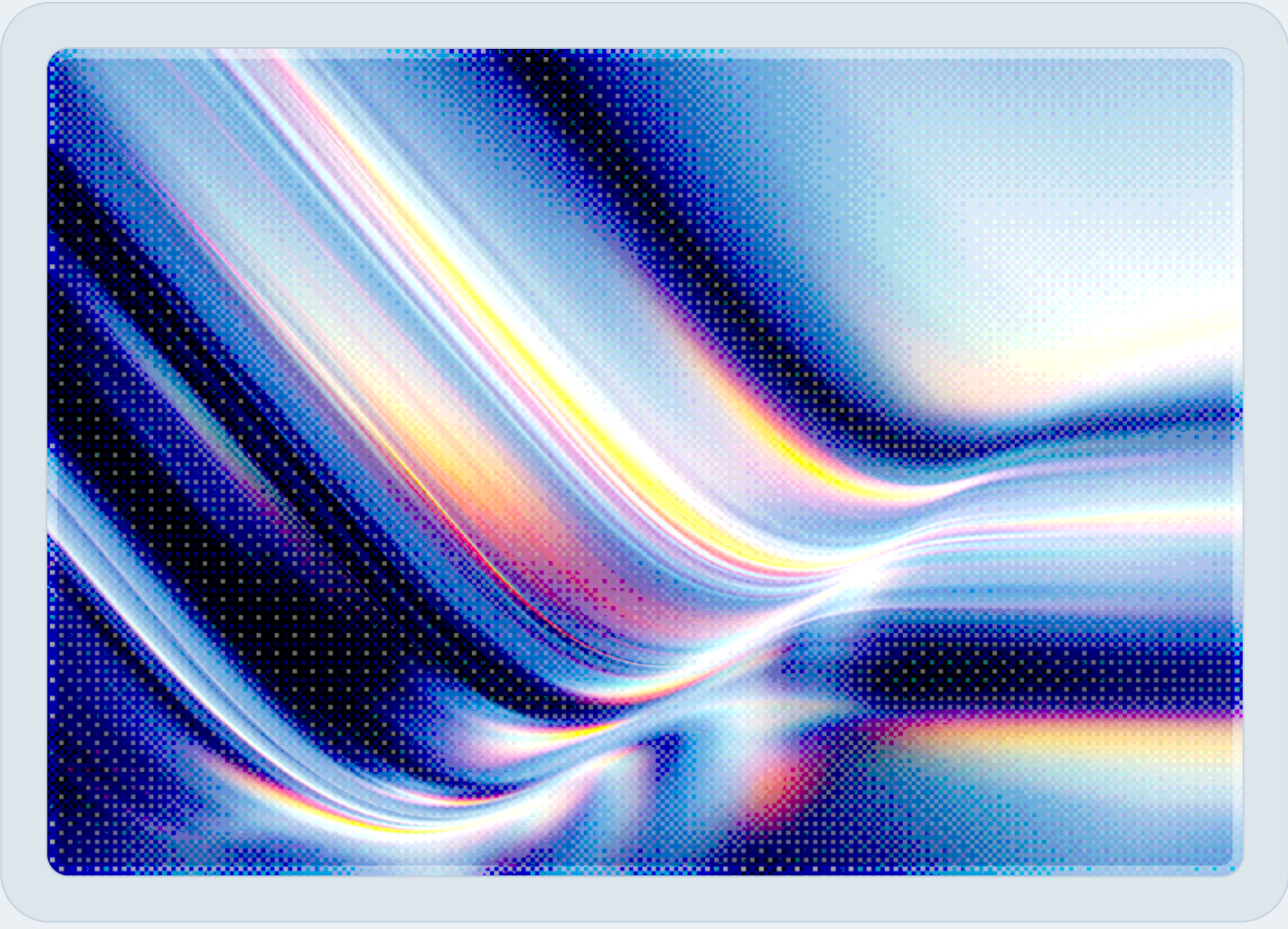
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: Signed npm Worm Targets AI Developer Supply Chain

How TeamPCP Defeated Provenance Attestation to Compromise Mistral AI, TanStack, and 170+ Packages

2026-05-14

AI-assisted Rapid Research



CSAI

cloud security alliance®

Key Takeaways

- Between May 10 and May 12, 2026, a threat actor group known as TeamPCP deployed a self-propagating worm called Mini Shai-Hulud that compromised more than 170 npm and PyPI packages — including the official Mistral AI SDK — publishing over 400 malicious versions with a cumulative registry download count exceeding 518 million — a figure that includes automated CI/CD fetches and mirrors, not solely end-user installations [1][2].
- The worm exploited GitHub Actions OIDC federation to mint legitimate npm publish tokens without ever stealing a static secret, then invoked the Sigstore stack to generate valid SLSA Build Level 3 provenance attestations for malicious packages. This is the first documented npm worm to produce cryptographically valid supply chain provenance for malicious releases, directly undermining a class of defenses organizations adopted specifically to counter supply chain attacks [3][4].
- AI developer ecosystems have emerged as high-value targets: the inclusion of the Mistral AI SDK and Guardrails AI framework among the primary victims suggests TeamPCP is deliberately expanding its scope beyond commodity infrastructure tooling to encompass the AI application layer [5][6].
- Standard lockfile pinning provides conditional protection only: organizations that ran fresh installs or dependency updates during the attack window may have captured the malicious versions as their pinned baseline [7].
- Immediate priorities for affected organizations are: audit lockfiles and CI artifacts for known malicious versions, rotate all credentials that may have transited CI/CD environments running affected packages, and scan developer workstations for the `gh-token-monitor` persistence daemon before revoking tokens to avoid triggering a destructive wiper [8].

Background

The Mini Shai-Hulud campaign is the latest evolution of sustained supply chain operations attributed to TeamPCP, a cloud-focused threat actor that first emerged in late 2025. The group gained significant attention in December 2025 with the React2Shell campaign [9], which targeted cloud environments through compromised JavaScript build tooling. By March 2026, TeamPCP had shifted its primary operational model toward what researchers describe as "smash and grab" supply chain compromise — highly automated attacks designed to compromise as many packages as possible in a narrow time window, maximizing credential yield before defenders respond [9].

The name "Mini Shai-Hulud" references the giant sandworm from Frank Herbert's *Dune*, a recurring theme in TeamPCP's operational branding. Researchers at StepSecurity and Wiz have documented the group's use of Dune-themed GitHub repositories — created using credentials stolen during campaign execution — as one of three primary exfiltration channels for the Mini Shai-Hulud campaign, alongside the domain `git-tanstack[.]com` and the Session encrypted messaging network [3][4]. The consistent Dune-themed branding across infrastructure — repositories, domain names, tooling — suggests a group with deliberate operational tradecraft rather than an opportunistic script collective.

The May 2026 campaign targeted the npm and PyPI registries simultaneously, a first for TeamPCP's documented operations. The primary targets included the TanStack router ecosystem (42 packages), Mistral AI's SDK suite on both registries, UiPath's automation tooling (65 packages), the OpenSearch JavaScript client (which alone carries approximately 1.3 million weekly npm downloads), and the Guardrails AI framework on PyPI [1][2][11]. The breadth of targets is notable for spanning both commodity infrastructure and AI-specific developer tooling, suggesting that TeamPCP has identified AI SDK adoption as a credentialing surface worth prioritizing.

Security Analysis

The OIDC Token Extraction Chain

The attack's technical sophistication centers on a multi-step exploitation chain that operates entirely within the trust model of modern CI/CD infrastructure. Standard npm publishing workflows have, in recent years, migrated toward passwordless authentication using GitHub Actions OIDC federation: rather than storing a long-lived npm token as a repository secret, a workflow requests a short-lived OIDC token from GitHub's identity service and exchanges it at npm's federation endpoint for a scoped publish credential.

This architecture was intended to reduce static secret exposure. Mini Shai-Hulud exploited misconfigured `pull_request_target` workflows to extract OIDC tokens from otherwise-intended-to-be-secure publishing pipelines.

TeamPCP's initial access relied on a chained GitHub Actions exploit involving the `pull_request_target` trigger, which runs with elevated permissions even when triggered by pull requests from forks. By combining this trigger with GitHub Actions cache poisoning, attackers caused a victim repository's workflow to execute attacker-controlled code during the test or cleanup phase — after code checkout but within the same authenticated runner context. The payload then scanned `/proc` on the runner host, located the `Runner.Worker` process, read its memory directly, and extracted masked secret structures including the runner's pending OIDC token. The extracted OIDC token was exchanged with npm's federation endpoint for a valid publish credential, all without touching any stored repository secret [3][10].

This extraction technique is particularly difficult to detect because it generates no anomalous API calls to GitHub's secrets API, produces no failed authentication events, and leaves no artifact in standard workflow logs. The OIDC token extraction happens at the process memory level, below the visibility of most CI/CD audit tooling.

Self-Propagating Worm Mechanics

What distinguishes Mini Shai-Hulud from a conventional supply chain compromise is its worm behavior. Once the malware obtained a valid npm publish credential for one maintainer's account, it enumerated all packages that maintainer had publish rights over, injected its payload into each package's archive, bumped the version number, and published the infected versions. This propagation happened autonomously, without further attacker interaction, compressing what might take a human operator hours into an automated loop that published 84 malicious versions of TanStack packages in under six minutes during the initial wave [3][7]. The credential harvest from one compromised package directly funded the compromise of additional packages, creating a self-reinforcing spread mechanism across the ecosystem.

TeamPCP extended this propagation logic across the PyPI registry as well. For the Mistral AI Python SDK, malicious code was inserted into `mistralai/client/__init__.py` such that it executed silently on package import, spawning a background process to harvest credentials from common locations on Linux systems [5]. The payload checked for Linux environment before executing — a design choice that limits execution to Linux hosts, the standard runner OS for cloud-native CI environments, while avoiding execution on developer machines where macOS and Windows are prevalent.

The Provenance Attestation Problem

The campaign's most consequential technical innovation is its subversion of SLSA Build Level 3 provenance attestation. Provenance attestation, powered by Sigstore's Rekor transparency log and Cosign signing toolchain, has been advocated as a high-assurance mechanism for verifying that a published package artifact was built from a specific source commit in a known CI environment. Because the attacker's payload ran inside the legitimate CI/CD workflow using a legitimately obtained OIDC token and then published through the project's own trusted-publisher binding, the resulting packages received genuine, cryptographically valid provenance attestations from Sigstore. The malicious packages were indistinguishable from legitimate ones at the attestation layer [3][4].

This is better characterized as a scope limitation than an implementation flaw in Sigstore: the attestation system performed exactly as designed, answering a narrower question than security teams may have assumed it answered. Provenance attestation answers the question "was this artifact produced by the expected workflow?" — and in this case, the answer was yes. The attestation system had no visibility into the fact that attacker-controlled code was executing within that workflow. Security teams that have treated provenance attestation as a sufficient terminal signal for package trustworthiness must revisit that assumption: attestation verifies build provenance, not the absence of adversarial code in the build pipeline.

Credential Harvest and Exfiltration

The worm's payload is a modular credential stealer that harvests credentials from more than 100 hardcoded paths spanning cloud providers (AWS, GCP, Azure, Kubernetes), CI/CD systems (GitHub tokens, npm tokens, Docker credentials), and developer tooling including password managers (1Password, Bitwarden) and cryptocurrency wallets [1][2]. The expansion to target password managers and crypto wallets marks an escalation beyond infrastructure credential theft into personal account compromise, consistent with TeamPCP's reported shift toward data extortion and ransom-adjacent monetization.

Stolen credentials were exfiltrated via three channels: the attacker-controlled `git-tanstack[.]com` domain, the Session encrypted messaging network through `*.getsession.org`, and Dune-themed GitHub repositories created using tokens stolen during the campaign itself [3][4]. Security researchers estimate the actor may have exfiltrated upward of 300 GB of data and 500,000 credentials, though these figures — reported in connection with TeamPCP's broader operational history — remain preliminary and may encompass activity beyond the May 2026 Mini Shai-Hulud window specifically [9].

The npm-specific payload included a persistence mechanism — a daemon named `gh-token-monitor` — installed on developer machines during the post-install phase. Organizations that identify this daemon and revoke their GitHub tokens before removing it risk triggering a wiper behavior; the recommended sequence is to locate and remove the daemon first, then rotate credentials [8].

Mistral AI and the AI Developer Tooling Attack Surface

Mistral AI published two security advisories confirming the compromise of its packages on both npm and PyPI [6]. The PyPI package `mistralai==2.4.6` was the primary malicious version, published one day after the legitimate `2.4.5` release on May 7. On npm, versions `2.2.2` through `2.2.4` and `1.7.1` through `1.7.3` of `@mistralai` packages were compromised. Notably, the npm payload was non-functional — `setup.mjs` referenced a file that did not exist — while the PyPI payload executed successfully on import [5][6]. Mistral confirmed that its own infrastructure was not compromised; the attack vector was the CI/CD pipeline for SDK publishing.

The window of exposure for the npm packages was narrow: they were live from May 11 at 22:45 UTC to May 12 at 01:53 UTC, approximately three hours [6]. Any organization that ran `npm install` or resolved package manifests during that window against a project depending on the Mistral AI SDK may have installed a malicious version. Because the Mistral AI SDK is commonly installed in the environments of organizations building production AI applications, the blast radius extends beyond individual developers to the CI/CD pipelines and cloud runtime environments of AI product teams.

The compromise of Guardrails AI, a framework widely used to enforce output validation and safety constraints on large language model applications, carries additional implications. Organizations relying on Guardrails AI as a trust boundary within their AI pipelines may have inadvertently installed a credential-stealing payload in the component responsible for enforcing safety controls. The compromise of a safety-enforcement component through a supply chain vector illustrates how supply chain risk pervades every tier of the AI application stack, including the controls layer.

Recommendations

Immediate Actions

Organizations should begin by auditing their dependency lockfiles (`package-lock.json`, `pnpm-lock.yaml`, `yarn.lock`, and `requirements.txt` / `poetry.lock` on Python) for any of the known malicious versions published between May 10 and May 12, 2026. The Socket.dev list of affected packages provides a machine-parseable reference for automated checks [14]. This audit should cover not only current development environments but also CI/CD build artifacts, container images built during the attack window, and any production deployments provisioned from pipelines that ran during that period.

Before rotating GitHub tokens or other CI credentials, security teams must first search developer machines and CI runners for the `gh-token-monitor` daemon and remove it. Revoking credentials while the daemon is still present may trigger the wiper component and result in data loss [8]. Once the daemon is confirmed absent, all credentials that may have been present in environments running the compromised packages should be rotated: GitHub personal access tokens, npm tokens, cloud provider API keys, and Kubernetes service account tokens are the highest priority.

Short-Term Mitigations

GitHub Actions workflows that use the `pull_request_target` trigger should be audited immediately, as this trigger is the initial access vector for the OIDC token extraction chain. Where `pull_request_target` is not strictly necessary, it should be replaced with `pull_request`, which does not grant elevated access to fork-sourced code. Where it is required, workflows should be restructured so that the elevated-permission portion of the workflow never executes code from the pull request branch directly [3][10].

npm OIDC trusted-publisher configurations should be reviewed to apply the narrowest possible scopes. Organizations should evaluate whether their Sigstore-based publish attestation configurations would allow detection of unexpected OIDC subjects or workflow paths, and implement monitoring on the Rekor transparency log for unexpected provenance entries referencing their packages. Deploying software composition analysis tooling with real-time registry monitoring — rather than point-in-time scanning — provides earlier warning of newly published malicious versions before they propagate into lockfiles.

CI/CD pipeline hardening should treat GitHub Actions cache as an untrusted, potentially adversarial input channel. Workflows should use cache keys scoped to the exact commit SHA of the primary branch rather than branch names, and critical secrets-handling steps should run in isolated job contexts that do not share a runner with checkout or test steps that may execute third-party code.

Strategic Considerations

The Mini Shai-Hulud campaign requires organizations to revisit a foundational assumption in modern software supply chain security: that provenance attestation, combined with signed releases, provides sufficient assurance of package integrity. This campaign demonstrates that an attacker operating within a legitimate CI/CD trust boundary can produce attestations that are cryptographically valid yet malicious. Supply chain assurance must therefore be multi-layered, combining provenance verification with behavioral monitoring of package installation scripts, anomaly detection on outbound network connections from CI environments, and continuous dependency risk scoring.

For organizations building on AI SDKs and frameworks, the attack surface extends beyond infrastructure dependencies. The AI tooling layer — model client SDKs, safety and guardrails libraries, orchestration frameworks — now carries supply chain risk comparable to core infrastructure libraries, and deserves proportionate security controls. Organizations that have treated AI SDK dependencies as inherently lower-risk than infrastructure libraries should reassess that posture: this campaign demonstrates that AI tooling is within scope for advanced supply chain actors, particularly given the rapid version publishing cadence common in the AI ecosystem.

More broadly, the ecosystem should not wait for another iteration of this campaign to address the `pull_request_target` attack surface. GitHub, npm, and the Sigstore project have existing mechanisms and roadmap items for tightening OIDC federation scopes, but anecdotal evidence suggests adoption of hardened configurations remains uneven across the npm maintainer community. Security practitioners should use this incident to drive internal policy requiring minimal-privilege OIDC configurations and mandatory workflow audits as conditions of any new CI/CD trusted-publisher registration.

CSA Resource Alignment

This incident intersects with several areas of CSA's ongoing research and framework development.

The **AI Controls Matrix (AICM)** addresses AI supply chain security as a distinct control domain, covering both model provider and application provider responsibilities for ensuring the integrity of AI software components [13]. The Mini Shai-Hulud campaign illustrates precisely the threat scenario the AICM's supply chain controls are designed to address: a dependency in the AI application stack that is itself compromised upstream, affecting consumers at every tier. Organizations should use the AICM to assess whether their current controls provide detection and response capability for CI/CD-level compromise of AI SDK dependencies.

The **MAESTRO framework** for agentic AI threat modeling, which addresses developer tooling and supply chain components as trust boundaries in agentic AI architectures, applies directly to the threat scenario this campaign illustrates. When an AI SDK is also the component used to interact with production AI endpoints — as is the case with the Mistral AI SDK — a supply chain compromise at the SDK level can provide an attacker with both credential access to AI API keys and the ability to manipulate the behavior of production AI agents. MAESTRO's threat enumeration for the developer tooling tier should be revisited in light of this campaign.

CSA's **Zero Trust guidance** is directly applicable to CI/CD pipeline architecture. The OIDC token extraction attack succeeded in part because the compromised workflow ran in an environment where elevated identity tokens were accessible to code executing within the workflow. A Zero Trust interpretation of CI/CD pipeline design would apply least-privilege identity scoping at the step level, not just the job level, and would treat all code executing in a workflow as potentially hostile until verified against a known-good checksum — a posture that would significantly limit the propagation potential of worm-style CI/CD exploits.

The **Cloud Controls Matrix (CCM)** supply chain management domain (STA-09 through STA-11) provides control language for organizations establishing vendor risk programs that include open source dependency governance. The CCM controls for change management and release integrity (CCC domain) are relevant to organizations reviewing their npm and PyPI publishing pipeline configurations in response to this incident.

References

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

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

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

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

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

[6] Mistral AI. "Security Advisories." Mistral AI Documentation, May 2026.

[7] Snyk. "TanStack npm Packages Hit by Mini Shai-Hulud." Snyk Blog, May 2026.

[8] Expel. "Mini Shai-Hulud: Cross-ecosystem supply chain worm targeting npm & PyPI." Expel Blog, May 2026.

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

[10] Picus Security. "Mini Shai-Hulud: The npm Supply Chain Worm Explained." Picus Security Blog, May 2026.

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

[12] BleepingComputer. "Shai Hulud attack ships signed malicious TanStack, Mistral npm packages." BleepingComputer, May 2026.

[13] Cloud Security Alliance. "AI Controls Matrix (AICM) – Introductory Guidance and Implementation Guidelines." Cloud Security Alliance, 2025–2026.

[14] Socket.dev. "TanStack npm Packages Compromised: Mini Shai-Hulud Supply Chain Attack." Socket.dev Blog, May 2026.