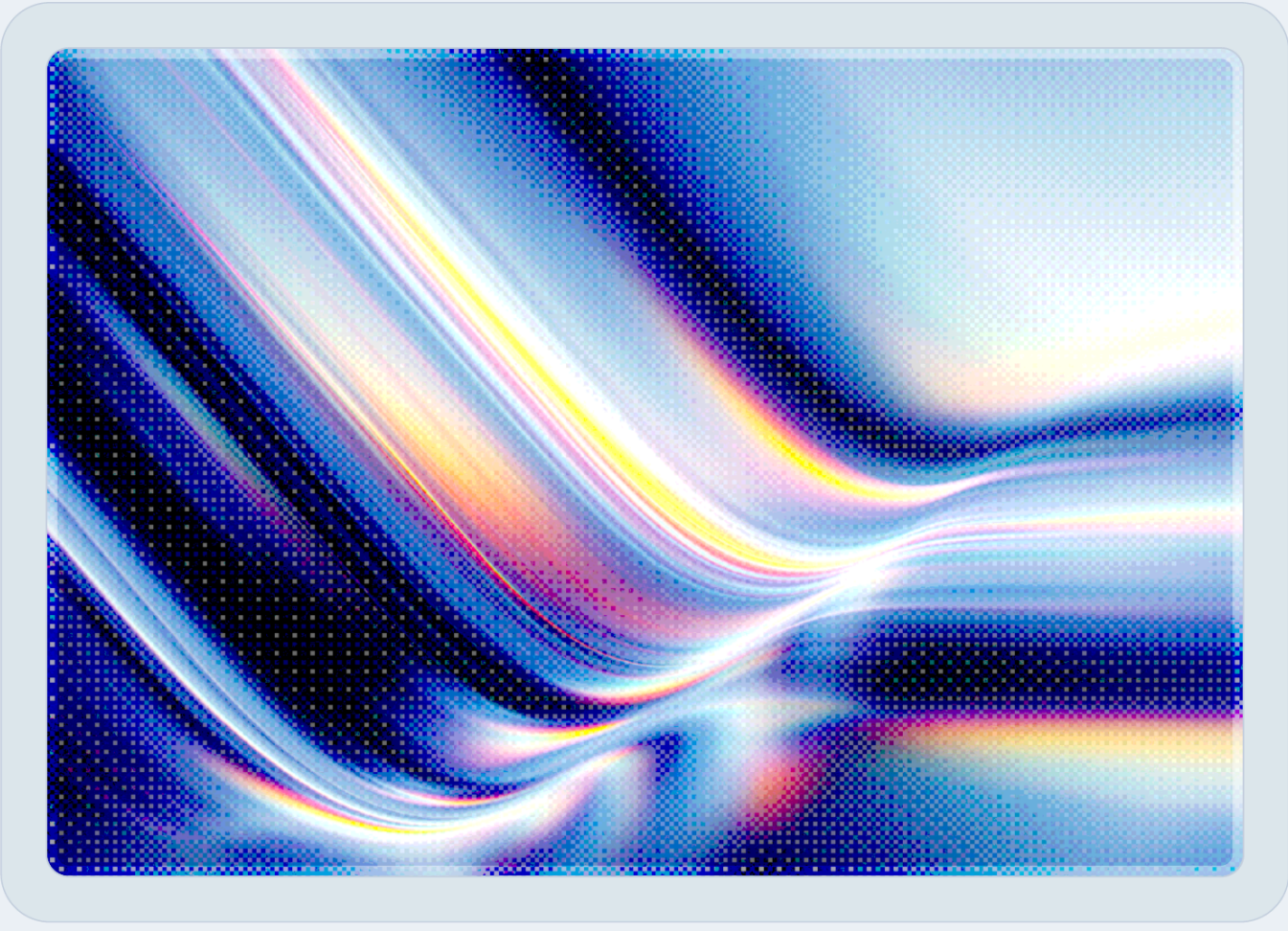
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: AI Dev Infrastructure as Supply Chain Target

TeamPCP's Worm Campaign and the Escalating Risk to AI Development Pipelines

2026-05-18

AI-assisted Rapid Research



CSAI

cloud security alliance®

Key Takeaways

- The Mini Shai-Hulud campaign, attributed to threat actor TeamPCP, compromised over 170 npm and PyPI packages with 518 million cumulative downloads between May 11–12, 2026 [1].
- TeamPCP targeted AI development tooling—including the Mistral AI Python SDK, Guardrails AI, and LiteLLM—suggesting the group prioritized packages that aggregate LLM provider credentials and occupy privileged positions in AI training and inference pipelines [1][2].
- The campaign introduced the first documented npm worm capable of producing cryptographically attested malicious packages with valid SLSA Build Level 3 provenance, defeating a key supply chain security control [1].
- OpenAI confirmed that two employee devices were compromised and limited credential material was exfiltrated from internal repositories, including code-signing certificates [4].
- TeamPCP subsequently open-sourced its attack toolkit, lowering the technical barrier for copycat campaigns against AI development infrastructure [6].

Background

Supply chain attacks against open-source ecosystems are not new, but the May 2026 Mini Shai-Hulud campaign escalated the threat on at least two fronts: the degree of automation enabling worm-like self-propagation, and the deliberate selection of AI development tooling as primary attack infrastructure. TeamPCP—a financially motivated extortion group that has operated since at least late 2025—has demonstrated a consistent pattern of targeting high-leverage packages that sit at trust boundaries in modern software delivery. Their previous campaign, documented by CSA labs in April 2026, compromised Trivy, Checkmarx KICS, LiteLLM, and the Telnyx Python SDK in a cascade attack that leveraged stolen credentials to traverse organizational boundaries [7].

The name "Shai-Hulud" references the sandworms of Frank Herbert's *Dune* universe—creatures that consume everything in their path as they move unseen beneath the surface. The naming convention appears intentional: the original Shai-Hulud toolkit was an internal offensive framework, and TeamPCP released a public version on GitHub on May 12, 2026, immediately following the campaign [6]. That decision—open-sourcing a successfully deployed attack toolkit after deployment—may signal a move toward commoditizing supply chain attack techniques for a broader threat actor audience.

The AI development ecosystem creates attack conditions that differ from general software supply chains in important ways. A growing class of AI-focused packages—including API gateways like LiteLLM—bundle or broker access to multiple LLM provider credentials simultaneously. LiteLLM, compromised in March 2026, served as an API gateway aggregating keys for over 100 LLM providers across potentially 95–97 million monthly download instances [7][8]. Packages like Guardrails AI and the Mistral AI SDK are deeply embedded in training and fine-tuning workflows, where CI/CD pipelines routinely have access to cloud credentials for GPU compute, model storage, and inference infrastructure. The value of a single compromised developer token in an AI company's pipeline can significantly exceed what equivalent access yields in a traditional software organization, given this aggregated credential scope.

Security Analysis

The Attack Sequence

The May 11 campaign began with a compromise of the TanStack router ecosystem's publishing infrastructure. Attackers exploited a GitHub Actions workflow configured with the `pull_request_target` trigger—a setting that executes workflow code in the context of the target repository's permissions even when triggered by a pull request from a fork. By staging a malicious payload in an orphaned commit on a repository fork, TeamPCP caused the legitimate TanStack CI pipeline to execute attacker-controlled code. That code extracted an OIDC token from the GitHub Actions runner process memory, bypassing GitHub's secret-masking layer, then exchanged the OIDC token for a per-package npm publish token scoped to the entire TanStack namespace [1][3].

From that foothold, the worm became self-sustaining. Each infected CI/CD run became a new publisher: the malware enumerated packages associated with the same maintainers, identified npm tokens with `bypass_2fa` privileges, and published trojanized versions across 42 `@tanstack/*` packages spanning 84 versions. Within five hours of initial compromise, TeamPCP had published over 400 malicious package versions across 172 distinct packages [5]. The propagation path from a single misconfigured GitHub Actions workflow to ecosystem-wide compromise required no direct human attacker involvement after the initial payload delivery.

The provenance bypass deserves particular attention. Modern supply chain security guidance strongly recommends verifying SLSA (Supply-chain Levels for Software Artifacts) provenance attestations before trusting package installations. The Mini Shai-Hulud worm defeated this control: because it operated within the authentic TanStack CI pipeline, the attested malicious packages carried valid SLSA

Build Level 3 signatures. Downstream systems and CI runners that verified provenance still received and executed malicious code [1][3]. The implication is that cryptographic provenance verification, while necessary, is insufficient when the pipeline generating that provenance has itself been compromised.

AI Development Infrastructure as a Target

The campaign's most significant dimension for AI-focused organizations is the deliberate selection of AI development tooling as propagation targets and credential sources. The PyPI packages compromised included `guardrails-ai@0.10.1` and `mistralai@2.4.6`—both tools embedded in production AI development workflows [1][2]. Guardrails AI sits between LLM calls and application logic in safety-conscious architectures, giving it runtime visibility into both model outputs and the cloud credentials funding inference. The Mistral AI SDK provides authenticated access to commercial model APIs from within developer and CI environments.

Mistral AI confirmed on May 12, 2026, that attackers had "temporarily compromised one of the company's codebase management systems through a third-party software supply chain attack," resulting in contaminated SDK packages being distributed during the exposure window [5]. TeamPCP subsequently claimed on a hacking forum to possess approximately 5GB of Mistral internal repositories—covering training systems, fine-tuning projects, benchmarking tools, and inference infrastructure—listed for sale at $25,000. Mistral stated that hosted services, managed user data, and research environments were not affected, and that only "certain non-core code repositories" were accessed [5]. Independent verification of the broader repository claim has not been established, but the incident illustrates how a supply chain foothold in developer tooling can become a pivot to intellectual property theft.

OpenAI's exposure followed a different path: two employee devices downloaded malicious TanStack packages before updated protections were deployed. The malware exfiltrated "limited credential material" from a small number of internal repositories accessible to the affected employees, including code-signing certificates used to validate OpenAI software releases [4]. OpenAI confirmed no customer data, production systems, or model weights were compromised, and began rotating affected credentials and certificates immediately. The company also noted that macOS users of ChatGPT Desktop, Codex App, Codex CLI, and Atlas must update before June 12, 2026, when older certificates are revoked [4].

The Shai-Hulud toolkit demonstrated targeted design for AI development environments, extending beyond generic credential theft. The toolkit installs persistence hooks into Claude Code and VS Code IDE configurations, monitors GitHub token validity via a daemon named `gh-token-monitor`, and embedded attacker commits attributed to `claude@users.noreply.github.com`, likely

designed to blend malicious activity into the noise of AI-assisted development workflows [1][6]. This represents a meaningful operational security evolution: the attackers did not merely target AI infrastructure but actively camouflaged their activity within it.

Malware Capabilities and Exfiltration Infrastructure

The Shai-Hulud toolkit, now publicly available, is a modular TypeScript/Bun offensive framework targeting over 100 sensitive file paths [6]. The JavaScript variant steals npm tokens, GitHub credentials, AWS access keys, Kubernetes JWTs, SSH keys, and CI environment secrets. The Python variant extends coverage to Azure Key Vault, GCP Secret Manager, HashiCorp Vault, password managers (1Password, Bitwarden), and Docker credentials. Exfiltration uses AES-256-GCM encryption with RSA-4096 public key wrapping, with data sent to attacker-controlled endpoints including `git-tanstack[.]com`, or staged in GitHub repositories as encrypted dead drops using Dune-themed naming conventions [6].

The dead-man's switch is an important detail for incident responders. The malware monitors stolen GitHub tokens by polling `api.github.com/user` every 60 seconds. If a token is revoked, the persistence daemon executes `rm -rf ~/` on the infected system. Commit messages on compromised repositories carry the warning: *"IfYouRevokeThisTokenItWillWipeTheComputerOfTheOwner"* [6]. This coercive mechanism has significant implications for credential rotation: organizations should isolate and image affected systems before revoking tokens when forensic preservation is a priority.

The following table summarizes the primary attack vectors and their security implications:

Attack Vector	Mechanism	Security Control Defeated
GitHub Actions OIDC abuse	Runner memory extraction of OIDC token	Secret masking layer bypassed at process level
`pull_request_target` exploitation	Orphaned commit triggers privileged workflow	Workflow permission isolation
SLSA provenance forgery	Attested packages built by compromised pipeline	Package provenance verification
IDE hook persistence	Claude Code and VS Code config injection	Endpoint detection (low-signature daemon)

Attack Vector	Mechanism	Security Control Defeated
Dead-man's switch	Token revocation triggers `rm -rf ~/`	Standard credential rotation procedures

Threat Actor Profile and Campaign Continuity

TeamPCP has operated as a financially motivated extortion group, with the Supply Chain Cascade and Mini Shai-Hulud campaigns representing a sustained investment in supply chain attack capability. The April 2026 CSA research note on the Cascade campaign documented TeamPCP's use of blockchain command-and-control infrastructure via Internet Computer Protocol canisters, steganographic payload delivery embedded in WAV audio files, and persistence mechanisms using Python `.pth` files that survive package uninstallation [7]. The May 2026 campaign suggests deliberate evolution rather than improvisation: the OIDC token abuse and worm propagation techniques appear in 19 of 22 documented TeamPCP tactics implemented in the publicly released toolkit [6].

The decision to open-source the toolkit after deployment warrants attention. Analyst assessments suggest possible motivations include complicating future attribution as other actors adopt the same techniques, building reputation in criminal marketplaces, and pressuring victim organizations to settle quickly before defenders adapt. These assessments are inferential, as no post-deployment communications from TeamPCP have been attributed confirming these motivations. The release also means organizations should anticipate Mini Shai-Hulud-style attacks from less sophisticated actors in the near term, since the primary technical barriers to replicating the campaign have been eliminated.

Recommendations

Immediate Actions

Organizations using any of the 170+ compromised packages during May 11–12, 2026, should treat the remediation as a potential active breach. The first priority is identifying whether affected package versions were installed in development environments, build pipelines, or production systems during the exposure window. Any environment that executed a compromised version should be assumed to have had

credentials extracted, and full credential rotation should begin with cloud provider access keys, GitHub personal access tokens, npm and PyPI publish tokens, Kubernetes service account tokens, SSH keys, and any LLM provider API keys that were accessible in the environment.

Because the malware installs a monitoring daemon that triggers destructive actions upon token revocation, affected systems should be isolated—disconnected from network access and credential stores—before credentials are rotated. Forensic imaging of affected developer workstations and CI runners should occur prior to remediation if investigation is required. Security teams should search all developer environments and Python site-packages directories for the `gh-token-monitor` daemon and any `.pth` files introduced during the exposure window, as these may survive package uninstallation [7].

Organizations relying on the OpenAI ChatGPT Desktop, Codex App, Codex CLI, or Atlas applications on macOS should update before June 12, 2026, when the compromised code-signing certificates used in the affected OpenAI repositories will be revoked [4].

Short-Term Mitigations

A primary enabling vulnerability was the combination of GitHub Actions workflow configurations using `pull_request_target` without branch restrictions, and OIDC trusted publisher settings that lacked specific branch or workflow scope. Organizations should audit all GitHub Actions workflows using `pull_request_target` triggers and evaluate whether each workflow requires the elevated permissions this trigger provides. Where the trigger is necessary, workflows should be restricted to specific branches and workflow file paths in the OIDC trusted publisher configuration to prevent orphaned-commit abuse [1][3].

Package provenance verification remains a meaningful control even after this campaign demonstrated its limits. SLSA provenance attestation failed in this case because the issuing pipeline was itself compromised—but provenance verification would still catch the majority of simpler supply chain attacks that do not involve pipeline compromise. Organizations should combine provenance checks with hash-pinned dependency versions, using lockfiles or `pip install --require-hashes` to ensure that a substituted package version, even one with valid provenance, does not silently execute [7]. Registry-based controls like Artifactory's remote repository caching or similar proxying can prevent direct registry pulls without a review step.

CI/CD credential architecture should apply least-privilege principles aggressively. AI development pipelines in particular tend to accumulate broad cloud permissions over time as new model training, fine-tuning, and evaluation tasks are added. Security scanning tools, build systems, and test runners should not

have ambient access to LLM API keys, model storage credentials, or cloud compute credentials. These should be injected at runtime using short-lived secrets from a secrets management system, scoped narrowly to the specific pipeline task requiring them.

Strategic Considerations

The targeting pattern across TeamPCP's two 2026 campaigns suggests that developer and infrastructure credentials—not model weights or training data directly—may represent the highest-value targets in AI development organizations, because they provide access to those assets. The LiteLLM compromise in March gave TeamPCP potential access to API keys for over 100 LLM providers simultaneously. The Mistral AI SDK and Guardrails AI compromises targeted the tooling closest to model inference and safety enforcement in production pipelines. Organizations building AI products should map their credential exposure surface with specific attention to packages that aggregate or broker access to LLM providers, cloud GPU compute, model storage, and fine-tuning infrastructure.

The open-sourcing of the Shai-Hulud toolkit substantially lowers barriers and suggests supply chain attacks against AI development infrastructure are likely to grow in frequency. The technical capability to execute a Mini Shai-Hulud-style attack—GitHub Actions OIDC abuse, npm worm propagation, and attested package delivery—is now accessible to a broader set of threat actors. AI development organizations that have not yet conducted a formal software supply chain security assessment should treat this as a near-term priority given the documented threat landscape.

CSA Resource Alignment

Mapping to MAESTRO's layer model, the Mini Shai-Hulud campaign appears to involve multiple layers: the compromise of AI development tooling (Layer 7, Agent Ecosystem) enabled lateral movement into internal model infrastructure (Layer 2, Data Operations), with the Shai-Hulud toolkit's IDE hooks extending attacker persistence into the agent execution environment itself (Layer 3, Agent Frameworks) [9]. MAESTRO's explicit treatment of "marketplace manipulation" and "tool misuse" at the ecosystem layer directly describes the package registry targeting observed in this campaign.

CSA's AI Controls Matrix (AICM) provides specific control domains applicable to this threat. Supply chain security controls under the Model Provider and Orchestrated Service Provider implementation guidelines address dependency management, CI/CD pipeline integrity, and publishing workflow security. The requirement to validate software provenance and restrict publishing credentials aligns directly with

the OIDC misconfiguration that enabled worm propagation. Organizations using AICM as a compliance framework should review their control implementations against the specific attack vectors documented here, particularly around trusted publisher configuration and build pipeline isolation.

CSA's Software Transparency guidance on securing the digital supply chain—including guidance on SBOMs and CI/CD pipeline security—addresses the structural conditions TeamPCP exploited. Generating and maintaining SBOMs for AI development projects would not have prevented this attack, but it accelerates incident response by enabling rapid identification of whether compromised package versions are present in an environment. Combined with hash-pinned dependencies, SBOM practices reduce the exposure window between a supply chain compromise and detection.

Prior CSA AI Safety Initiative research on the TeamPCP Supply Chain Cascade (April 2026) provides foundational context for this campaign and documents remediation procedures for the March 2026 wave that remain partially relevant, particularly the guidance on manual removal of `.pth` persistence files and SHA-pinned GitHub Actions references [7].

References

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

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

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

[4] Cyberinsider. "OpenAI confirms exposure in recent 'Shai-Hulud' supply-chain attack." Cyberinsider, May 2026.

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

[6] Datadog Security Labs. "Shai-Hulud Goes Open Source." Datadog Security Labs, May 2026.

[7] CSA AI Safety Initiative. "TeamPCP Supply Chain Cascade: When Security Tools Become Attack Infrastructure." CSA Labs, April 2026.

[8] CSA AI Safety Initiative. "TeamPCP and the Cascading AI/ML Supply Chain Campaign." CSA Labs, March 2026.

[9] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

