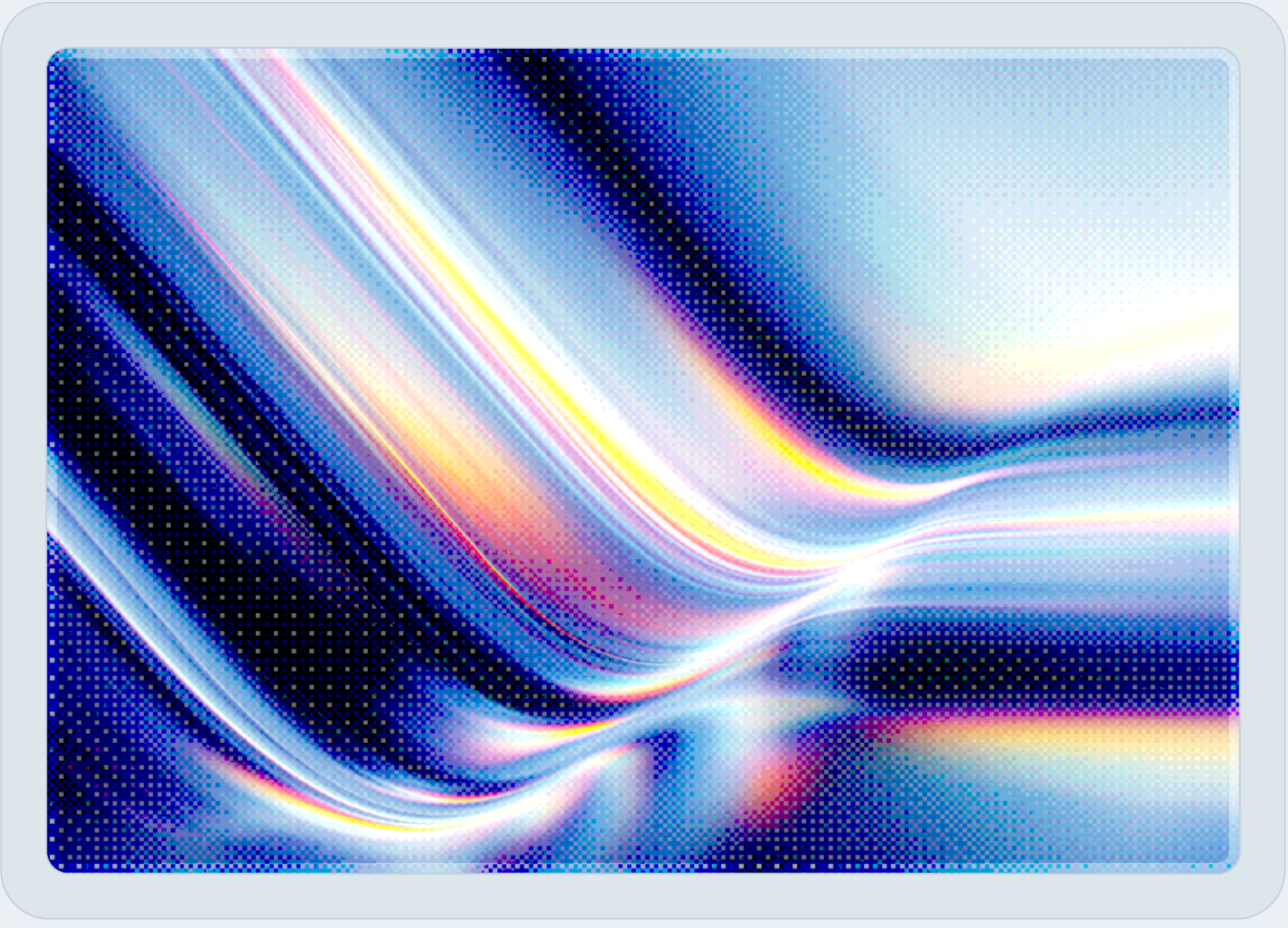
CSAI Foundation | Cloud Security Alliance

AI-Native Adversaries: Concentration Risk in the Enterprise AI Stack

How TeamPCP's 2026 Campaigns Exposed a Systemic Attack Surface in AI Tooling

2026-05-15

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Table of Contents

Executive Summary

Between March and May 2026, the threat group known as TeamPCP executed a series of supply chain attacks that constitute a turning point in adversarial behavior against enterprise AI infrastructure. Rather than targeting individual organizations directly, TeamPCP systematically compromised the shared tools that AI development teams across thousands of organizations rely upon — vulnerability scanners, AI gateways, LLM SDKs, and AI coding agent configuration directories. The campaigns were not opportunistic. They demonstrated a calculated understanding of how modern AI development toolchains are structured and where leverage points concentrate.

The March 2026 wave began with the compromise of Aqua Security's Trivy vulnerability scanner, a tool that runs inside CI/CD pipelines at more than 10,000 organizations globally [1]. From there, TeamPCP pivoted through Checkmarx's KICS scanner, then to the LiteLLM AI gateway — a package downloaded approximately 3.4 million times per day [5] — and finally to the Telnyx SDK, all within eight days. Each step leveraged credentials stolen in the prior step. The May 2026 wave, dubbed Mini Shai-Hulud, took a different approach: a self-propagating worm that exploited OIDC trust federation in GitHub Actions to publish malicious versions of over 170 packages — including Mistral AI SDKs, Guardrails AI, TanStack, UiPath automation tools, and the OpenSearch JavaScript client — across npm and PyPI, reaching packages with a combined download count exceeding 518 million [8]. Both waves demonstrated clear targeting of AI developer credentials: API keys for every major commercial LLM provider, AI coding agent configuration files, and the infrastructure secrets needed to operate AI workloads in cloud environments.

The concentrated nature of the AI tooling ecosystem amplified every step of these campaigns. LiteLLM is not merely a Python package; it functions as the routing layer between enterprise applications and potentially dozens of LLM providers simultaneously. A single compromised version — live on PyPI for approximately forty minutes on March 24 [5] — was capable of exfiltrating every API key, cloud credential, and Kubernetes secret from every environment that pulled it. The blast radius of a traditional library compromise scales with adoption. The blast radius of an AI gateway compromise scales with the number of LLM providers, cloud environments, and production pipelines that flow through it.

This paper argues that the TeamPCP campaigns should be understood not as isolated supply chain incidents but as adversarial probing of a structural property of enterprise AI stacks: the AI toolchain has consolidated around a small number of high-leverage packages, and that consolidation creates systemic risk that exceeds what any individual organization's vulnerability management program is designed to handle. Security teams that treat these events as "patch and move on" incidents will remain exposed. The appropriate response is a structural one, addressing dependency concentration, credential segmentation, and the emerging attack surface created by AI coding agents.

Introduction and Background

Enterprise adoption of AI tooling has followed a familiar pattern: rapid initial fragmentation, followed by consolidation around a small number of dominant libraries, frameworks, and gateways. In 2023 and 2024, AI engineering teams faced a rapidly expanding field of options — dozens of LLM providers, hundreds of orchestration frameworks, and an emerging ecosystem of evaluation, guardrail, and observability tools. By 2025, consolidation was well underway. LangChain and LlamaIndex had become widely adopted orchestration frameworks. LiteLLM had emerged as a broadly deployed open-source AI gateway for organizations seeking a unified interface to multiple LLM providers — a position reflected in its reported download volume of approximately 3.4 million daily downloads [5]. Guardrails AI had become a frequently referenced framework for output validation, and Mistral's Python SDK had seen broad adoption among developers integrating European open-weight models into production applications.

This consolidation followed a pattern familiar from prior technology markets, driven by operational incentives that favored reduced integration complexity and shared institutional knowledge. Standardizing on a small number of AI infrastructure components enables teams to develop deep expertise in fewer tools and accelerates the pace at which organizations can deploy AI capabilities. But it also creates the structural preconditions for the kind of attacks TeamPCP executed: when thousands of organizations use the same AI gateway, compromise that gateway and you compromise all of them simultaneously. The concentration that makes AI development efficient also makes it systemically fragile.

The concept of concentration risk is not new to security. Financial regulators apply it to portfolios. Infrastructure architects apply it to cloud dependencies. The software supply chain security community applies it to shared open-source packages. What is new is its application to the AI tooling layer specifically — a layer that sits at the intersection of developer credential management, LLM provider access, cloud infrastructure credentials, and increasingly, autonomous agent execution. AI tooling does not merely process code; it processes the keys that unlock cloud environments, the secrets that authenticate production services, and the configuration files that govern what AI agents are permitted to do. When that layer is compromised, the attacker's access extends not just to the software artifact but to the operational infrastructure it connects.

TeamPCP's campaigns make this abstract risk concrete. This paper analyzes the full scope of those campaigns, examines the specific mechanisms by which concentrated AI tooling amplified the attacks' reach, and draws out the implications for enterprise security strategy. Section three covers the anatomy of the March 2026 campaign. Section four analyzes the May Mini Shai-Hulud wave and its novel AI agent persistence mechanisms. Section five develops the concentration risk argument in detail. Section six addresses enterprise implications and the alignment between these findings and CSA's established security frameworks. The paper concludes with specific strategic recommendations.

The March 2026 Campaign: A Calculated Kill Chain

Phase One: Turning Security Infrastructure Against Itself

TeamPCP's operational logic in March 2026 reflected a clear strategic calculus: instead of attacking organizations directly, the group attacked the tools organizations use to protect themselves. Aqua Security's Trivy is among the most widely deployed open-source vulnerability scanners in the world, embedded by default in CI/CD pipelines across enterprise, startup, and government environments. Its GitHub Actions workflow — used to automate scanning as part of software release processes — ran with permissions that made it a valuable credential target.

On March 19, 2026, TeamPCP exploited a `pull_request_target` misconfiguration in Trivy's GitHub Actions workflows to steal a Personal Access Token with sufficient privileges to take over the repository [2]. The subsequent attack was methodical. The group force-pushed malicious code to 75 existing tags in the `aquasecurity/trivy-action` repository [3]. GitHub Actions workflows typically reference Actions by tag rather than by commit hash — a performance convenience that becomes a security liability when the tag can be moved. Any workflow in any organization's repository that referenced `aquasecurity/trivy-action@v0.x.x` at those tag versions now silently executed TeamPCP's code rather than Aqua's [3]. The malicious payload — which the attacker infrastructure described as a "TeamPCP Cloud Stealer" — harvested SSH keys, cloud provider credentials, Kubernetes secrets, database passwords, CI/CD tokens, and cryptocurrency wallet files from GitHub Actions runner environments, then exfiltrated them encrypted using AES-256 with RSA-4096 key wrapping to attacker-controlled infrastructure [1].

The Microsoft Security Response Center, which published guidance within hours of the compromise being discovered, estimated global impact at more than 10,000 CI/CD workflows [4]. The majority of affected organizations would have had no awareness that their Trivy-based security scanning had been subverted. The tool continued to return vulnerability scan results. The credential exfiltration ran silently in the background.

Phase Two: Credential Laundering Through the Toolchain

Two days after the Trivy compromise, on March 21, 2026, TeamPCP used credentials harvested from Trivy-compromised pipelines to attack Checkmarx's KICS scanner, force-pushing malicious commits to 35 version tags of the `checkmarx/kics-github-action` repository [15][27]. KICS — Keep It Clean and Safe — is an infrastructure-as-code security scanner used by enterprise DevSecOps teams to identify misconfigurations in Terraform, Helm, and CloudFormation templates. Its compromise followed the same tag-poisoning methodology as Trivy and similarly targeted credentials from CI/CD runner environments.

The pattern established a fundamental problem for defenders: because each compromise yielded new credentials that could be used in the next attack, the campaign's blast radius expanded combinatorially. An organization compromised through Trivy might find that the credentials stolen from their pipeline were then used to compromise a third-party tool they also used, and so on. Credential compartmentalization — the practice of ensuring that different systems authenticate with different, isolated credentials — was the control that would have broken this chain. In many affected organizations, credential compartmentalization appears to have been absent — a conclusion consistent with widely documented CI/CD credential hygiene challenges across enterprise environments.

Phase Three: Compromising the AI Gateway

On March 24, 2026, the campaign reached its most consequential target. LiteLLM's CI/CD pipeline included a Trivy scan step that pulled the scanner from `apt` without a pinned version — meaning it consumed whatever version the compromised release infrastructure was currently serving. That scan step ran with access to the `PYPI_PUBLISH` token used to authenticate LiteLLM releases to the Python Package Index [5]. TeamPCP extracted that token from the runner environment and used it to publish two backdoored versions: `litellm 1.82.7` at 10:39 UTC and `litellm 1.82.8` at 10:52 UTC [5][6].

LiteLLM is not a peripheral tool in the AI development ecosystem. It functions as an abstraction gateway, providing a single OpenAI-compatible API interface that routes requests to more than 100 LLM providers including Anthropic, OpenAI, Google Gemini, Amazon Bedrock, Azure OpenAI, Mistral, and dozens of others. Organizations that adopt LiteLLM place every LLM provider credential they hold behind a single package. The malicious payload embedded in versions 1.82.7 and 1.82.8 targeted this property explicitly: in addition to the standard credential sweep seen in the Trivy and KICS payloads, the LiteLLM-specific payload extracted LLM provider API keys from environment variables, configuration files, and runtime memory [5][18].

In Kubernetes environments — where LiteLLM is frequently deployed as a centralized routing service — the payload went further. It deployed privileged pods to every node in the target cluster, enabling lateral movement across the entire Kubernetes environment, and installed a persistent systemd unit that polled attacker infrastructure every 50 minutes for additional command-and-control instructions [5][19]. The sophistication of the Kubernetes-specific payload suggests that TeamPCP had studied LiteLLM's deployment patterns and written exploit code tailored to the most common production architecture.

PyPI administrators quarantined the malicious versions approximately 40 minutes after the first was published [5]. But LiteLLM's download volume — reported at approximately 3.4 million downloads per day, with the package trusted by Fortune 500 enterprises and federal agencies [5] — meant that the window was sufficient for widespread exposure. LiteLLM's own security disclosure [6] acknowledged the compromise and advised all users who had pulled the package during the window to rotate credentials and audit their environments for indicators of compromise.

By March 27, TeamPCP had also compromised the Telnyx SDK — a telephony and communications library with its own enterprise deployment footprint — using the same credential-chaining approach [1]. The FBI Cyber Division issued a formal advisory that same day confirming the scope of the campaign [7].

The May 2026 Campaign: A Self-Propagating Worm Targets AI Infrastructure

The Mini Shai-Hulud Mechanism

The Mini Shai-Hulud campaign, which peaked on May 11, 2026, represented a tactical evolution from the targeted credential-chaining approach of March [8]. Rather than pivoting from one compromised project to the next via stolen credentials, TeamPCP deployed a self-propagating worm that used OIDC trust federation — a modern, security-forward CI/CD authentication mechanism — as its propagation vector.

The worm's entry technique exploited a subtle but widespread misconfiguration. Many open-source projects have configured GitHub Actions workflows to use OpenID Connect trusted publishing, which allows CI/CD pipelines to authenticate to npm or PyPI without storing long-lived credentials. This is considered security best practice. However, OIDC federation grants publish access to any workflow run that satisfies the trust policy, including workflows triggered from repository forks and orphaned workflow files that maintainers may have forgotten exist. TeamPCP identified such orphaned configurations in the TanStack repository ecosystem, pushed to the relevant fork, triggered the trusting workflow, and extracted a short-lived OIDC token from runner process memory — all without ever possessing a credential in the traditional sense [10][17].

Once publish access was established, the worm added `preinstall` lifecycle hooks to compromised packages. Any developer or CI pipeline that ran `npm install` or `pip install` against affected packages would execute the hook, which downloaded the Bun JavaScript runtime and a 2.3 MB obfuscated payload. That payload swept credentials, then used harvested publishing tokens to infect additional packages, creating the self-propagating characteristic that earned the campaign its name [17]. Within five hours, over 400 malicious versions of 170 packages had been published to npm and PyPI [8].

Targeting the AI Toolchain

The packages compromised in Mini Shai-Hulud were not a random sample of the open-source ecosystem. TeamPCP selected targets with a clear logic: they chose packages that sit at the center of enterprise AI development workflows. The full scope of affected packages included all three Mistral AI SDKs on both npm and PyPI, the Guardrails AI framework, the TanStack router and query libraries (heavily used in AI application

front-ends), the OpenSearch JavaScript client, the entire UiPath `@uipath` npm scope (65 packages), and various additional AI tooling dependencies [8][11]. TeamPCP also claimed to have obtained access to Mistral AI's source code repositories and advertised them for sale [25][28].

The credential harvesting payload targeted file paths specifically associated with AI developer tooling, including Anthropic API keys, OpenAI credentials, LLM configuration files, and MCP (Model Context Protocol) server configuration directories [16]. This specificity indicates that TeamPCP had mapped the AI developer credential footprint in advance and built their credential sweeper to match it — not simply reusing generic infostealer code, but developing AI-native collection capabilities.

AI Coding Agent Configuration as a Persistence Vector

The most technically novel aspect of Mini Shai-Hulud, documented by StepSecurity and described in CSA's companion research note on the campaign, was its persistence mechanism inside AI coding agent configuration directories [12]. The malware wrote hooks into `.claude/settings.json` — the Claude Code agent configuration file — and `.vscode/tasks.json` using the `runOn: folderOpen` trigger. These hooks execute not when an infected package runs, but when a developer opens any project directory in an environment where Claude Code or VS Code is installed [12].

The implications extend beyond conventional supply chain persistence. Traditional package-based persistence depends on the infected package remaining installed. This mechanism survives package removal: the hooks are written to the developer's personal configuration directories, not to the package itself. When the infected package is removed and replaced with a clean version, the configuration-level backdoor remains. Any subsequent project the developer opens reactivates the payload. On macOS, the malware additionally installed a `LaunchAgent` plist; on Linux, a systemd user unit, providing reboot persistence independent of project or package state [12][16].

To our knowledge, this is the first publicly documented supply chain attack to exploit AI coding agent configuration as a persistence substrate — a technique that, while mechanically straightforward, targets a trust relationship with significant implications for developer environments. Its significance lies not in the mechanism itself — writing to configuration directories is a known technique — but in the target: AI coding agents are becoming deeply integrated into software development workflows, and their configuration files grant significant implicit trust. A hook installed in `.claude/settings.json` that executes on every project open has, from the perspective of the developer's environment, legitimate authority to invoke shell commands. It runs with the developer's credentials, in their environment, with access to their active workspace.

Concentration Risk: The Structural Problem

The Geometry of the AI Tooling Stack

Understanding why the TeamPCP campaigns were as damaging as they were requires understanding the topology of the enterprise AI tooling stack. Traditional software supply chain attacks derive their leverage from a package's adoption breadth: if a malicious package is installed by millions of developers, millions of developers are affected. The AI tooling stack creates an additional dimension of leverage: architectural centrality.

Consider LiteLLM's position in a typical enterprise AI deployment. Development teams build AI-powered applications using a variety of LLM providers — perhaps Anthropic Claude for reasoning tasks, OpenAI GPT-4o for code generation, and Google Gemini for document processing. Managing separate authentication, rate limiting, cost tracking, and failover logic for each provider is operationally complex. LiteLLM solves this by presenting a single unified API: applications call one endpoint, and LiteLLM routes to the appropriate provider. The result is that a single LiteLLM deployment handles every LLM API call across an organization's entire AI application portfolio, authenticated with every LLM provider API key the organization holds.

When that single component is compromised, the attacker does not merely gain the credentials for one LLM provider. They gain credentials for all of them, simultaneously, from the production environment where those credentials are actively used. The concentration that makes LiteLLM operationally efficient — its position as the single unified interface to all LLM providers — is precisely the property that makes its compromise so consequential. The blast radius of a LiteLLM compromise is not proportional to its individual adoption; it is proportional to the API key footprint of every organization that has routed their AI traffic through it.

The Second-Order Concentration Problem

The TeamPCP campaigns also exposed a second-order concentration problem that has received less attention: the concentration of trust in AI developer tooling ecosystems amplifies the speed and difficulty of defense. When Trivy's GitHub Actions were compromised, the attack propagated invisibly through legitimate workflows. There was no malicious dependency to detect, no unusual network request to flag. The CI/CD pipeline continued to run, continued to produce scan results, continued to pass security gates — while silently exfiltrating credentials in the background. The trust that organizations had placed in Trivy as a security tool became the mechanism of attack.

This inversion — security tools turned into attack vectors — appears in both major waves of the TeamPCP campaign, suggesting it is a deliberate strategic choice rather than coincidence. Trivy is a vulnerability scanner. KICS is an infrastructure security scanner. LiteLLM includes guardrail integration points. Guardrails AI is explicitly a safety and compliance framework. TeamPCP targeted each of these tools in sequence. The

pattern suggests that the group understood something about the AI security tooling ecosystem that many defenders had not yet fully internalized: the tools used to secure AI deployments are themselves part of the supply chain, and they tend to run with elevated trust, broad network access, and rich credential stores. They are attractive targets precisely because they are security infrastructure.

The Ardan Labs analysis of the LiteLLM incident articulated this problem clearly, describing LiteLLM's gateway position as creating a single point of trust failure — an architectural pattern in which one component's integrity is assumed by the entire downstream system [20]. When that assumption is violated, there is no secondary verification layer to catch the violation. The enterprise application calling LiteLLM trusts that LiteLLM is passing its requests to the intended provider. It has no visibility into whether LiteLLM's runtime code has been modified.

Quantifying the Exposure Window

One of the most significant aspects of the March 2026 LiteLLM compromise is how short the malicious package's availability window was relative to its potential impact. PyPI administrators quarantined the malicious versions within approximately 40 minutes [5]. Yet LiteLLM's download rate — approximately 3.4 million per day at the time of the incident [5] — means that roughly 95,000 download events may have occurred during that window. The packages are used in automated CI/CD pipelines that build and test continuously, not just during business hours. Organizations with fast-moving deployment pipelines, particularly those running continuous integration across multiple time zones, faced elevated exposure.

The standard enterprise response to supply chain compromise — "update to the latest clean version and rotate credentials" — is necessary but insufficient against a gateway compromise. Rotating the API keys that LiteLLM managed requires rotating credentials with every LLM provider, every cloud provider, and potentially every downstream service whose credentials were present in the compromised environment at the time of infection. For large enterprises with mature AI deployments, that rotation exercise can span dozens of provider relationships and take days to complete. During that window, organizations face the uncomfortable choice between continuing to operate with potentially compromised credentials or taking AI-dependent production services offline.

The Cascading Failure Pattern in AI Infrastructure

The March 2026 campaign demonstrated that AI tooling concentration creates not just larger individual blast radii but also cascading failure potential. Because TeamPCP used credentials from each compromised tool to attack the next, a successful defense at any single point could have shortened or terminated the chain. But defenses rarely work that way in practice. Trivy's compromise was not detected immediately; it

persisted long enough for LiteLLM's Trivy-dependent CI pipeline to pull the compromised version and expose its PyPI publishing credentials. The cascade ran faster than organizational detection and response cycles.

This is a systemic property, not an individual organization's failure. Supply chain attacks exploit the fact that the trust relationships between components in a software ecosystem are not designed with adversarial pivoting in mind. Package managers assume that the registry is trustworthy. GitHub Actions runners assume that the Actions they reference are unmodified. CI/CD pipelines assume that the scanning tools they invoke are running legitimate code. None of these assumptions are inherently unreasonable — they are necessary for a functional development ecosystem — but collectively they create attack paths that span organizational and tool boundaries in ways that no single organization's security controls can fully address.

Before drawing strategic conclusions from this analysis, it is worth acknowledging that concentration also creates genuine defensive advantages. When a single package is quarantined at the registry level, the action reaches every downstream organization simultaneously — the PyPI quarantine of LiteLLM's malicious versions within approximately 40 minutes is itself an example of this dynamic working in defenders' favor. A patch to a central package reaches the entire user base with the next dependency update, without requiring thousands of organizations to independently discover and apply remediation. Active monitoring of a small number of critical AI packages is operationally tractable in ways that monitoring a fragmented, bespoke-dependency landscape would not be. These advantages are real and should not be discounted. The argument advanced here is not that AI tooling concentration should be reversed, but that its risk properties — specifically, the amplified blast radius of any single compromise and the cascading failures that architectural centrality enables — have not been adequately incorporated into enterprise AI security strategies. The structural mitigations this paper advocates are designed to preserve the operational benefits of concentration while reducing the systemic exposure that TeamPCP exploited.

Enterprise Implications and the AI-Native Adversary

Defining the AI-Native Adversary

This paper introduces the term *AI-native adversary* as an analytical construct to describe a threat actor that has specifically modeled the AI development ecosystem, identified its structural leverage points, built collection capabilities tailored to AI-specific credential formats and file paths, and developed persistence mechanisms inside AI agent infrastructure. (This term is an original analytical coinage in this paper; it had not achieved standard industry usage at the time of publication.) The category is distinct from conventional supply chain actors, who may encounter AI tools incidentally in the course of broader campaigns but have not specifically built capabilities to exploit the AI tooling ecosystem as a primary target.

TeamPCP exemplifies this category. The evidence includes the targeting logic of Mini Shai-Hulud — which selected AI SDK packages rather than the highest-download packages across all of npm — the LiteLLM payload's specific sweep for LLM API keys, the exploitation of AI coding agent configuration files for persistence, and the group's claimed access to Mistral AI's source repositories [25][28]. These are not the actions of an opportunistic actor applying generic infostealer tradecraft to whatever is popular. They reflect deliberate intelligence collection about the AI development toolchain and deliberate capability development to exploit it.

AI-native adversaries present different targeting priorities than conventional software supply chain actors. Their primary objective is not necessarily code execution on end-user systems, though that capability was clearly present in the TeamPCP payloads. Their primary objective appears to be the acquisition of AI infrastructure credentials: LLM API keys, model deployment tokens, training pipeline access, and proprietary dataset exposure. These credentials have monetary value — both through direct resale and through the competitive intelligence they enable — and their theft may not be immediately detectable by the credential's owner, since API key exfiltration leaves no visible modification to the compromised system.

The Extortion Amplifier

The March 2026 campaigns did not end with credential theft. On March 25, 2026, TeamPCP announced a partnership with the Vect ransomware-as-a-service group, pivoting from credential theft to active extortion [7][13]. Halcyon AI's tracking of the Vect group documented the Trivy supply chain compromise entering its extortion phase shortly after the announcement [13]. The FBI advisory of March 27 confirmed that TeamPCP was attempting to leverage compromised AI software access for broader network penetration, ransomware deployment, and extortion [7].

This escalation trajectory — credential theft, then ransomware partnership — follows a pattern that security researchers have documented in other financially motivated threat groups. What makes the AI tooling context distinctive is the sensitivity of the credentials involved. LLM API keys at high-volume inference workloads may carry direct billing exposure in the hundreds of thousands of dollars, while access to proprietary model weights, training datasets, and production AI pipelines may represent competitive intelligence with significant market value — factors that combine to create extortion leverage qualitatively different from what a conventional enterprise network breach would yield. The leverage available to an extortionist holding AI infrastructure credentials scales with an organization's depth of AI adoption in ways that have no direct analogue in traditional network compromise scenarios.

The Developer Workstation as Enterprise Risk

Mini Shai-Hulud's exploitation of AI coding agent configuration files introduced a risk vector that enterprise security programs have not traditionally included in their scope: the individual developer workstation, specifically the AI coding assistant configuration that developer has installed. Enterprise security programs generally address server-side and CI/CD-side supply chain risks through centralized controls: software composition analysis in the build pipeline, registry proxies that cache and screen dependencies, and endpoint detection and response on production servers. Developer workstations have historically been treated as endpoints within the enterprise perimeter — protected by antivirus and MDM — but not as part of the AI supply chain threat model.

The Mini Shai-Hulud persistence mechanism changes that calculus. A developer who installs an infected package from a compromised npm scope has, after package removal, a backdoored Claude Code or VS Code configuration that will reactivate on every subsequent project open. If that developer works on multiple internal projects — as most developers do — the hook executes in each project context, with access to the credentials, tokens, and secrets that those projects contain. The attack surface is the developer, wherever they open a project, on whatever network, with whatever mix of internal and external repository access they have at the moment.

This is a materially different threat model than the server-side supply chain attack. It requires enterprise security programs to extend supply chain thinking to developer workstations, AI coding agent configurations, and the trust relationships between individual developers and the shared npm/PyPI ecosystems their tools consume.

CSA Resource Alignment

The TeamPCP campaigns and the structural vulnerabilities they exploited connect directly to several active areas of CSA research and framework development that provide practical guidance for enterprise response.

CSA's MAESTRO framework (Multi-Agent Environment Security Threat Reference Overview), which addresses agentic AI threat modeling, is directly applicable to the AI coding agent persistence mechanisms documented in Mini Shai-Hulud. MAESTRO's threat model for agent-level attacks anticipates adversaries that seek to manipulate agent configuration and tool access to achieve persistence and lateral movement — precisely the attack pattern used against `.claude/settings.json` and `.vscode/tasks.json`. Organizations applying MAESTRO guidance to their AI agent deployments should treat agent configuration directories as security-sensitive artifacts, applying the same change-detection and integrity-monitoring controls they would apply to authentication configuration files.

The AI Controls Matrix (AICM) v1.0 provides a structured framework for addressing supply chain security in AI deployments. AICM Domain SC (Supply Chain) addresses dependency management, third-party tool assessment, and artifact integrity verification for AI components. The LiteLLM compromise illustrates a gap that AICM SC controls are designed to address: organizations that had implemented AICM SC-3 (Artifact Integrity) controls — requiring cryptographic verification of package artifacts against known-good hashes — would likely have detected the tampered LiteLLM versions before installation. AICM Domain IM (Identity Management) addresses the credential segmentation failures that allowed TeamPCP's cascade: the practice of using the same CI/CD credential footprint for both security scanning and artifact publishing created the pivot path that TeamPCP exploited.

CSA's Zero Trust guidance is relevant to the Kubernetes-specific persistence mechanisms in the LiteLLM payload. Zero Trust architecture principles — assume breach, verify explicitly, least privilege access — applied to Kubernetes workloads would have limited the scope of the privileged pod deployment that the malicious LiteLLM version attempted. Kubernetes Pod Security Standards, namespace isolation, and network policy enforcement are the specific controls that contain this class of attack. Organizations that had implemented Zero Trust Kubernetes configurations consistent with CSA guidance would have found that the payload's cluster-wide deployment attempt was blocked at the admission controller layer.

The STAR (Security Trust Assurance and Risk) program provides a mechanism for evaluating the security posture of third-party AI tool providers before adoption. The Trivy, KICS, and LiteLLM compromises each occurred in part because the projects' CI/CD security postures — specifically their GitHub Actions configurations and credential management practices — were not evaluated as part of enterprise procurement decisions. STAR assessments that include CI/CD security posture, artifact signing practices, and supply chain security controls could reasonably be expected to surface the kinds of vulnerabilities that TeamPCP exploited, enabling organizations to make more informed adoption decisions or to impose compensating controls.

CSA's published research note on the Mini Shai-Hulud campaign [12] provides immediate guidance for organizations responding to the active threat. The present whitepaper is intended to complement that tactical guidance with a strategic analysis of the underlying structural vulnerabilities and the framework-level responses they require.

Conclusions and Recommendations

The TeamPCP campaigns of 2026 should serve as a forcing function for enterprise security teams to reassess their AI tooling strategy through a concentration risk lens. This analysis suggests that the incidents are most productively understood as failures of the ecosystem's structural properties rather than primarily as failures of individual organizations' security practices — though both contributed and the distinction

matters. A primarily organizational diagnosis leads to guidance centered on faster patching and credential rotation, which is necessary but insufficient; an ecosystem diagnosis points toward the structural mitigations this paper advocates.

Several strategic conclusions emerge from this analysis. First, AI gateways, orchestration frameworks, and shared AI infrastructure components should be treated as critical infrastructure within enterprise security programs, subject to the same level of scrutiny as identity providers and key management systems. The operational efficiency they provide comes with proportional systemic risk, and that tradeoff should be made explicitly rather than implicitly.

Second, credential segmentation for AI infrastructure is not merely best practice — it is a structural necessity given the cascading attack patterns TeamPCP demonstrated. Every LLM provider API key, cloud credential, and CI/CD publishing token should be scoped and isolated such that the compromise of any single tool or pipeline does not yield the full credential footprint of the organization's AI deployment. Secrets management systems that enforce this isolation — HashiCorp Vault, AWS Secrets Manager, Azure Key Vault — should be used as the credential store for AI tooling rather than environment variables and configuration files.

Third, enterprise AI supply chain governance must extend to developer workstations and AI coding agent configurations. The Mini Shai-Hulud persistence mechanism established that AI coding agents are now part of the attack surface. Security programs that have not yet addressed AI coding agent configuration integrity, or that treat developer workstations as outside the supply chain threat model, have a material gap that this campaign explicitly exploited.

Fourth, the attack pattern used against OIDC trusted publishing — exploiting orphaned workflow configurations to obtain publish tokens without possessing long-lived credentials — highlights that security best practices can become attack vectors when their implementation is incomplete. Organizations and open-source projects using OIDC trusted publishing should audit their OIDC trust policies for orphaned configurations, restrict publish permissions to specific workflow paths and branches, and implement npm and PyPI provenance attestation verification in their dependency intake processes.

Fifth, enterprise security teams should incorporate the concept of the AI-native adversary into their threat modeling. TeamPCP's demonstrated capability to identify AI infrastructure leverage points, build AI-specific credential collection capabilities, and exploit AI agent configuration surfaces suggests that this threat archetype will persist and develop. Threat models that assume supply chain attackers are primarily interested in code execution on end-user systems may underweight the risk of AI credential theft and AI agent manipulation as primary attack objectives.

The enterprise AI adoption wave of the past several years has produced significant operational benefits, but it has also created a new layer of critical infrastructure — AI tooling — that was not designed with adversarial targeting in mind. TeamPCP's campaigns in 2026 demonstrated that sophisticated actors have now modeled this infrastructure and identified its leverage points. The response requires not just patching

individual vulnerabilities but addressing the structural concentration properties that make those vulnerabilities so consequential. Security programs that treat the AI tooling layer with the same rigor they apply to identity and key management infrastructure will be substantially better positioned than those that treat AI tools as ordinary software dependencies.

References

[1] Palo Alto Networks Unit 42. "Weaponizing the Protectors: TeamPCP's Multi-Stage Supply Chain Attack on Security Infrastructure." Unit 42 Threat Research, March 2026.

[2] Wiz Research. "Trivy Compromised: Everything You Need to Know about the Latest Supply Chain Attack." Wiz Blog, March 2026.

[3] The Hacker News. "Trivy Security Scanner GitHub Actions Breached, 75 Tags Hijacked to Steal CI/CD Secrets." The Hacker News, March 2026.

[4] Microsoft Security Response Center. "Guidance for detecting, investigating, and defending against the Trivy supply chain compromise." Microsoft Security Blog, March 24, 2026.

[5] Datadog Security Labs. "LiteLLM and Telnyx compromised on PyPI: Tracing the TeamPCP supply chain campaign." Datadog Security Labs, March 2026.

[6] LiteLLM. "Security Update: Suspected Supply Chain Incident." LiteLLM Documentation Blog, March 2026.

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

[8] The Hacker News. "Mini Shai-Hulud Worm Compromises TanStack, Mistral AI, Guardrails AI & More Packages." The Hacker News, May 2026.

[9] Wiz Research. "Mini Shai-Hulud Strikes Again: TanStack + more npm Packages Compromised." Wiz Blog, May 2026.

[10] JFrog Security Research. "Shai-Hulud: Here We Go Again – Worm by TeamPCP Hits NPM and PyPI." JFrog Security Research, May 2026.

[11] SecurityWeek. "TanStack, Mistral AI, UiPath Hit in Fresh Supply Chain Attack." SecurityWeek, May 2026.

[12] StepSecurity. "TeamPCP's Mini Shai-Hulud Is Back: A Self-Spreading Supply Chain Attack Compromises TanStack npm Packages." StepSecurity Blog, May 2026.

[13] CSA AI Safety Initiative. "TeamPCP: Cascading Supply Chain Attack on AI/ML Tooling." CSA Labs, March 30, 2026.

[14] Halcyon AI. "Trivy Supply Chain Compromise Enters Extortion Phase as Vect Ransomware Publishes First Victim." Halcyon AI Ransomware Alerts, March 2026. (Note: URL may require authentication to access.)

[15] The Hacker News. "TeamPCP Hacks Checkmarx GitHub Actions Using Stolen CI Credentials." The Hacker News, March 2026.

[16] RedRays. "SAP npm Packages Hijacked to Steal Cloud Credentials and Weaponize AI Coding Agents." RedRays Blog, April 2026.

[17] SafeDep. "Mass Supply Chain Attack Hits TanStack, Mistral AI npm and PyPI Packages." SafeDep Blog, May 2026.

[18] Cycode. "Shedding The Lite: Unfolding The Dramatic Turn of Events with the LiteLLM Compromise." Cycode Blog, March 2026.

[19] Kaspersky. "Trojanization of Trivy, Checkmarx, and LiteLLM solutions." Kaspersky Official Blog, March 2026.

[20] Ardan Labs. "LiteLLM Security Incident: AI Supply Chain & LLM Gateway Risk." Ardan Labs, March 2026.

[21] CyberScoop. "'Mini Shai-Hulud' malware compromises hundreds of open-source packages in sprawling supply-chain attack." CyberScoop, May 2026.

[22] Snyk. "How a Poisoned Security Scanner Became the Key to Backdooring LiteLLM." Snyk Blog, March 2026.

[23] Sysdig Threat Research. "TeamPCP expands: Supply chain compromise spreads from Trivy to Checkmarx GitHub Actions." Sysdig Blog, March 2026.

[24] Phoenix Security. "LiteLLM Backdoored by TeamPCP: PyPI Supply Chain Attack." Phoenix Security, March 2026.

[25] BleepingComputer. "TeamPCP hackers advertise Mistral AI code repos for sale." BleepingComputer, May 2026.

[26] Hackread. "TeamPCP Used Mini Shai-Hulud Worm to Poison Over 400 npm and PyPI Packages." Hackread, May 2026.

[27] SANS Internet Storm Center. "TeamPCP Weekly Analysis: 2026-W18 (2026-04-27 through 2026-05-03)." SANS ISC Diary, May 2026.

[28] Hackread. "TeamPCP Claims Sale of Mistral AI Repositories Amid Mini Shai-Hulud Attack." Hackread, May 2026.