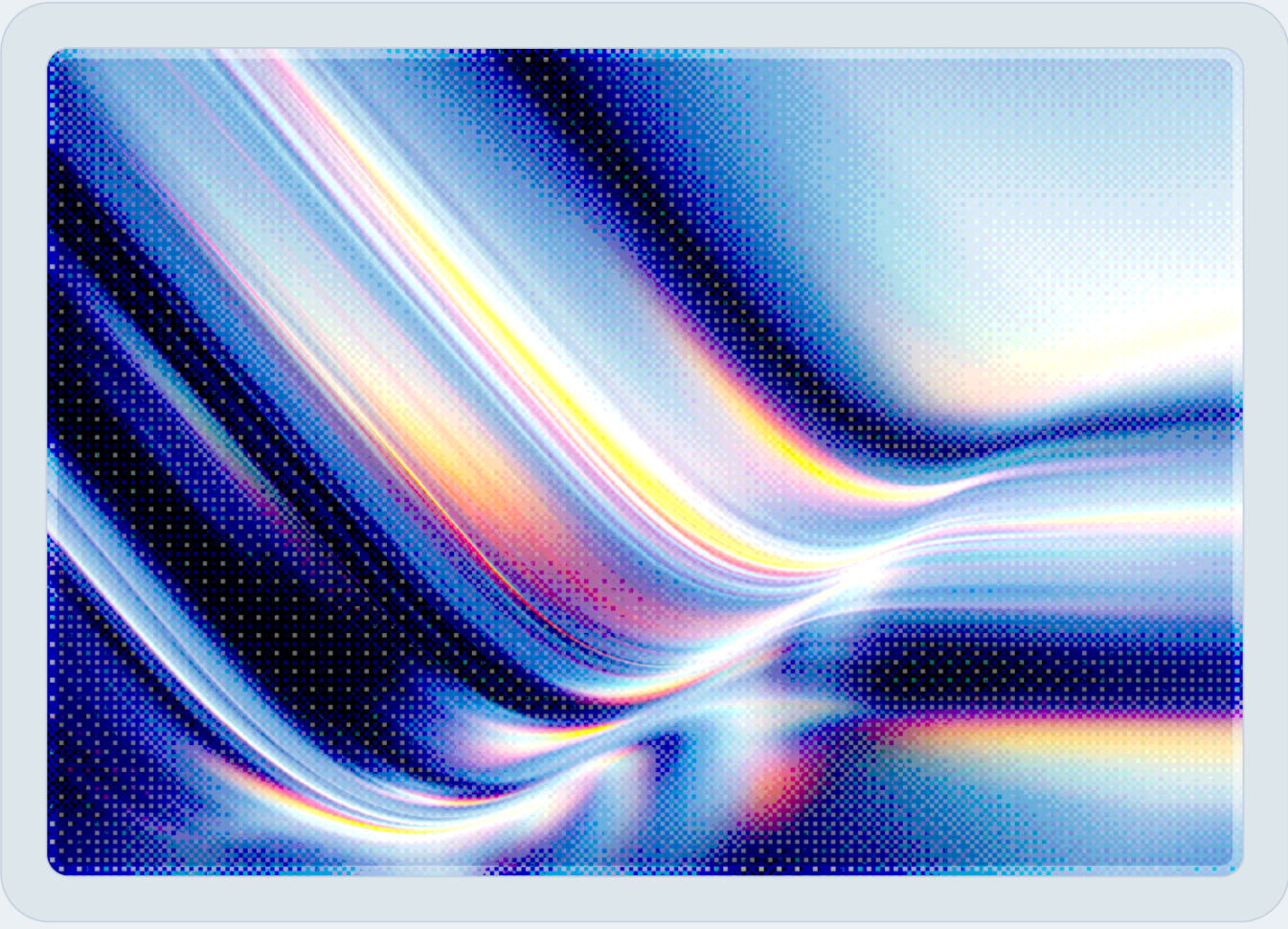
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: Supply Chain Worm Targets AI Developer Tooling

TeamPCP's Multi-Ecosystem Campaign Poisons npm, PyPI, and AI Coding Agent Configurations

2026-05-21

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

- The TeamPCP threat group has run a sustained, multi-ecosystem supply chain campaign since at least March 2026, poisoning over 400 package versions across 170+ npm and PyPI packages affecting an estimated 518 million cumulative downloads [1][2][4].
- AI developer tooling has been a deliberate target: LiteLLM (AI gateway), Mistral AI's SDK suite, and Guardrails AI (AI output validation) are among the confirmed compromises, meaning organizations running AI inference pipelines may have had credentials silently exfiltrated through trusted dependencies [3][4].
- The campaign introduced the first documented supply chain attack to weaponize AI coding agent configuration files—specifically Claude Code's `.claude/settings.json` and VS Code's `.vscode/tasks.json`—as persistence vectors that survive package removal [5].
- Mini Shai-Hulud forges valid SLSA Build Level 3 provenance attestations using stolen OIDC tokens, undermining a widely recommended artifact-integrity defense [6][7].
- TeamPCP publicly released the Shai-Hulud worm source code in mid-May 2026, lowering the barrier for copycat campaigns and making the threat actor's techniques broadly accessible [8].

Background

The software supply chain has emerged as one of the most consequential attack surfaces in modern development environments. Many developers install packages from public registries—npm for JavaScript and PyPI for Python—with limited per-release scrutiny, relying on registry curation, maintainer reputation, and automated tooling as implicit integrity controls. The threat actor known as TeamPCP has systematically exploited this trust across a campaign that, as of this writing, remains active.

TeamPCP's current operation, self-branded "Mini Shai-Hulud" after the giant sandworms of the *Dune* novels, began its documented activity in March 2026 with the compromise of LiteLLM, one of the most widely deployed open-source AI proxy and gateway libraries. LiteLLM version releases 1.82.7 through 1.82.8 on PyPI contained a multi-stage credential stealer that leveraged Python's `.pth` file mechanism to execute malicious code during interpreter initialization—before any application code ran [3][9]. This

choice of target was not incidental. LiteLLM sits at the intersection of developer environments and production AI inference pipelines, meaning a compromised version could silently harvest credentials from both local workstations and CI/CD systems deploying AI applications.

The campaign expanded steadily through April and into May 2026, adding the Telnyx communications SDK, SAP's Cloud Application Programming model packages, PyTorch Lightning, and multiple other ecosystems before culminating in the largest single wave on May 11, 2026, when TeamPCP published more than 400 malicious versions across 172 distinct packages within approximately five hours [1][10]. The sheer operational tempo—hundreds of malicious releases published faster than registries could respond—places this among the largest single-wave supply chain events documented in terms of package volume and publication speed [1][10].

What makes Mini Shai-Hulud strategically distinct from previous supply chain attacks is the deliberate prioritization of AI developer tooling as a target category. The packages compromised are not merely popular; several occupy chokepoints in the AI development lifecycle: the proxy that routes requests to language model APIs, the SDK for a major model provider's services, and the output validation library that enforces guardrails on model responses. Compromise at these layers gives an attacker not just developer credentials but potential visibility into AI application architectures, model API keys, and the inference traffic flowing through them.

Security Analysis

Attack Mechanism and Self-Propagation

The technical architecture of Mini Shai-Hulud centers on a three-stage exploit chain that converts legitimate CI/CD infrastructure into a publishing instrument. Researchers at StepSecurity and Wiz documented the mechanism in detail following the TanStack compromise [6][11]. The attack begins by identifying target repositories that use `pull_request_target` GitHub Actions workflows—a workflow trigger that, when misconfigured, grants fork-submitted pull requests write access to the base repository's cache. TeamPCP submitted a pull request that appeared benign but poisoned the repository's pnpm dependency cache with a malicious binary payload. When a legitimate maintainer merged an unrelated change approximately eight hours later, the standard release workflow pulled the poisoned cache and executed the attacker's code in an environment with full CI/CD publishing credentials.

The executed payload operates as a memory scraper, targeting the GitHub Actions `Runner.Worker` process directly. By reading runner process memory, it extracts OIDC tokens that are deliberately masked in workflow logs—a technique that bypasses the log-scrubbing protections GitHub applies to sensitive values [6][7]. With those OIDC tokens in hand, the malware exchanges them for Fulcio signing certificates and constructs in-toto attestation statements using the standard GitHub Actions build type, producing what external tooling correctly identifies as valid SLSA Build Level 3 provenance [7]. This is the first confirmed instance of npm malware shipping with forged but cryptographically valid SLSA attestations at the highest build level. Organizations that have adopted SLSA verification as a trusted supply chain defense would have received false confidence in the integrity of these packages.

Beyond CI/CD hijacking, TeamPCP also exploited compromised maintainer credentials for targets without accessible OIDC pipelines. The Datadog Security Labs analysis of the LiteLLM and Telnyx compromises traced a common infrastructure pattern linking these earlier intrusions to the same campaign [9].

AI Developer Tooling as a Priority Target

The pattern of AI-adjacent package selections is consistent with a deliberate focus on the AI development toolchain. LiteLLM routes API calls to commercial model providers including OpenAI, Anthropic, and others, making it a natural aggregation point for API keys across an organization's AI stack [3]. Mistral AI's SDK is the primary integration path for developers building on Mistral's hosted models [4]. Guardrails AI provides the output validation and policy enforcement layer for AI applications, sitting between model responses and production code [4]. Microsoft's DurableTask Python client, compromised via PyPI on May 19, is widely used in enterprise automation workflows that increasingly incorporate AI components [12].

The 2.3 MB obfuscated payload recovered from multiple compromised packages harvests credentials from over 100 distinct file paths spanning AWS IAM credentials, Azure and GCP service account keys, HashiCorp Vault tokens, Kubernetes configurations, npm publish tokens, 1Password vaults, and AI tool configuration files [1][2]. This breadth reflects an attacker with clear knowledge of how AI development environments are structured: developers working with AI tooling often hold credentials for cloud GPU instances, model API services, vector database deployments, and CI/CD pipelines simultaneously.

Persistence Through AI Coding Agent Configuration

A previously undocumented persistence technique in this campaign targets AI coding agent configuration files directly. The malware modifies two configuration files that AI-assisted coding tools read automatically on every session start: Anthropic's Claude Code reads

`.claude/settings.json`, and Visual Studio Code processes `.vscode/tasks.json` on folder open [5][13]. By appending `SessionStart` hooks to Claude Code's configuration and injecting task entries into VS Code's folder-open task runner, the malware ensures re-execution every time a developer opens a project—regardless of whether the infected package has since been removed or the dependency lockfile has been cleaned.

This is the first documented supply chain attack to treat an AI coding agent as a persistence target. Claude Code and VS Code—the specific targets of Mini Shai-Hulud's configuration-file persistence—operate with the same filesystem and network access as the developer running them; they can read repository contents, execute shell commands, and access any secrets present in the environment. AI coding assistants vary in their default permissions and sandboxing models, but any agent with ambient shell execution and file access creates a comparable persistence risk. An attacker with a persistent hook in the agent's configuration effectively maintains a foothold inside the developer's trusted execution context, potentially influencing subsequent AI-assisted code generation or exfiltrating the contents of files the agent processes.

Organizations whose developers run AI coding agents should treat this as a categorically new persistence surface that sits outside traditional endpoint detection models focused on process execution and file modification at standard OS paths.

Destructive Payload and Geofencing

The malware embeds a geofenced destructive branch. On systems whose locale settings suggest Iranian or Israeli origin, the payload includes a one-in-six probability of executing a recursive filesystem destruction command [4][10]. Country-aware logic separately avoids execution in Russian-language environments—a tactic commonly associated in threat intelligence with actors seeking to avoid domestic prosecution—consistent with threat actors operating from Russian-speaking regions. This destructive component has already resulted in confirmed credential exfiltration from at least two OpenAI employee devices that installed affected TanStack packages [14].

Source Code Publication and Escalation Risk

On or around May 15, 2026, TeamPCP published the Shai-Hulud worm source code publicly, framing it as an open-source release [8]. The publication transforms what was a novel, operationally complex campaign into reproducible capability. Security teams should anticipate that lower-capability actors will adapt the OIDC memory-scraping technique, the AI agent configuration persistence mechanism, and

the SLSA forgery approach for their own operations. The publication substantially lowers the barrier for adoption of these techniques; security teams should no longer rely on attacker sophistication as a meaningful deterrent for OIDC memory-scraping or AI agent persistence attacks.

Recommendations

Immediate Actions

Development and security teams should treat this as an active incident requiring immediate verification. Begin by auditing all CI/CD environments for evidence of Mini Shai-Hulud execution: look for unexpected processes reading runner memory, unusual npm or pip publish events in workflow logs, and modifications to `.claude/settings.json` or `.vscode/tasks.json` files in active repositories. Any developer workstation or CI runner that installed a package within the compromised namespaces—including any `@tanstack/*`, `litellm`, `mistralai`, `guardrails-ai`, `microsoft-durabletask-python`, or SAP CAP packages—between March 2026 and present should be treated as potentially compromised.

Credential rotation should be treated as a priority action for affected systems, completed before resuming normal operations. Rotate all secrets that were present on systems that executed compromised packages: GitHub tokens, AWS IAM credentials, API keys for model providers, Kubernetes service account tokens, and npm publish tokens. Pay particular attention to short-lived OIDC tokens whose exchange might not appear in standard credential audit logs. Organizations should verify that credential rotation includes any API keys for AI services, as these are among the most specifically targeted credential categories in the payload [1][9].

Inspect AI coding agent configuration files immediately. Review `.claude/settings.json` in all developer repositories and home directories for unexpected `SessionStart` entries. Review `.vscode/tasks.json` for unexpected `runOn: folderOpen` task definitions. Remove any entries not explicitly added by your team. This inspection should be automated at scale using file integrity monitoring or periodic configuration drift detection.

Short-Term Mitigations

The OIDC hijacking technique at the core of the TanStack-style attacks exploits a well-documented `pull_request_target` misconfiguration. Audit all GitHub Actions workflows for this pattern and restrict cache write access to workflows triggered by trusted events only. Implement `CACHEKEY` namespacing or cache isolation between fork and base repository workflows to prevent cross-fork poisoning. StepSecurity's Harden-Runner GitHub Action provides automated detection of unexpected outbound network calls from runners and can surface anomalous memory access attempts [6].

Treat SLSA provenance attestations as a necessary but no longer sufficient control. Mini Shai-Hulud demonstrated that provenance can be forged using legitimate infrastructure credentials; attestations verify the build system's identity, not the integrity of the build inputs. Complement provenance verification with dependency pinning to exact content hashes in lockfiles, behavioral analysis of postinstall scripts before execution, and review of new releases from dependencies before they enter production pipelines.

For AI pipeline dependencies specifically, evaluate whether LiteLLM, Mistral AI SDKs, Guardrails AI, or DurableTask packages installed during the exposure windows have been rebuilt from verified clean source. Use Sonatype, Datadog, or equivalent software composition analysis tooling to cross-reference installed package versions against the known-malicious version list, which is maintained in the SANS ISC TeamPCP tracking diary [15].

Strategic Considerations

The Mini Shai-Hulud campaign illustrates a structural tension in modern AI development: the same features that make AI coding agents productive—deep file access, automatic configuration loading, ambient shell execution—also make them attractive persistence targets. Organizations integrating AI coding agents into developer workflows should define explicit trust boundaries for those agents, treat their configuration files with the same integrity controls applied to CI/CD pipeline definitions, and include AI agent configuration directories in endpoint detection scope.

Supply chain security programs should be revised to account for multi-ecosystem attacks. The three-ecosystem chain observed in the April–May wave (npm to PyPI to Packagist within 36 hours) [15] means that a compromise detected in one registry may already have propagated to others before response actions are complete. Continuous monitoring across all package ecosystems used by an organization is a prerequisite for timely detection.

The public release of the worm source code warrants proactive preparation. Expect future campaigns using similar OIDC memory-scraping and AI agent persistence techniques from actors with fewer resources than TeamPCP. Incorporating these techniques into adversary emulation exercises now, before they appear in commodity toolkits, allows security teams to validate detection capability while the techniques are still relatively novel.

CSA Resource Alignment

The Mini Shai-Hulud campaign maps to several threat categories and control domains addressed across CSA's AI security frameworks.

The MAESTRO threat modeling framework for agentic AI directly addresses the risk surface exposed by AI coding agent configuration persistence. MAESTRO's Layer 3 (Agent Trust and Identity) covers the mechanisms by which AI agents establish trusted execution contexts; Mini Shai-Hulud's exploitation of `SessionStart` hooks illustrates a concrete pathway to violating those trust boundaries through dependency-layer compromise. Layer 4 (Infrastructure and Pipeline Security) encompasses the CI/CD integrity concerns raised by the OIDC hijacking and cache poisoning techniques. Security teams using MAESTRO to model their AI development environments should add supply chain compromise of agent configuration files as an explicit threat scenario.

The AI Controls Matrix (AICM) v1.0 supply chain security domain covers controls relevant to both the AI package dependencies (model SDKs, inference proxies) and the build pipeline integrity required to ensure that AI components are not tampered with between development and deployment. The LiteLLM and Guardrails AI compromises demonstrate that AI-specific libraries are supply chain risks that merit the same scrutiny applied to core infrastructure dependencies, and the AICM's shared responsibility model should be applied to model provider SDKs as well as to model artifacts themselves.

CSA's Zero Trust guidance is directly applicable to the credential theft aspects of this campaign. The principle that no artifact, token, or attestation should be trusted solely on the basis of its apparent provenance applies with particular force here: Mini Shai-Hulud demonstrated that SLSA Build Level 3 attestations can be forged using legitimate CI credentials. A zero trust approach to artifact verification requires behavioral and contextual controls layered beneath cryptographic attestation, not in place of it.

The CSA STAR program's continuous monitoring requirements provide an audit framework organizations can apply to verify that their software supply chains meet integrity baselines. Organizations seeking to document their supply chain security posture in the context of AI development should consider incorporating TeamPCP's techniques into their STAR-level control assessments.

References

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

[2] Orca Security. "TanStack and 160+ npm/PyPI Packages Compromised in Supply Chain Worm Attack." Orca Security Blog, May 2026.

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

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

[5] MSN / Morning Overview. "The 'mini Shai-Hulud' attack hides inside AI coding agent configs – the first supply chain attack to weaponize Claude Code and VS Code as persistence vectors." MSN, May 2026.

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

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

[8] SecurityWeek. "TeamPCP Ups the Game, Releases Shai-Hulud Worm's Source Code." SecurityWeek, May 2026.

[9] Sonatype. "Compromised litellm PyPI Package Delivers Multi-Stage Credential Stealer." Sonatype Blog, 2026.

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

[11] ReversingLabs. "Team PCP's Mini Shai-Hulud tears at open-source trust." ReversingLabs Blog, May 2026.

[12] Wiz. "durabletask: TeamPCP's Latest PyPI Compromise." Wiz Blog, May 2026.

[13] Phoenix Security. "Mini Shai-Hulud: SAP CAP and mbt npm Packages Backdoored via Bun-Loaded Credential Stealer with Claude Code Persistence." Phoenix Security, 2026.

© 2026 Cloud Security Alliance. AI-assisted Rapid Research.

[14] The Hacker News. "TanStack Supply Chain Attack Hits Two OpenAI Employee Devices, Forces macOS Updates." The Hacker News, May 2026.

[15] SANS Internet Storm Center. "TeamPCP Supply Chain Campaign: Activity Through 2026-05-17." SANS ISC, May 2026.