD supply chain attacks, building on techniques demonstrated by earlier campaigns to achieve industrial-scale credential harvest by targeting the security toolchain itself.

Introduction and Background

The Defender's Dilemma at Scale

The security industry's response to the proliferation of cloud-native infrastructure has been to build an equally distributed security toolchain: container image scanners that run in every CI/CD pipeline, infrastructure-as-code analyzers that gate every infrastructure change, secrets detectors that monitor every commit. This toolchain is deeply trusted and, by design, deeply integrated — security scanners run with elevated privileges, access to credentials, and broad network reach that application code rarely receives. The premise is that this integration is acceptable because the tools themselves are assumed safe.

TeamPCP's March 2026 campaign invalidated that premise in concrete, measurable terms. By compromising a security scanner at the distribution layer and exploiting the trust that CI/CD pipelines extend to that scanner, the group achieved credential access at industrial scale before any single affected organization could detect the intrusion. The attack did not require a vulnerability in the target enterprises' own code, systems, or defenses. It required only that those enterprises trust Trivy — which, by all conventional measures, they were entirely reasonable to do.

TeamPCP: Threat Actor Profile

TeamPCP is a cloud-focused, financially motivated criminal group that emerged in late 2024 and has been tracked under multiple aliases including DeadCatx3, PCPcat, PersyPCP, and ShellForce [7]. The group's early activity centered on opportunistic exploitation of misconfigured cloud infrastructure: exposed Docker daemon APIs, unauthenticated Kubernetes dashboards, publicly accessible Ray ML cluster endpoints, and Redis servers without authentication. From these initial footholds, TeamPCP harvested credentials and deployed cryptomining payloads — a monetization model shared by many cloud-opportunistic groups.

What distinguishes TeamPCP from that broad category of actors is the sophistication and intentionality of its infrastructure development. The group uses the Sliver open-source command-and-control framework, which supports encrypted communications over mTLS, WireGuard, HTTP/S, and DNS channels — providing operational flexibility and resilience that most financially motivated actors do not invest in [1][8]. It uses FRP (Fast Reverse Proxy) and Go Simple Tunnel for traffic relay, and it has demonstrated sustained operational security across multi-week campaigns. By early 2026, threat

intelligence from both Wiz and Flare documented explicit group communications indicating a planned pivot to ransomware, with members stating intent to "chain these compromises into devastating follow-on ransomware campaigns" [7][9][18].

TeamPCP's targeting profile reflects its cloud-native exploitation methodology. The group concentrates on sectors that rely heavily on cloud-native environments with externally accessible services: banking and financial services, consumer goods, and professional services. Its exploitation pattern exploits Azure infrastructure disproportionately, with Azure endpoints comprising approximately 61 percent of observed targets and AWS endpoints comprising approximately 36 percent [9]. This distribution tracks with enterprise cloud adoption patterns in the industries the group favors.

The March 2026 attack burst represents a step-change in operational scope. Over an eight-day period, TeamPCP conducted simultaneous or closely sequenced operations across five ecosystems: GitHub Actions (Trivy and KICS), Docker Hub, npm (via CanisterWorm), OpenVSX, and PyPI (targeting LiteLLM and Telnyx). Delivery techniques evolved rapidly across the campaign, progressing from inline Base64 encoding to `.pth` auto-execution to split-file WAV steganography for payload delivery [1][5]. The campaign also expanded from Linux-only targeting to include Windows persistence mechanisms, indicating that the group's initial Linux focus was a resource constraint rather than a strategic limitation.

The Significance of the Trivy Compromise

Trivy was not a random target of opportunity. It is one of the most widely deployed open-source security tools in the cloud-native ecosystem, integrated by default into GitHub Actions workflows, GitLab CI pipelines, Jenkins configurations, and Kubernetes admission control systems at organizations ranging from individual developers to Fortune 100 companies. Security teams typically do not scrutinize the runtime behavior of Trivy because doing so would be operationally unusual — organizations run security scanners specifically because they trust them. The `aquasecurity/trivy-action` GitHub Action alone was referenced in more than 10,000 workflow files on GitHub at the time of the compromise [2][19].

This scale of deployment, combined with the privileged access that CI/CD runners require for security scanning (access to registries, deployment credentials, infrastructure authentication), meant that a single compromise of Trivy's distribution channel could yield credential access at a scope that would be impossible to achieve through targeted intrusion alone. TeamPCP's choice of target appears strategically deliberate rather than opportunistic. The attack methodology is consistent with a strategy that prioritizes maximum credential yield from a single high-leverage compromise: instead of targeting ten thousand organizations individually, compromise the tool they all depend on.

Stage One: The Trivy GitHub Actions Compromise

Initial Access: The `pull_request_target` Misconfiguration

The Trivy compromise was not a zero-day exploit. Its root cause was a well-documented class of misconfiguration in GitHub Actions workflows: the improper use of the `pull_request_target` trigger in a context that grants access to repository secrets [1][2]. The `pull_request_target` event was designed to allow CI/CD workflows to access privileged secrets when triggered by pull requests from forks — a necessary feature for workflows that need to post test results back to a repository, for example — but it introduces substantial risk if the triggered workflow also executes code from the pull request itself. When a workflow with this configuration runs untrusted code from a fork while having access to secrets, an attacker can craft a malicious pull request that extracts those secrets.

In late February 2026, an autonomous exploitation bot identified by researchers as `hackerbot-claw` exploited exactly this misconfiguration in Trivy's GitHub Actions configuration to steal a Personal Access Token (PAT) from the pipeline environment [2]. Aqua Security discovered the breach and rotated credentials — but the rotation was incomplete. Incomplete credential rotation is a common failure mode in complex multi-token environments; the surviving credentials allowed TeamPCP to retain access through the initial remediation.

Attack Execution: Tag Poisoning at Scale

Between March 19 and March 23, 2026, TeamPCP used its retained access to execute a coordinated rewriting of Trivy's release infrastructure. The attack methodology — commonly called retroactive tag poisoning — exploited the fact that version tags in Git repositories are mutable references that can be force-pushed to point to different commits. Most CI/CD pipelines that reference `aquasecurity/trivy-action@v0.20.0`, for example, do not verify the cryptographic identity of that tag; they fetch whatever commit the tag currently points to.

TeamPCP force-pushed 76 of 77 release tags in the `aquasecurity/trivy-action` repository, redirecting them to malicious commits while preserving the tag names [1][2][16]. All seven tags in the related `aquasecurity/setup-trivy` repository were similarly hijacked. A malicious Trivy binary — identified in threat intelligence reporting as version v0.69.4 — was published to official GitHub Releases and container registries with the appropriate signatures and metadata to pass routine integrity checks. Critically, the malicious binary executed legitimate Trivy functionality in parallel with its payload, ensuring that CI/CD pipeline logs showed expected scanner output while the malicious component ran silently in the background.

The malicious binary distributed through this mechanism is described in reporting as the "TeamPCP Cloud Stealer." It dumped the memory of the Runner.Worker process, harvested SSH keys and known_hosts files, collected cloud provider credential files (AWS `~/.aws/credentials`, GCP application default credentials, Azure CLI tokens), extracted Kubernetes service account tokens from standard mount paths, captured Docker configuration files (which frequently contain registry authentication), and gathered Git credential helper configurations. All harvested material was encrypted using AES-256 with an RSA-4096 wrapped key and exfiltrated to attacker-controlled infrastructure [1] [2].

Extent of Exposure

Quantifying the full scope of credential theft from the Trivy compromise is inherently difficult: organizations that executed compromised Trivy workflows may not have detected the intrusion, and many credential theft victims do not publicly disclose. However, the scale of known Trivy deployment in production CI/CD environments — more than 10,000 GitHub workflow files, plus private GitLab and Jenkins configurations that are not publicly enumerable — establishes a theoretical exposure surface that is extraordinarily broad [2].

Microsoft's Defender for Cloud team, which published detection guidance on March 24, 2026, identified anomalous exfiltration behavior from CI/CD runners across enterprise customer environments in the days following the compromise [10]. Wiz's threat intelligence team similarly documented post-compromise activity in cloud environments consistent with credential use derived from the Trivy theft [1] [17]. Safe versions of the affected tooling were identified as: Trivy binary v0.69.3 or earlier, `trivy-action` pinned to commit `57a97c7` (v0.35.0), and `setup-trivy` pinned to commit `3fb12ec` (v0.2.6) [2]. The definitive remediation for workflows using either action is to pin to full commit SHAs rather than mutable version tags.

On March 23, 2026, TeamPCP used CI/CD secrets stolen from the Trivy pipeline environment to compromise GitHub Actions associated with Checkmarx's KICS infrastructure-as-code scanner, extending the credential harvest to a second widely deployed security tooling ecosystem [11][6].

Stage Two: LiteLLM Backdoor and the AI Infrastructure Attack Surface

Why LiteLLM Matters

LiteLLM is a Python library that provides a unified interface for routing requests across multiple large language model providers, including OpenAI, Anthropic, Google, and dozens of others. It has become a foundational infrastructure component in enterprise AI deployments, used by teams building AI agents, chatbots, internal tooling, and AI-powered automation pipelines. Its approximately 95 million monthly downloads reflect not marginal adoption but deep integration into the production AI infrastructure of a substantial fraction of the enterprise market [3].

The significance of LiteLLM's position in the attack chain is not merely quantitative. LiteLLM handles API authentication material — provider keys, organization identifiers, and routing credentials — that, if exfiltrated, would give an attacker the ability to make API calls on behalf of an organization against AI providers. Beyond direct credential value, an implant in LiteLLM occupies a privileged observation point in enterprise AI systems: it processes the prompts sent to AI models and the responses returned, giving it access to whatever sensitive information flows through AI agent operations.

The Backdoor Mechanism

Using credentials obtained from the Trivy pipeline compromise, TeamPCP inserted malicious code into LiteLLM versions 1.82.7 and 1.82.8 [3][12]. The injection mechanism exploited the dependency chain and publishing credentials rather than a vulnerability in LiteLLM itself — the attack was a supply chain intrusion against the package's distribution infrastructure, consistent with the template established by the Trivy compromise.

Reporting does not fully disclose the technical specifics of what the backdoored LiteLLM versions collected or transmitted, which is itself informative: at the time of publication, incident response investigation was ongoing, and the complete scope of data accessible to the implant — including whether AI prompt content was harvested — had not been established. What is confirmed is that the backdoored versions were distributed through PyPI with LiteLLM's normal release channels and were available for download before the compromise was detected and the packages were yanked [3].

Implications for Agentic AI Security

The LiteLLM compromise introduces a threat category that CSA's AI Safety Initiative has theoretical coverage for but that had not, until this campaign, been realized in a confirmed production incident: the supply chain compromise of AI orchestration infrastructure. AI agents built on frameworks that use LiteLLM for model routing — including many implementations based on LangChain, LlamaIndex, CrewAI, and custom agent frameworks — potentially executed compromised code in their model routing layer during the exposure window. An implant in this layer could, depending on implementation, observe prompts and responses, extract API keys for all configured LLM providers, or manipulate routing behavior.

This represents an expansion of the attack surface that enterprise AI risk models must account for. Supply chain risk for AI systems is not limited to model weights, training data, or the AI framework code itself; it extends to every library in the dependency chain that handles authentication material or processes data that flows through the AI system. Organizations that have not applied the same software composition analysis scrutiny to their AI infrastructure dependencies that they apply to their application security dependencies have a material gap.

Stage Three: CanisterWorm and the Geopolitical Payload

Propagation Mechanics

CanisterWorm is a self-propagating worm payload that uses credentials stolen in the Trivy compromise to seed malicious code across npm packages. Using the stolen SSH keys, Git credentials, and npm authentication tokens harvested from developer environments, CanisterWorm gained publish access to packages maintained by the developers whose credentials were stolen and injected itself as a dependency modification or as a directly injected payload into the package code [5][1]. By the time the campaign was publicly disclosed, the worm had propagated to 47 or more npm packages through this mechanism.

The worm's propagation stage is operationally significant because it converts a CI/CD credential theft — which requires the attacker to use credentials manually or through automation — into a self-sustaining operation that expands without requiring further attacker intervention. Each successfully infected npm package becomes a new vector for credential theft, potentially yielding additional npm publishing credentials that allow further propagation. The choice of npm as the propagation substrate reflects the

ecosystem's characteristics: npm packages frequently have many maintainers with varying security practices, package update mechanisms are widely automated, and the ecosystem's install-time execution model means malicious code runs during package installation before security tooling can inspect runtime behavior.

The Kubernetes Wiper Payload

The destructive payload delivered by CanisterWorm targets Kubernetes cluster infrastructure through a DaemonSet-based wiper mechanism. A Kubernetes DaemonSet is a workload resource that schedules a pod on every node in a cluster; TeamPCP's use of a DaemonSet for wiper delivery is architecturally deliberate, as it ensures that the destructive payload executes across all cluster nodes simultaneously rather than being containable through selective pod termination. The DaemonSet destroys node filesystems and forces node reboots, rendering the cluster unrecoverable through software-only recovery procedures [5][1].

On non-Kubernetes systems, CanisterWorm also deploys local wiper components that delete critical system files and overwrite filesystem structures. The combination of cluster-level and node-level destruction is consistent with a threat actor whose objective is maximum operational disruption with minimal opportunity for partial recovery.

The Geopolitical Trigger

Perhaps the most technically distinctive feature of CanisterWorm is an embedded geopolitical trigger: the malware inspects the system's locale and timezone configuration, and when it detects the `Asia/Tehran` timezone or Iranian locale settings, it activates the wiper payload without waiting for any additional command-and-control signal [5][1]. This trigger appears to function as both an operational safety mechanism — avoiding inadvertent destruction in environments the attacker wishes to preserve — and as a geopolitical statement embedded in the malware's code.

The design of this trigger merits careful analysis. It does not require network connectivity to activate, making it resistant to C2 takedown as a mitigation against wiper deployment. It activates based on system configuration rather than IP geolocation, meaning that a victim in any geography running infrastructure configured for an Iranian locale — for example, an Iranian diaspora organization, a multinational with Iranian operations, or a developer who has configured their system locale to match their native timezone while working outside Iran — would trigger the wiper. The precision of the trigger, combined with its irreversibility, strongly suggests the decision to include it was deliberate and operationally considered rather than incidental.

KrebsOnSecurity documented the first confirmed wiper deployment associated with this trigger in a March 23, 2026 report describing the activation against systems at an unidentified organization [5]. The incident represents one of the few publicly documented cases of geopolitically targeted destructive malware embedded within a financially motivated supply chain attack campaign — a combination that blurs the conventional distinction between criminal and state-adjacent threat actors.

The Cisco Source Code Breach

A credential thread from the Trivy compromise that did not lead to CanisterWorm propagation instead yielded access to Cisco Systems' internal development environment. BleepingComputer reported on March 31, 2026 that customer-facing source code was stolen in this breach, which was linked to credentials obtained through the Trivy pipeline [4]. Cisco's development environment breach illustrates the breadth of the downstream impact from a single high-value CI/CD credential harvest: the stolen credentials traced a path through an ordinary developer's CI/CD credentials to source code in a major enterprise vendor's internal repository, demonstrating that the lateral movement potential of cloud-native credential theft is constrained primarily by which organizations happen to share credential chains with each other.

Technical Analysis: The Kill Chain Structure

Attack Phase Mapping

TeamPCP's March 2026 campaign follows a coherent kill chain structure that can be mapped to established adversary behavior frameworks. Understanding the phases and the dependencies between them is essential for identifying where defensive controls would have disrupted the campaign.

The initial compromise phase exploited a misconfiguration in Trivy's GitHub Actions workflow — specifically the `pull_request_target` configuration that allowed an unauthenticated pull request from a fork to access repository secrets. This class of misconfiguration is well-documented in security literature and tooling documentation; GitHub itself publishes guidance on avoiding it. However, the misconfiguration persisted in a widely used, actively maintained security tool, which underscores that security teams responsible for their own tooling's security posture face the same vulnerability management challenges they address for application code.

Credential persistence across partial rotation represents the second critical phase. Aqua Security's detection and credential rotation response was competent but not complete, and TeamPCP's retained access through the surviving credentials allowed the group to proceed with tag poisoning after an initial disruption. The operational implication is that credential rotation in response to a suspected compromise must be comprehensive across all access tokens, service account credentials, and secret material associated with the affected system — a more demanding standard than routine credential rotation practices typically enforce.

The tag poisoning and malicious binary distribution phase exploited a broadly deployed assumption in CI/CD security: that version-pinned dependencies from trusted sources are safe. Security guidance on SHA-pinned dependencies has been available from GitHub and CI/CD security frameworks for several years, though exact adoption rates in production environments are not publicly documented. TeamPCP's ability to redirect 76 version tags to malicious code simultaneously demonstrates the magnitude of exposure that mutable version references create at scale.

The Self-Amplifying Cascade

A distinctive characteristic of this campaign is the way each stage amplified the reach of the next. The Trivy credential harvest provided not merely the keys to Trivy's own distribution infrastructure but a diverse collection of credentials from every CI/CD environment that executed the compromised scanner. This diversity of credential types — cloud provider credentials, container registry authentication, Git credentials, npm tokens — meant that the attacker gained access to distribution channels across multiple package ecosystems simultaneously, enabling parallel attacks on LiteLLM (via PyPI) and CanisterWorm propagation (via npm) using material collected through a single initial access event.

This cascade structure exposes a gap in how traditional attack surface analysis models third-party risk: the conventional model focuses on direct data and access exposure per dependency, not on the cumulative credential scope that a privileged pipeline phase accumulates. Security teams typically model their third-party risk in terms of the direct access and data exposure associated with each third-party dependency. The Trivy campaign demonstrates that a dependency used during a privileged pipeline phase accumulates the credentials of every other system that pipeline phase touches — transforming a single dependency compromise into a credential sweep across the entire CI/CD supply chain.

Implications for AI Security Infrastructure

The Agentic AI Supply Chain

The LiteLLM compromise demonstrates that agentic AI infrastructure faces supply chain risks that are structurally similar to, but operationally distinct from, those faced by traditional software. Traditional supply chain security focuses on ensuring that code dependencies do not contain malicious functionality. Agentic AI supply chain security must additionally address the runtime data plane: the prompts, responses, API keys, and context that flow through AI infrastructure represent sensitive material that a compromised intermediary library can observe, alter, or exfiltrate without modifying any application-layer behavior that a user would detect.

An AI agent running on a compromised routing library does not behave differently from the user's perspective. Its outputs may be unchanged. Its tool calls may be unaltered. The compromise operates in the substrate beneath the agent's observable behavior — a threat model that existing AI security monitoring approaches, which tend to focus on prompt injection and output manipulation, are not designed to detect.

Trust Model Failures in Cloud-Native AI

The Vertex AI privilege escalation vulnerability disclosed by Palo Alto Networks Unit 42 on March 31, 2026, though not directly connected to the TeamPCP campaign, reinforces the same underlying theme: cloud-native AI infrastructure is designed with implicit trust assumptions that attackers are now systematically exploiting [13]. The Per-Project Per-Product Service Agent (P4SA) for deployed Vertex AI workloads holds excessive default permissions, allowing a compromised AI agent to pivot across cloud infrastructure in ways that, according to Unit 42's analysis, the service's documentation does not make apparent [13]. The combination of a compromised LiteLLM installation and the default over-privileged cloud identity of the AI workload running it would create an attack path from supply chain compromise to cloud infrastructure control.

For enterprise AI security teams, this combination of risks argues for a defense architecture built around the principle of mutual distrust: AI orchestration libraries should be treated with the same scrutiny as external network connections; cloud identities for AI workloads should be minimally scoped; and network and data egress from AI workload environments should be monitored with the same fidelity applied to privileged infrastructure hosts.

Detection and Response Guidance

Detecting Compromised Trivy Execution

Organizations that ran `aquasecurity/trivy-action` or `aquasecurity/setup-trivy` between March 19 and March 23, 2026, without pinning to the safe commit SHAs should treat their CI/CD credential environments as potentially compromised. Detection indicators described in published threat intelligence and the Microsoft Security Blog include: outbound network connections from CI/CD runners to unfamiliar IP addresses or domains during scan execution; anomalous process spawning by the Trivy binary, particularly process tree activity inconsistent with a vulnerability scanner; and Runner.Worker memory access patterns that deviate from baseline scanner behavior [10].

Immediate containment actions for organizations in the exposure window should include rotation of all secrets accessible to the CI/CD environment in which Trivy ran, including cloud provider credentials, Kubernetes service account tokens, container registry credentials, and SSH keys used in deployment. Rotation should be comprehensive across all tokens associated with the pipeline environment, not limited to the credentials that appear most directly exposed.

GitHub, GitLab, and other CI/CD platform providers offer audit log capabilities that can identify which workflows ran the affected actions during the exposure window. Organizations should enumerate all workflow executions in this window and triage them based on the credential scope available to each runner.

Detecting CanisterWorm and LiteLLM Compromise

Organizations running LiteLLM versions 1.82.7 or 1.82.8 should downgrade to a known-good version and audit their AI provider API key usage for anomalous requests during the exposure window. API providers including OpenAI, Anthropic, and Google offer audit logs for API activity that can reveal unauthorized usage even if the compromised installation has been removed [3].

CanisterWorm's Kubernetes wiper is delivered via a DaemonSet workload, which provides a detection opportunity. Kubernetes audit logs record DaemonSet creation events, and DaemonSet creation by service accounts or identities that do not normally manage workloads should trigger high-priority alerting. Organizations running Kubernetes cluster admission control — whether through Open Policy Agent, Kyverno, or native admission webhooks — should validate that their policies block DaemonSet creation from non-administrative identities. Wiz Research has published indicators of compromise associated with the CanisterWorm propagation chain [1].

Hardening CI/CD Pipelines Against Future Tag Poisoning

The fundamental mitigation for tag poisoning attacks is pinning GitHub Actions references to immutable commit SHAs rather than mutable version tags. This practice ensures that a workflow references a specific, cryptographically identified commit that cannot be changed retroactively. Tools including Dependabot, Renovate, and dedicated GitHub Actions pinning utilities automate the conversion of tag-pinned references to SHA-pinned equivalents and can be configured to maintain those pins automatically as upstream releases are published.

Beyond pin hardening, organizations should implement cryptographic verification of GitHub Actions using provenance attestation where available. GitHub's artifact attestation capabilities, combined with SLSA framework compliance verification, provide a supply chain integrity verification layer that complements SHA pinning by confirming that a given commit was produced through an expected build process [14].

For container image scanning specifically, organizations should evaluate alternative scanner configurations, diversified toolchains, or airgapped distributions that reduce their exposure to compromise of a single upstream distribution channel.

Broader Threat Landscape Context

Preceding Campaign: Shai-Hulud 2.0

The TeamPCP campaign did not emerge in isolation. In December 2025, a supply chain worm called Shai-Hulud 2.0 demonstrated a closely related technique: compromising an npm package to deploy a weaponized version of TruffleHog — a secrets-scanning tool — that was reconfigured to steal secrets from victim environments rather than report them for remediation [15]. Shai-Hulud 2.0 established that security research tooling could be repurposed as an attack delivery mechanism at scale, and its propagation pattern through npm packages parallels CanisterWorm's propagation methodology.

The GhostAction campaign, also documented in 2025, compromised 327 GitHub users by injecting malicious workflows that stole 3,325 secrets across affected repositories [21]. Taken together, Shai-Hulud, GhostAction, and the TeamPCP campaign represent a coherent trajectory of escalating sophistication in CI/CD and supply chain attack methodology over a 12-month period. This pattern is consistent with deliberate capability development across the threat actor ecosystem, whether through shared tradecraft, common tooling markets, or parallel independent discovery of effective techniques.

The Compression of Attack Timelines

A structural feature of cloud-native attack campaigns that TeamPCP's operation illustrates is the dramatic compression of time-to-impact relative to traditional enterprise intrusions. Microsoft's threat intelligence reporting on recent cloud campaigns documents that attackers in cloud environments can achieve full compromise — from initial access to credential use across downstream systems — in under 10 minutes in well-orchestrated campaigns [10]. The Trivy compromise's cascade through LiteLLM, Cisco's development environment, and npm propagation occurred across a period of days rather than the weeks or months typical of traditional APT operations.

This compression changes the calculus for detection and response. A response playbook designed for "detect within 24 hours, contain within 72 hours" is structurally inadequate for a threat that can propagate from a single compromised CI/CD action to credential use across dozens of organizations' cloud environments within hours. CI/CD security monitoring must operate at a timescale commensurate with automated pipeline execution, with real-time or near-real-time alerting on anomalous behavior rather than batch log review.

Recommendations

Immediate Actions

Organizations that ran the affected versions of `aquasecurity/trivy-action` or `aquasecurity/setup-trivy` between March 19 and March 23, 2026 should treat the event as a confirmed compromise and proceed with credential rotation across all secrets accessible to those pipeline environments. Rotation that is limited to "likely exposed" credentials based on a partial audit underestimates the credential sweep the malicious binary performed. All credentials — cloud provider keys, Kubernetes tokens, SSH keys, container registry credentials, and Git authentication — accessible from the affected runner environment should be rotated.

Organizations running LiteLLM versions 1.82.7 or 1.82.8 should downgrade immediately and audit all AI provider API usage during the exposure window. If the organization's AI agents processed sensitive data through the affected versions, that data should be assumed observable by the attacker. Downstream notifications may be required depending on the sensitivity of the data and applicable regulatory requirements.

Short-Term Mitigations

GitHub Actions workflows should be audited for mutable tag references and converted to SHA-pinned equivalents. This remediation is automatable through tooling and addresses the root mechanism that made the Trivy tag poisoning universally effective. Organizations should also review their CI/CD secrets management practices, ensuring that the credential scope available to any given pipeline step is the minimum required for that step's function. Broad-scope credentials available in CI/CD runners represent a structural amplifier for credential theft regardless of which specific tool is compromised.

For AI infrastructure, organizations should apply the same software composition analysis scrutiny to AI framework dependencies — including model routing libraries, orchestration frameworks, and inference infrastructure — that they apply to application security dependencies. AI supply chain risk assessments should account for the data plane exposure of orchestration libraries, not only their code execution surface.

Kubernetes cluster admission control policies should be reviewed to confirm that DaemonSet creation is restricted to administrative identities. Where admission control is not deployed, organizations should evaluate its implementation as a near-term priority, as DaemonSet-based wiper delivery is not specific to CanisterWorm and represents a reusable attack pattern.

Strategic Considerations

The TeamPCP campaign argues for a structural re-examination of how enterprises model and manage risk in their security toolchain. The conventional security risk model treats the security toolchain as trusted infrastructure — a necessary simplification that worked adequately when the toolchain was relatively small and primarily composed of internally deployed software. The cloud-native toolchain, distributed across hundreds of open-source projects with complex dependency graphs and shared distribution infrastructure, does not support this simplification.

Enterprises should develop and maintain a software bill of materials for their security toolchain with the same discipline they apply to production application dependencies. Security tools that run with elevated CI/CD privileges — scanners, linters, infrastructure analyzers, secrets detectors — warrant scrutiny that is at least as rigorous as the applications they scan. Distribution channel integrity for security tooling should be verified through artifact attestation, checksum verification against authoritative sources, and, where feasible, reproducible builds.

Organizations building or running agentic AI systems should treat AI orchestration infrastructure as critical path infrastructure and apply corresponding security controls: privileged access management for AI provider credentials, network segmentation for AI workload environments, and monitoring of AI

infrastructure package updates with the same process discipline applied to production application deployments.

CSA Resource Alignment

MAESTRO and Agentic AI Threat Modeling

CSA's MAESTRO framework, designed for threat modeling multi-agent AI systems, provides the most directly applicable analytical structure for the AI-layer implications of this campaign. The LiteLLM compromise represents a MAESTRO-class threat at Layer 2 (Orchestration Layer), where the library that routes requests between AI agents and model providers becomes a potential observation point for prompt and response content. MAESTRO's threat categories for orchestration-layer compromise — including unauthorized tool invocation, context manipulation, and credential exfiltration through AI infrastructure — are directly relevant to the risk profile of a compromised model routing library. Security teams applying MAESTRO to their AI architecture should explicitly include third-party AI infrastructure dependencies in their threat modeling scope, accounting for the data and credential access those dependencies hold.

Cloud Controls Matrix (CCM) and AICM

Several domains within CSA's Cloud Controls Matrix address controls directly relevant to the TeamPCP attack chain. The Supply Chain Management, Transparency, and Accountability domain (STA) encompasses vendor risk assessment and supply chain integrity verification controls that apply to CI/CD pipeline dependencies, including security tooling. The Identity and Access Management domain (IAM) covers credential management practices relevant to CI/CD secrets hygiene, including the minimum-privilege principles that, if applied consistently to pipeline runner credentials, would have limited the blast radius of the Trivy compromise. The Application and Interface Security domain (AIS) addresses software composition analysis and dependency integrity verification that provides detective and preventive controls against supply chain attacks.

CSA's AI Controls Matrix (AICM), as a superset of CCM that extends coverage to AI-specific risks, adds control domains for AI system supply chain integrity, AI inference infrastructure security, and AI data pipeline protection that directly address the LiteLLM compromise scenario. Organizations developing AICM compliance programs should include AI orchestration library dependencies in their supply chain inventory and apply the AICM's AI-specific controls to the full dependency chain of their AI systems.

STAR and Third-Party AI Risk

CSA's STAR (Security Trust Assurance and Risk) program provides a mechanism for organizations to assess and disclose the security posture of cloud services, including AI platform providers. In the context of the Vertex AI privilege escalation vulnerability and the LiteLLM supply chain compromise, STAR assessments for AI platform vendors should include scrutiny of default service agent permission scopes, supply chain integrity practices for distributed AI libraries, and incident response capability for supply chain compromises affecting AI infrastructure.

DevSecOps Six Pillars Framework

CSA's Six Pillars of DevSecOps framework, and specifically the Pragmatic Implementation pillar's guidance on CI/CD security, provides an established baseline for the pipeline hardening practices that this campaign highlights. The framework's guidance on secrets management, pipeline integrity verification, and supply chain security — developed before the scale of CI/CD supply chain attacks became apparent — remains directionally correct and warrants renewed emphasis in light of TeamPCP's demonstrated exploitation of gaps in each area.

Zero Trust Architecture Principles

CSA's Zero Trust guidance is directly applicable to the architectural failures that amplified this campaign's impact. The credential material available to CI/CD pipeline runners in affected organizations routinely encompassed far more access than any given pipeline step required. A Zero Trust architecture for CI/CD infrastructure would apply the principle of least privilege at the pipeline step level, ensuring that a security scanning step holds only the credentials needed for scanning and not the deployment credentials, cloud API keys, or SSH material that legitimate scanner operations never require. This granularity of access control would not have prevented the Trivy compromise but would have substantially limited the credential sweep its payload could perform.

Conclusions

The TeamPCP campaign of March 2026 represents a significant escalation in cloud-native threat actor capabilities, extending the CI/CD supply chain attack techniques demonstrated by earlier campaigns to achieve industrial-scale credential harvest and multi-ecosystem impact. The group's operational template — compromise a broadly trusted component of the security toolchain, harvest credentials at

scale through that component's privileged CI/CD position, then use the harvested credential diversity to execute parallel operations across multiple ecosystems — is replicable, improvable, and now demonstrated to be effective at industrial scale. This operational template is replicable by other sophisticated threat actors, and security teams should anticipate similar campaigns targeting security toolchain components.

For the security industry, this campaign carries a lesson that is uncomfortable precisely because it implicates the field's own practices. The security toolchain — the scanners, analyzers, and detectors that organizations deploy to protect their software supply chains — is itself a supply chain requiring protection. The widespread expectation that security tools are safe by virtue of their purpose is a cultural assumption that has not been tested rigorously at the distribution layer, and TeamPCP's operation demonstrates that this assumption fails badly when it is tested by a capable adversary.

The AI security implications are equally significant. The LiteLLM compromise introduced a supply chain attack against AI orchestration infrastructure that had not previously been documented in a production incident. It validates the threat model that CSA's MAESTRO framework anticipates and confirms that AI infrastructure supply chain risk is not hypothetical. Organizations that have built AI agent systems without considering the security posture of their AI framework dependencies have inherited an unmodeled risk that this campaign makes concrete.

The geopolitical trigger embedded in CanisterWorm adds a dimension that resists simple categorization. A financially motivated criminal group embedding a geopolitically targeted wiper in its payload suggests either ideological motivation, a state-adjacent relationship, or an attempt to create deniability for targeted destructive operations by embedding them in financially motivated campaigns. The attribution ambiguity is itself operationally significant: defenders cannot rely on the conventional distinction between financially motivated criminals who do not deploy wipers and geopolitically motivated state actors who do.

References

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

[2] Aqua Security. "Trivy Supply Chain Attack: What You Need to Know." Aqua Security Blog, March 2026.

[3] Endor Labs. "TeamPCP Isn't Done: Hits LiteLLM's 95M Monthly Downloads." Endor Labs Blog, March 2026.

[4] BleepingComputer. "Cisco source code stolen in Trivy-linked dev environment breach." BleepingComputer, March 31, 2026. (Specific article URL unavailable at publication time.)

[5] KrebsOnSecurity. "CanisterWorm Springs Wiper Attack Targeting Iran." KrebsOnSecurity, March 23, 2026. (Specific article URL unavailable at publication time.)

[6] Wiz Research. "KICS GitHub Action Compromised: TeamPCP Strikes Again in Supply Chain Attack." Wiz Blog, March 2026.

[7] Wiz Cloud Threat Landscape. "TeamPCP Threat Actor Profile." Wiz Intelligence, 2026.

[8] Cyble. "TeamPCP Threat Actor Profile." Cyble Research, 2026. (URL `https://cyble.com/threat-actor-profiles/teampcp/` may require authentication to access.)

[9] Flare. "TeamPCP: An Emerging Force in the Cloud Native and Ransomware Landscape." Flare Blog, 2026.

[10] Microsoft Security Blog. "Detecting, Investigating, and Defending Against the Trivy Supply Chain Compromise." Microsoft, March 24, 2026.

[11] Sysdig. "TeamPCP Expands: Compromise Spreads from Trivy to Checkmarx GitHub Actions." Sysdig Blog, March 2026.

[12] The Hacker News. "TeamPCP Backdoors LiteLLM Versions 1.82.7–1.82.8 via Trivy CI/CD Compromise." The Hacker News, March 2026. (Specific article URL unavailable at publication time.)

[13] The Hacker News. "Vertex AI Vulnerability Exposes Google Cloud Data and Private Artifacts." The Hacker News, March 31, 2026. Citing Palo Alto Networks Unit 42 research by Ofir Shaty. (Specific article URL unavailable at publication time.)

[14] GitHub Security. "GitHub Advisory: GHSA-69fq-xp46-6x23." GitHub Security Advisories, March 2026.

[15] Microsoft Security Blog. "Shai-Hulud 2.0: Guidance for Detecting, Investigating, and Defending Against the Supply Chain Attack." Microsoft, December 9, 2025.

[16] CrowdStrike. "From Scanner to Stealer: Inside the trivy-action Supply Chain Compromise." CrowdStrike Blog, March 2026.

[17] Palo Alto Networks Unit 42. "Weaponizing the Protectors: TeamPCP's Multi-Stage Supply Chain Attack." Unit 42 Blog, March 2026.

[18] Help Net Security. "TeamPCP's attack spree slows, but threat escalates with ransomware pivot." Help Net Security, March 30, 2026.

[19] Phoenix Security. "Trivy Supply Chain Compromise: A Security Scanner Weaponised Across 10,000 Pipelines." Phoenix Security Blog, March 2026.

[21] GitGuardian. "The GhostAction Campaign: 3,325 Secrets Stolen Through Compromised GitHub Workflows." GitGuardian Research, September 2025. (Source for GhostAction campaign statistics — 327 GitHub users compromised, 3,325 secrets stolen.)