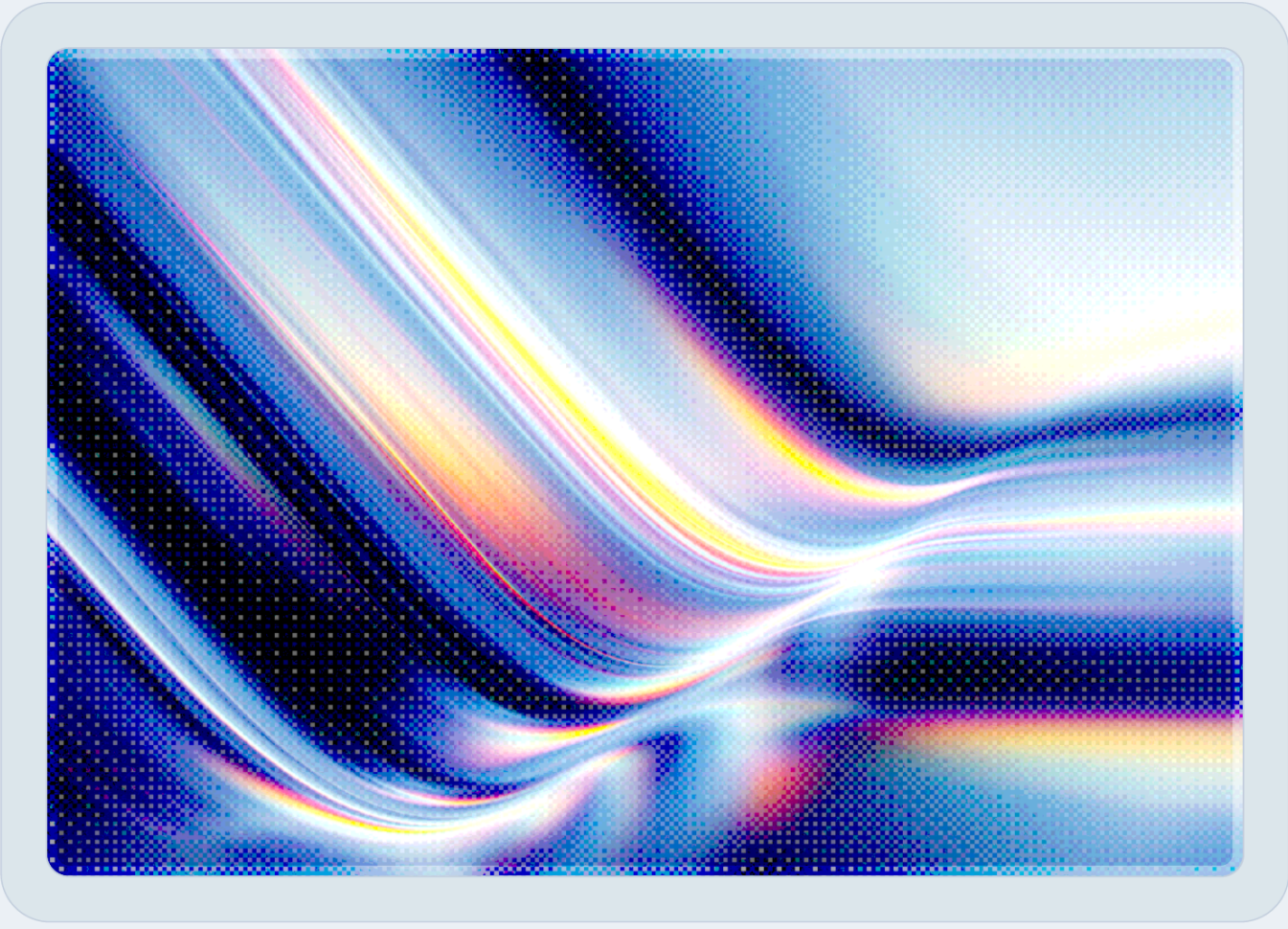
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: npm Worm Targets AI Developer Tooling

TeamPCP's Self-Propagating Supply Chain Attack and the OIDC Token Extraction Vector

2026-05-16

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

- On May 11, 2026, the TeamPCP threat group launched the second wave of the Mini Shai-Hulud campaign, compromising 172 packages across npm and PyPI — including prominent AI ecosystem packages from Mistral AI, Guardrails AI, and TanStack — within a 48-hour window [1][2][14].
- The campaign exploits GitHub Actions' `pull_request_target` trigger and extracts ephemeral OIDC tokens from runner process memory at runtime, bypassing all static credential-based supply chain defenses and enabling publication of malicious package versions through victims' own trusted CI/CD pipelines [3][4].
- Mini Shai-Hulud operates as a true self-replicating worm: after stealing maintainer credentials, it enumerates all packages the victim maintains, injects the malicious payload, increments version numbers, and republishes — creating a compounding blast radius that extends far beyond the initially targeted packages [5].
- A variant designated SANDWORM_MODE specifically targets AI developer tooling — Claude Desktop, Cursor IDE, VS Code with the Continue extension, and Windsurf — injecting rogue Model Context Protocol (MCP) servers into configuration files and harvesting LLM API keys for nine AI providers [6].
- For the first time in documented npm supply chain attacks, Mini Shai-Hulud produced malicious package versions carrying valid SLSA Build Level 3 provenance attestations, undermining a trust signal that many organizations treat as a meaningful security control [7].
- OpenAI confirmed that two employee workstations were compromised through the TanStack supply chain attack and that credential material was exfiltrated from internal source code repositories; users were required to update macOS applications by June 12, 2026 due to code-signing certificate exposure [8].

Background

Software supply chain attacks targeting the npm and PyPI ecosystems have escalated sharply in sophistication and scale since 2023. Early campaigns relied on simple typosquatting — registering package names that closely mimicked legitimate ones — to catch inattentive developers. Subsequent

waves introduced dependency confusion attacks, where malicious packages with names matching internal enterprise libraries were published to public registries and pulled automatically by build systems resolving external-first. Neither technique required an attacker to compromise a legitimate maintainer's account; both were mitigated, at least partially, by scoped package namespaces, private registry precedence rules, and organizational policies around version pinning.

The original Shai-Hulud campaign, tracked by multiple security vendors beginning in late 2025, marked a meaningful inflection point. Rather than impersonating packages, TeamPCP demonstrated that GitHub Actions' trust model could be weaponized to compromise maintainers from the inside out. The `pull_request_target` trigger — a GitHub Actions event designed to allow workflows from external forks to access repository secrets with elevated permissions — proved to be a persistent and underappreciated attack surface. By crafting pull requests that triggered vulnerable workflow files in target repositories, TeamPCP was able to extract short-lived OIDC tokens from the GitHub Actions runner environment. Those tokens, when exchanged with npm's Trusted Publishing system, granted the attacker the same publication authority as the legitimate maintainer — without ever stealing a static credential [3].

The "Mini Shai-Hulud" designation emerged from a late-April 2026 resurgence that initially targeted the SAP npm ecosystem and was initially assessed to be of limited scope. The May 11, 2026 wave revealed that assessment to have been premature. Over a 48-hour period, Microsoft Security Research observed the campaign extend to 42 `@tanstack/*` packages, Mistral AI's npm namespace, Guardrails AI, and dozens of additional packages spanning frontend frameworks, AI tooling, and enterprise software components [1][2][15]. At the same time, SANDWORM_MODE — a variant first identified by Socket Security — introduced capabilities that made plain the campaign's strategic orientation: AI developer tools were not incidental targets but the primary objective tier [6].

Security Analysis

The OIDC Token Extraction Vector

The technical mechanism that gives Mini Shai-Hulud its staying power is the exploitation of GitHub Actions' ephemeral OIDC token system rather than any stored secret. Conventional supply chain attack mitigations — rotating npm tokens, enforcing multi-factor authentication on registry accounts, monitoring for leaked credentials — are ineffective against this vector because no static credential is ever stolen. Instead, the attacker targets the runner process at execution time.

GitHub's Trusted Publishing implementation grants workflows access to a short-lived OIDC token that can be presented to the npm registry in place of a traditional API token. The `pull_request_target` trigger is the critical enabling condition: unlike the standard `pull_request` trigger, workflows responding to `pull_request_target` run in the context of the base repository rather than the fork, and they carry the repository's full secret access. When a target repository's workflow files respond to `pull_request_target` without adequately restricting what actions the workflow will take, an attacker who submits a carefully constructed pull request can cause the legitimate workflow to execute attacker-controlled code with legitimate repository permissions [3][4]. The resulting OIDC token has a short validity window — typically fifteen minutes — but that is more than sufficient to publish a malicious package version.

This technique invalidates the security assumption underlying most enterprise supply chain postures. Organizations that rely on secret rotation schedules, token expiry policies, or audit logs of long-lived credentials will not see the compromise reflected in those controls. The attack leaves no stolen credential to detect; it exploits legitimate process flow.

Self-Propagation Architecture

What distinguishes Mini Shai-Hulud from prior supply chain attacks of similar technical sophistication is its worm architecture. Previous campaigns — including the 2024 `xz` Utils backdoor and the original Shai-Hulud 2.0 wave — required manual targeting of each victim package. Mini Shai-Hulud automates lateral movement within the npm ecosystem using a two-phase propagation engine embedded in its primary payload, `router_init.js`.

In the first phase, the payload executes during the npm `preinstall` lifecycle hook — before most security scanning tools or test suites run — and harvests the full set of package registries and maintainer identities accessible from the compromised build environment. It scans more than 100 file paths and environment variables for credentials, targeting not only npm tokens but also GitHub PATs and OIDC tokens, GitLab CI/CD variables, AWS Secrets Manager values, Azure Key Vault contents, GCP Secret Manager entries, Kubernetes service account JWTs, HashiCorp Vault tokens, SSH keys, and cryptocurrency wallet files [5][9]. The breadth of this credential sweep is designed to maximize lateral movement opportunities across both npm and connected cloud infrastructure.

In the second phase, the worm uses any captured npm publish authority to enumerate all packages the compromised maintainer controls, clone their repositories, inject a copy of the payload, increment the patch version number to trigger automatic updates in downstream consumers, and republish through the maintainer's own trusted pipeline. This cycle compounds: each newly infected package can, in turn, infect the packages of its own contributors. The result is a propagation pattern more characteristic of

network-layer worms than of prior software supply chain attacks, which has motivated security researchers to treat this campaign as a category escalation rather than an incremental threat evolution [5][10].

The payload itself uses triple-layered obfuscation — custom base64 string shuffling, IIFE rotation with checksum matching, and JavaScript-obfuscator control-flow flattening — to impede static analysis. It bundles the source code of the Session Messenger application to increase binary size and make automated content-based detection more costly. Execution persistence is established in Claude Code's hook configuration, VS Code's extension host, and OS-level services, allowing the implant to survive system reboots on developer workstations [5].

AI-Specific Targeting: SANDWORM_MODE and LLM API Key Harvesting

The SANDWORM_MODE variant reveals the campaign's strategic intent with unusual clarity. Rather than treating developer workstations as a means to reach corporate infrastructure — the conventional model for supply chain attacks — SANDWORM_MODE treats AI developer tooling as a high-value target in its own right. The variant's McpInject module enumerates configuration files for Claude Desktop, the Cursor IDE, VS Code with the Continue extension, and the Windsurf editor. It replaces or appends MCP server entries in those configurations with references to attacker-controlled infrastructure [6].

The consequences of an injected MCP server are more severe than a conventional credential theft. A rogue MCP server sits in the tool-use execution path of any AI agent the developer runs, giving the attacker the ability to intercept tool call results, inject prompt content that the model processes as legitimate context, and redirect agent actions toward attacker-controlled endpoints. SANDWORM_MODE explicitly includes prompt injection payloads designed to instruct any AI agent using the poisoned tool configuration to exfiltrate LLM API keys and session tokens through a secondary channel — a technique that exploits the AI agent's instruction-following behavior rather than any vulnerability in the underlying model [6].

The credential sweep in SANDWORM_MODE targets API keys for nine LLM providers: OpenAI, Anthropic, Google, Groq, Together AI, Fireworks AI, Replicate, Mistral AI, and Cohere [6]. This breadth reflects the heterogeneous API key landscape of production AI development environments, where developers routinely hold active keys for multiple providers in local `.env` files and IDE-level environment variable stores. The market value of high-quota LLM API keys on criminal marketplaces has risen substantially in parallel with the expansion of commercial AI development, making them a financially motivated target independent of any interest in the underlying code they are used to produce.

Signed Malicious Artifacts and the SLSA Trust Problem

Perhaps the most technically significant aspect of Mini Shai-Hulud's May 2026 wave is the production of malicious package versions carrying valid SLSA Build Level 3 provenance attestations. SLSA — Supply chain Levels for Software Artifacts — is an industry framework that defines a graduated set of supply chain integrity assurances, with Level 3 representing attestations produced by a hardened, hermetic build system with non-falsifiable provenance records. npm's ecosystem has promoted SLSA attestations as a meaningful signal for downstream consumers assessing package trustworthiness [7].

Because Mini Shai-Hulud exploits the maintainer's own CI/CD pipeline rather than injecting content at the registry level, the malicious packages are built by the legitimate build system and carry the legitimate attestation. From the perspective of a consumer verifying SLSA provenance, the attacker's package is indistinguishable from an authentic release. This does not indicate a flaw in the SLSA specification; it reflects a boundary condition that SLSA was explicitly not designed to address — SLSA attests to build process integrity, not to the integrity of source code or the absence of malicious injection upstream of the build. Security teams that have adopted SLSA attestation verification as a sufficient supply chain control must reassess that posture in light of this campaign [7][4].

Confirmed Breach Impact

OpenAI's public disclosure of its exposure to the TanStack supply chain attack provides the most concrete documentation of downstream harm from this campaign. Two OpenAI employee workstations ingested a compromised TanStack package before the organization's security controls were updated to block the affected versions. Credential material was exfiltrated from internal source code repositories accessed from those workstations, and macOS code-signing certificates were exposed — requiring OpenAI to announce a mandatory app update deadline of June 12, 2026 and to coordinate a certificate rotation that affects the distribution integrity of its entire macOS application portfolio [8].

TanStack is a widely used collection of framework-agnostic React, Solid, Svelte, and Vue utilities. Its presence in AI company developer environments is natural given the prevalence of React-based internal tooling in the industry. The exposure illustrates a consistent pattern: AI companies are prominent consumers of open-source JavaScript frameworks for internal tooling, they employ substantial numbers of developers who maintain active access to sensitive production systems, and their internal environments contain LLM API keys, model weights, training data access, and proprietary source code that have distinctive value to sophisticated adversaries.

Mistral AI's packages were also confirmed compromised in the May 2026 wave, with reports indicating that Mistral's internal source code appeared on criminal sale platforms following the breach [1][2]. Guardrails AI — a framework specifically designed to enforce safety constraints on LLM outputs — was

among the affected packages, which carries particular operational irony given the nature of its function.

Recommendations

Immediate Actions

Organizations should audit their dependency trees for any of the 172 packages confirmed compromised in the Mini Shai-Hulud campaign across npm and PyPI registries [11]. The affected version ranges have been documented in advisories from npm, PyPI, GitHub, and multiple security vendors; security teams should cross-reference their software bills of materials (SBOMs) against published package lists from Mend, Socket, Snyk, and Wiz. For any confirmed exposure, assume that all credentials accessible from the build environment were compromised and initiate full rotation — including npm tokens, cloud provider credentials, LLM API keys present in CI/CD environment variables, and any SSH keys or Kubernetes service account tokens discoverable from the infected build context.

Developers who use Claude Desktop, Cursor IDE, VS Code with the Continue extension, or Windsurf should immediately inspect their MCP server configuration files for unexpected entries. On macOS, Claude Desktop's configuration is located at `~/Library/Application Support/Claude/claude_desktop_config.json`; Cursor's is at `~/.cursor/mcp.json`. Any server entry referencing an unfamiliar hostname or IP address should be treated as a potential SANDWORM_MODE injection. LLM API keys stored in local `.env` files, IDE environment stores, or shell configuration files should be rotated if the developer's workstation was used during or after the May 11–12, 2026 attack window.

OpenAI macOS application users who have not yet updated their applications should do so before the June 12, 2026 certificate rotation deadline. After that date, applications signed with the exposed certificates will no longer be trusted by macOS Gatekeeper. Organizations that manage macOS fleets through mobile device management (MDM) should verify that application updates have been pushed and confirmed across affected devices.

Short-Term Mitigations

Organizations that maintain open-source packages or use GitHub Actions for software publishing should audit all workflow files for `pull_request_target` trigger usage. Any workflow that responds to `pull_request_target` and also performs publishing, credential access, or external network calls

should be reviewed against GitHub's security hardening documentation [12]. The recommended remediation is to separate privileged workflows from those that process external pull request content — using the `pull_request` trigger (which runs in the fork's context without elevated permissions) for CI validation, and reserving the `pull_request_target` trigger exclusively for workflows that never execute code from the incoming pull request.

Trusted Publishing configurations on npm should be reviewed to confirm that OIDC token scopes are as narrow as technically feasible. Where possible, publishing workflows should be restricted to specific branches and environments using GitHub's environment protection rules, which add an approval gate before a privileged workflow environment's secrets are accessible. This does not eliminate the OIDC token extraction vector, but it introduces a human control point that can limit the attack surface.

Dependency pinning — specifying exact versions rather than version ranges or `latest` tags — is a necessary but incomplete mitigation. It prevents automatic ingestion of newly published malicious versions but does not protect against worm propagation that increments a patch version within an already-trusted range. Organizations should supplement version pinning with integrity verification using npm's lockfile hash validation and automated SCA (Software Composition Analysis) scanning against known-compromised version databases maintained by vendors such as Socket, Snyk, Mend, and Endor Labs.

Strategic Considerations

The Mini Shai-Hulud campaign should prompt a fundamental reassessment of how organizations categorize AI developer tooling in their threat models. Historically, developer workstations and build environments have been treated as a lower-risk tier relative to production infrastructure — a classification that reflects the traditional assumption that an attacker seeking production access would target production systems directly. AI developer environments invert this hierarchy. A compromised AI developer workstation may contain active LLM API keys with high usage quotas, access to proprietary model architectures and training pipelines, MCP server configurations that bridge the developer's AI agents into production data stores, and source code for safety-critical systems. The value of that access is in many respects comparable to — and in some threat models, exceeds — the value of conventional production server access.

CSA recommends that security architects apply the same access control, monitoring, and incident response rigor to AI developer tooling that they apply to privileged infrastructure. LLM API keys should be managed through secrets management systems rather than stored in files or environment variables on developer workstations; they should carry the same rotation policies and audit logging as cloud provider credentials. MCP server configurations should be treated as a trust boundary and subject to

review before new entries are authorized, on the same basis as firewall rule changes. AI IDE integrations should be catalogued in the organization's software inventory and evaluated against the organization's software supply chain policy.

The open-sourcing of the Shai-Hulud framework by TeamPCP on May 13, 2026 substantially broadens the threat actor population capable of executing this class of attack [13]. Organizations should not plan on a mitigation window before lower-sophistication actors adapt the tooling. The OIDC token extraction technique and the MCP server injection module are now documented in publicly available code, which means that effective defense must treat these vectors as widely accessible rather than requiring nation-state or advanced persistent threat (APT) resources.

CSA Resource Alignment

The Mini Shai-Hulud campaign maps directly to threat categories and control domains addressed across several CSA frameworks, enabling organizations to anchor their response to existing governance structures rather than treating this as an entirely novel risk category.

The CSA MAESTRO framework for agentic AI threat modeling defines a threat class around "tool and plugin poisoning" — adversarial manipulation of the tool-use interfaces available to AI agents. SANDWORM_MODE's MCP server injection is a precise instantiation of this threat: by replacing a legitimate MCP server with an attacker-controlled one, the campaign gains the ability to manipulate every tool call the AI agent makes, inject context into the agent's reasoning process, and redirect agent actions without any vulnerability in the model itself. Security architects using MAESTRO to evaluate their agentic AI deployments should incorporate supply chain integrity of MCP server configurations as a first-class threat scenario.

The AI Controls Matrix (AICM), CSA's AI-specific extension of the Cloud Controls Matrix, addresses software supply chain risk in its supply chain and dependency management control domains. The AICM's controls around SBOM maintenance, third-party component vetting, and build pipeline integrity are directly applicable to the organizational mitigations described above. Practitioners implementing AICM controls should extend their SBOM coverage to include AI-specific tooling — LLM SDKs, agent frameworks, MCP server implementations, and AI IDE extensions — which have historically been excluded from software inventory processes focused on production application dependencies.

CSA's AI Organizational Responsibilities series — particularly its guidance on AI tools and applications — addresses the governance dimension of AI developer tooling risk. The series establishes that organizations are responsible for the security of the AI tools they deploy to employees, including the

supply chain integrity of those tools. This responsibility extends to MCP server configurations, which function as a form of AI plugin that significantly expands the attack surface of any AI agent deployment. Organizations should establish a review and approval process for MCP server additions that is proportionate to the access those servers provide.

The CSA STAR program's continuous monitoring controls provide a framework for operationalizing the detection capabilities recommended above. Continuous dependency scanning against known-compromised package databases, CI/CD pipeline anomaly monitoring, and LLM API key usage telemetry all align to STAR's monitoring control objectives and can be incorporated into existing STAR-based compliance programs without requiring new assessment categories.

References

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

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

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

[4] BleepingComputer. "Shai Hulud Attack Ships Signed Malicious TanStack, Mistral npm Packages." BleepingComputer, May 2026.

[5] Mend.io. "Mini Shai-Hulud Is Back: 172 npm and PyPI Packages Compromised in Latest Wave." Mend Security Blog, May 2026.

[6] Socket Security. "SANDWORM_MODE: Shai-Hulud-Style npm Worm Hijacks CI Workflow." Socket.dev Blog, May 2026.

[7] Vectra AI. "Shai-Hulud Part 2: When the Worm Forged Its Own Security Certificate." Vectra Blog, May 2026.

[8] OpenAI. "Our Response to the TanStack npm Supply Chain Attack." OpenAI, May 2026.

[9] CyberPress. "Shai-Hulud Worm Steals Developer Secrets Across npm, GitHub, AWS, and Kubernetes." CyberPress, May 2026.

[10] ReversingLabs. "Team PCP's Mini Shai-Hulud Tears at Open-Source Trust." ReversingLabs Blog, May 2026.

[11] NHS England Digital. "Supply Chain Attack Affecting Numerous npm and PyPI Packages." NHS Cyber Alert CC-4781, May 2026.

[12] GitHub. "Security Hardening for GitHub Actions." GitHub Docs. Accessed May 2026.

[13] SC Media. "TeamPCP Releases 'Vibe Coded' Shai-Hulud Source Code, Issues Challenge." SC Media, May 2026.

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

[14] Aikido Security. "Mini Shai-Hulud Is Back: npm Worm Hits Over 160 Packages, Including Mistral and TanStack." Aikido Blog, May 2026.

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