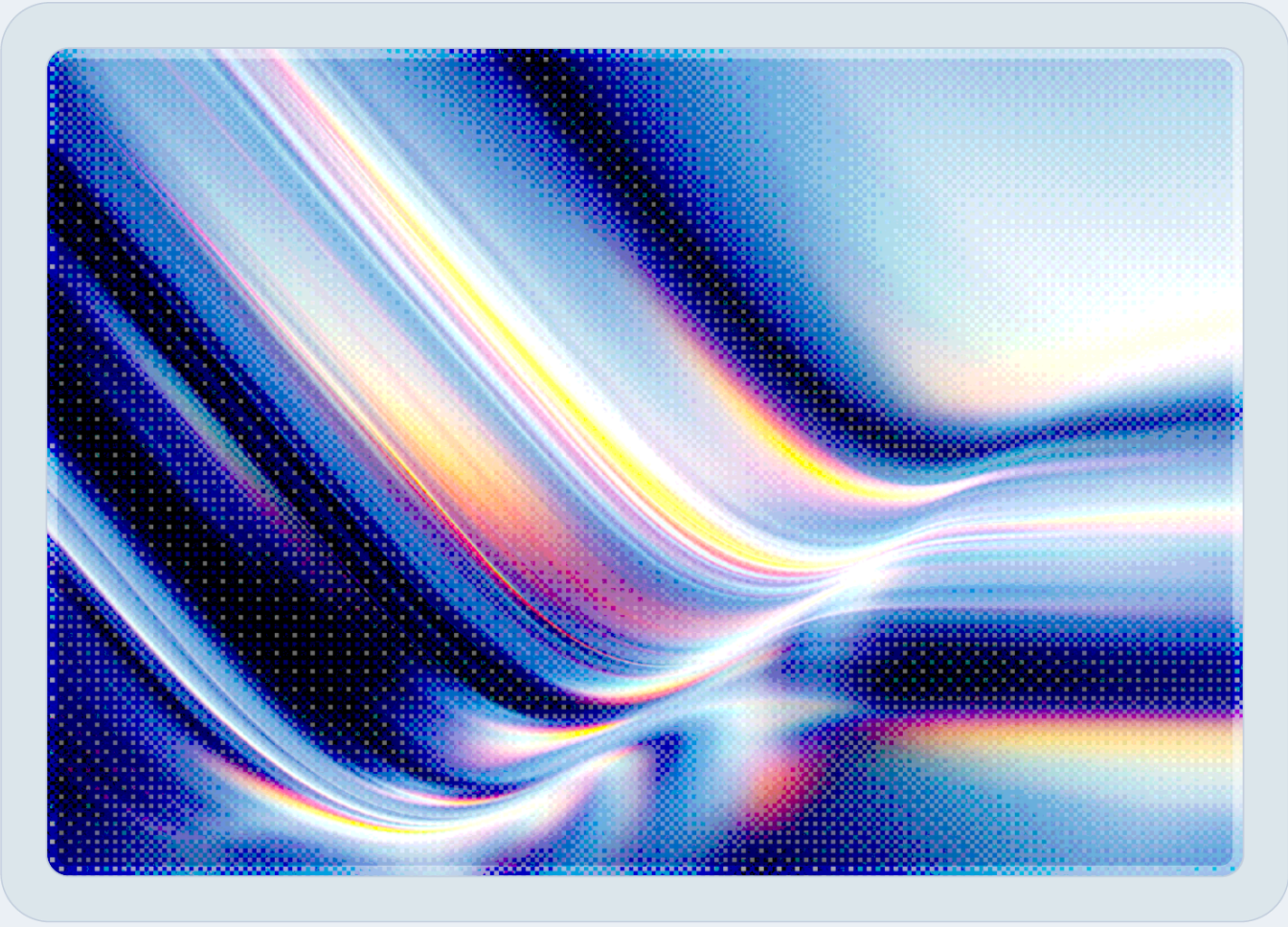
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: TeamPCP Worm Targets AI Developer Toolchain

Third Wave of the Shai-Hulud Campaign Family Introduces AI Agent Persistence and SLSA Provenance Bypass

2026-05-18

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- TeamPCP's "Mini Shai-Hulud" campaign — publicly disclosed May 12, 2026 — compromised over 170 npm and PyPI packages carrying more than 518 million cumulative downloads, including widely-adopted AI developer libraries from TanStack, Mistral AI, UiPath, and Guardrails AI [1][2].
- For the first time in a documented supply chain attack, the malware weaponized 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 remediation and credential rotation [3][4].
- The worm produces cryptographically valid SLSA Build Level 3 provenance attestations for its malicious packages by subverting the trusted publishing infrastructure rather than forging signatures, demonstrating that attestation-based trust is insufficient where CI/CD pipelines themselves are compromised [1][2].
- A "dead-man's switch" wiper daemon executes `rm -rf ~/` if a victim revokes the attacker's npm token; affected organizations must remove the persistence daemon before rotating any credentials [2][4].
- No CVE, GHSA, or OSV advisory record was assigned to any compromised package despite confirmed mass impact, exposing a structural gap in the transient-compromise detection model on which most software composition analysis tooling depends [5].

Background

The Shai-Hulud campaign family has shaped the npm supply chain threat landscape since the original worm was identified in September 2025. That campaign introduced a self-replicating pattern that used stolen npm publish tokens to propagate infected patch releases across maintainer package portfolios, transforming a single compromised account into a cascading registry-level infection [6]. Shai-Hulud 2.0 followed between November 24 and December 1, 2025, extending the same worm architecture to target maintainers associated with high-profile organizations including Zapier, PostHog, and Postman. Microsoft published detection and investigation guidance in December 2025 [6].

The threat actor now attributed to this campaign lineage — designated **TeamPCP**, also tracked as PCPcat, ShellForce, and DeadCatx3 — emerged as an identifiable group no later than September 2025. Unit 42 researchers describe TeamPCP as a cloud-focused cybercriminal organization with roots in cryptocurrency theft and ransomware operations that pivoted in early 2026 toward what the group itself characterizes as systematic supply chain compromise [7]. In March 2026, TeamPCP executed a cascading four-stage campaign that compromised Aqua Security's Trivy vulnerability scanner (March 19), Checkmarx's KICS static analysis tool (March 21), the LiteLLM AI gateway library (March 23), and the Telnyx Python SDK (March 27) — in that order, using credentials harvested from each tool to authenticate into the next target [7][8]. By the campaign's conclusion, researchers estimated the group had exfiltrated more than 300 gigabytes of data and 500,000 credentials from affected systems [7].

The broader operational tempo in the weeks preceding Mini Shai-Hulud is notable. GitGuardian documented three distinct coordinated campaigns striking npm, PyPI, and Docker Hub registries within a roughly 48-hour window spanning April 21–23, 2026 [9]. A week later, on April 29–30, TeamPCP executed the first recognizable Mini Shai-Hulud wave: a synchronized multi-ecosystem operation that compromised SAP's MTA Build Tool, the @cap-js database connector suite for SAP Cloud Application Programming Model, the Intercom Node.js client, and the PyTorch Lightning machine learning framework across npm, PyPI, and Packagist simultaneously [3]. That wave reached approximately 1,800 repositories and a combined weekly download volume exceeding 930,000 [3]. The root causes were an exposed npm automation token in CircleCI pull-request build logs and an overly permissive npm Trusted Publishing OIDC policy that allowed any repository workflow, rather than only designated release workflows, to mint publish tokens [3].

The May 2026 wave expanded dramatically in both scope and operational sophistication, introducing two techniques that elevate it to a distinct threat tier.

Security Analysis

The Attack Chain

Mini Shai-Hulud's compromise of the TanStack router ecosystem demonstrates a multi-stage GitHub Actions exploitation technique that is both technically sophisticated and reproducible wherever CI/CD workflows contain common misconfigurations. Attackers forked the TanStack/router repository and submitted a pull request designed to trigger a `pull_request_target` workflow — a GitHub Actions event type that runs in the context of the base repository rather than the fork, granting access to repository secrets unavailable in standard fork-triggered workflows [2][3]. Simultaneously, the attackers

poisoned the repository's GitHub Actions build cache by injecting a malicious pnpm store. When a legitimate TanStack maintainer subsequently triggered a release workflow, the runner restored the poisoned cache and executed attacker-controlled binaries, which extracted short-lived OIDC tokens directly from the GitHub Actions runner process memory via `/proc/<pid>/mem` [2]. Those tokens were then used to publish malicious package releases without ever possessing the maintainer's long-term npm credentials.

This technique produces artifacts that carry the full imprimatur of legitimate CI/CD pipelines. The malicious TanStack packages — assigned CVE-2026-45321 with a CVSS score of 9.6 [1][10] — carried cryptographically valid SLSA Build Level 3 provenance attestations because the signing infrastructure itself had been subverted rather than any signing key stolen. The provenance attestations were authentic in a technical sense: the compromised build runner generated them honestly, without knowing that its inputs had been poisoned. Organizations that rely on provenance verification as a terminal trust signal received a false assurance. Adding to the detection failure, no CVE, GHSA, or OSV advisory record existed for any of the compromised packages at the time of initial public disclosure [5], meaning organizations relying on vulnerability advisory feeds received no automated warning.

The malicious payload — an 11.7-megabyte obfuscated JavaScript file delivered through the Bun JavaScript runtime, which evades many Node.js-native endpoint detection tools — targeted 134 distinct credential file paths [3][5]. Targeted material included GitHub personal access tokens and SSH keys, npm automation tokens with `bypass_2fa: true` capability, AWS static credential files and Instance Metadata Service endpoints, Azure and GCP configuration files, Kubernetes `kubeconfig` files, HashiCorp Vault tokens, and AI tooling secrets including `~/.claude.json` and Model Context Protocol server configuration files [3][5]. Exfiltrated data was encrypted with AES-256-GCM under RSA-4096 key wrapping before transmission through a triple-channel command-and-control architecture: the typosquat domain `git-tanstack[.]com`, the Session decentralized messaging network (routing through `getsession.org` seed nodes to blend with legitimate encrypted chat traffic), and GitHub API dead drops in attacker-created repositories [2][4]. More than 400 repositories were created on victim accounts bearing the string "Shai-Hulud: Here We Go Again" in their descriptions [1].

AI Coding Agent Persistence

The feature that distinguishes Mini Shai-Hulud from all prior software supply chain worms is its deliberate exploitation of AI coding agent configuration files as persistent post-compromise infrastructure. After harvesting credentials, the payload modifies two files in the victim's repository: it inserts a `SessionStart` hook into `.claude/settings.json` that executes a malicious Node.js script (`node .vscode/setup.mjs`) on every Claude Code session, and it adds a `folderOpen` trigger

task to `.vscode/tasks.json` that fires on every VS Code project open [3][5]. Both modifications are committed to the repository under the commit author identity `claude@users.noreply.github.com` with the message "chore: update dependencies" — a social engineering detail designed to cause developers to dismiss the commit as ordinary AI-assisted housekeeping [3].

The consequence is that standard remediation steps are insufficient. A developer who uninstalls the infected package, pins their lockfile to a clean version, and rotates credentials remains fully compromised if these configuration files are not also inspected and cleaned. Security researcher Stephen Thoemmes noted that "directories like .claude/ and .vscode/ are typically excluded from version control via .gitignore and are rarely scrutinized as viable attack surfaces" [4]. This observation reflects a trust assumption built into AI-assisted development: that tool configuration directories in a developer's working tree are authoritative and benign. The attack exploits that assumption to establish a persistence mechanism that is both invisible to dependency scanning and durable against the remediation steps most incident response playbooks prescribe for supply chain compromise.

This is the first documented supply chain attack to weaponize AI coding agent configurations as a persistence vector [3][4]. Its significance extends beyond this specific incident: as AI coding agents become standard components of developer workflows, the configuration files, extension manifests, and session hooks they consume represent an expanding attack surface that security tooling has not historically monitored.

Self-Propagation, Wiper Payload, and Destructive Logic

The worm component enumerates all npm packages accessible via stolen tokens, autonomously publishes infected patch-version increments via the npm registry HTTP PUT endpoint, and generates routine-appearing version bumps that cascade the infection across each maintainer's entire portfolio [2][3]. Across the May campaign, this mechanism produced 373 malicious entries across 169 npm packages and 2 PyPI packages — the PyPI packages (`guardrails-ai 0.10.1` and `mistralai 2.4.6`) executing their payload on Python import without requiring installation lifecycle hooks [2].

The payload installs a persistent background daemon called "gh-token-monitor" that polls the GitHub API every 60 seconds to verify whether a specific npm token remains valid [2][4]. That token is named "IfYouRevokeThisTokenItWillWipeTheComputerOfTheOwner" on the victim's npm account. If the token is revoked, the daemon executes `rm -rf ~/`, wiping the developer's home directory. On macOS the daemon persists as a LaunchAgent plist at `~/Library/LaunchAgents/com.user.gh-token-monitor.plist`; on Linux as a systemd user service at `~/.config/systemd/user/gh-token-monitor.service`. The daemon auto-exits after 24 hours without triggering [2]. The PyPI

variants embed an additional geofenced destructive routine: on systems where locale settings indicate Israel or Iran, there is a one-in-six probability the payload executes `rm -rf /` rather than the home-directory-scoped wipe [3].

The wiper architecture creates a specific incident response trap. Credential rotation — the expected first response to a supply chain compromise — becomes a trigger for irreversible data destruction if the daemon has not already been removed. Socket CEO Feross Aboukhadjieh articulated the broader problem: "There is no single centralized kill switch for this kind of campaign. You can pull a package from the registry, but you cannot automatically pull back the credentials it may have already stolen." [4]

Scope and Affected Packages

The table below summarizes the principal affected packages disclosed as part of the May 2026 campaign based on available reporting at time of publication.

Package / Ecosystem	Versions Affected	Approximate Weekly Downloads
@tanstack/* namespace (npm)	84 malicious artifacts across 42 packages [11]	12,000,000+ (@tanstack/router) [1]
@uipath/* suite (npm)	66 entries	Undisclosed
@squawk/* (npm)	87 entries	Undisclosed
@mistralai/* (npm)	Multiple versions	Undisclosed
mistralai (PyPI)	2.4.6	Undisclosed
guardrails-ai (PyPI)	0.10.1	Undisclosed
@opensearch-project/opensearch (npm)	3.5.3, 3.6.2, 3.7.0, 3.8.0	Undisclosed

Total campaign scope: 373 malicious entries across 169 npm packages and 2 PyPI packages; 518 million+ cumulative downloads across affected packages, per OX Security analysis as reported by [1].

Recommendations

Immediate Actions

Organizations that have installed any affected packages should treat all systems where those versions executed as potentially compromised pending investigation. The sequence of remediation steps matters: performing them out of order risks triggering the destructive wiper payload.

Step one is daemon removal, not credential rotation. Before revoking any npm tokens, GitHub personal access tokens, or CI/CD secrets, locate and remove the "gh-token-monitor" persistence daemon. On macOS, check for and delete `~/Library/LaunchAgents/com.user.gh-token-monitor.plist`; on Linux, check `~/.config/systemd/user/gh-token-monitor.service`. Confirm the daemon process is not running (`ps aux | grep gh-token-monitor`) before proceeding.

Step two is persistence artifact removal. Audit all repository working directories for the injected files: `router_runtime.js`, `setup.mjs`, modified `.claude/settings.json` (look for unexpected `SessionStart` hooks), and modified `.vscode/tasks.json` (look for unexpected `folderOpen` tasks). Remove these files, then audit git history for commits authored as `claude@users.noreply.github.com` with the message "chore: update dependencies" — revert them if found.

Step three is full credential rotation covering npm tokens, GitHub personal access tokens, AWS and cloud provider credentials, Kubernetes service account tokens, and all CI/CD pipeline secrets. Audit GitHub for attacker-created repositories on victim accounts bearing "Shai-Hulud" in their descriptions and delete them. Block the known C2 infrastructure at DNS and proxy layers: `git-tanstack[.]com`, `*.getsession.org`, and `83.142.209[.]194`.

Short-Term Mitigations

The root cause in both the April and May campaign waves was CI/CD misconfiguration. Organizations should audit their npm Trusted Publishing OIDC subject claim policies and restrict them to require that publish credentials be issued only from designated, protected-branch release workflows. Any `pull_request_target` workflow that accesses secrets should be reviewed — this trigger is rarely necessary and carries elevated risk; most uses can be replaced with the standard `pull_request` trigger, which runs without base repository secret access.

Organizations should enforce package lockfile integrity: `npm ci` with cryptographically verified lockfiles in CI pipelines, and `pip install --require-hashes` for Python dependencies. Standard software composition analysis tools operating against CVE and GHSA advisory databases will not detect transient registry compromises that lack advisory records. Supplement them with behavioral monitoring that flags unexpected credential-file access patterns, anomalous outbound TLS connections to unfamiliar endpoints, and npm publish activity originating from developer workstations rather than CI runners.

AI coding agent configuration directories — `.claude/`, `.vscode/`, `.cursor/`, and analogous directories for any AI-assisted development tools in use — should be added to internal security review scope and, where feasible, to integrity monitoring. The assumption that these directories contain only benign developer preferences is no longer tenable.

Strategic Considerations

Mini Shai-Hulud exposes two structural gaps that will not be resolved by incident response alone. First, SLSA provenance attestation is a necessary but not sufficient trust signal. This campaign bypassed SLSA Build Level 3 attestation not by forging a signature but by compromising the build infrastructure that generated authentic signatures over poisoned inputs. Trustworthy supply chains require integrity verification at each pipeline stage — including protection against cache poisoning and OIDC token extraction from CI/CD runner memory — not only artifact-level provenance checking at the end of the pipeline.

Second, the software vulnerability advisory ecosystem lacks a standard mechanism for registering transient registry compromises. Because Mini Shai-Hulud involved malicious packages published through legitimate accounts rather than code defects in open-source software, the standard CVE and GHSA advisory channels had no applicable process. The lack of any advisory record despite confirmed 518-million-download exposure created a detection gap that every organization relying on automated vulnerability scanning experienced simultaneously. Registry operators, security advisory bodies, and package managers should collaboratively develop an advisory classification and notification channel for transient registry compromise events, distinct from the traditional CVE model for code-level defects.

CSA Resource Alignment

Mini Shai-Hulud maps to threat models and control frameworks that the Cloud Security Alliance has developed specifically for AI-native development environments.

CSA's **MAESTRO** framework for agentic AI threat modeling addresses the integrity of AI agent configuration and tool invocation pathways [12]. The abuse of `.claude/settings.json` SessionStart hooks is a concrete instantiation of the tool-injection threat class MAESTRO defines — an adversary gaining persistent control over the commands an AI agent executes without the developer's knowledge. Security teams applying MAESTRO to their development environments should explicitly model AI agent configuration files, session hooks, and editor extension manifests as a persistence attack surface in their threat models, and include integrity checks for those files in their CI/CD security gate requirements.

The **AI Controls Matrix (AICM)** provides control objectives covering build pipeline security, publish credential scoping, and software component provenance verification — the precise controls whose misconfiguration TeamPCP exploited across the March and May 2026 campaigns [13]. AICM extends the Cloud Controls Matrix to address AI-specific pipeline and deployment risks; organizations that have implemented cloud supply chain controls under CCM without extending them to AI development tooling have left the expanded surface AICM covers unaddressed.

CSA's **Zero Trust** guidance applies directly to CI/CD credential architecture. Publish credentials should be scoped using OIDC subject claims to the minimum necessary workflow, branch, and repository combination, and should not be available in pull-request or fork-triggered contexts. The long-lived token pattern exploited in these campaigns — where a single exposed CI/CD secret provides lateral movement far beyond the task that generated it — is a Zero Trust misconfiguration independent of the supply chain framing.

Finally, organizations publishing software packages should consider CSA's **STAR** program as a framework for communicating supply chain security posture to downstream consumers, including their OIDC policy configuration, provenance attestation practices, and CI/CD credential scoping policies. Transparent security self-assessment in these areas would enable ecosystem participants to evaluate publisher trustworthiness beyond the binary signal of a valid signature.

References

[1] The Hacker News. "Mini Shai-Hulud Worm Compromises TanStack, Mistral AI, Guardrails AI & More Packages." The Hacker News, May 12, 2026.

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

[3] Cloud Security Alliance AI Safety Initiative. "Mini Shai-Hulud: Multi-Ecosystem Developer Supply Chain Attack." CSA Labs, April–May 2026.

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

[5] Cloud Security Alliance AI Safety Initiative. "TeamPCP: Cascading Supply Chain Attack on AI/ML Tooling." CSA Labs, March 2026.

[6] Microsoft Security Response Center. "Shai-Hulud 2.0: Guidance for Detecting, Investigating, and Defending Against the Supply Chain Attack." Microsoft Security Blog, December 9, 2025.

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

[8] Cloud Security Alliance AI Safety Initiative. "TeamPCP and the Cascading AI/ML Supply Chain Campaign." CSA Labs, March 29, 2026.

[9] GitGuardian. "No Off Season: Three Supply Chain Campaigns Hit npm, PyPI, and Docker Hub in 48 Hours." GitGuardian Blog, April 2026.

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

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

[12] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." Cloud Security Alliance, February 6, 2025.

[13] Cloud Security Alliance. "AI Controls Matrix." Cloud Security Alliance, 2025.