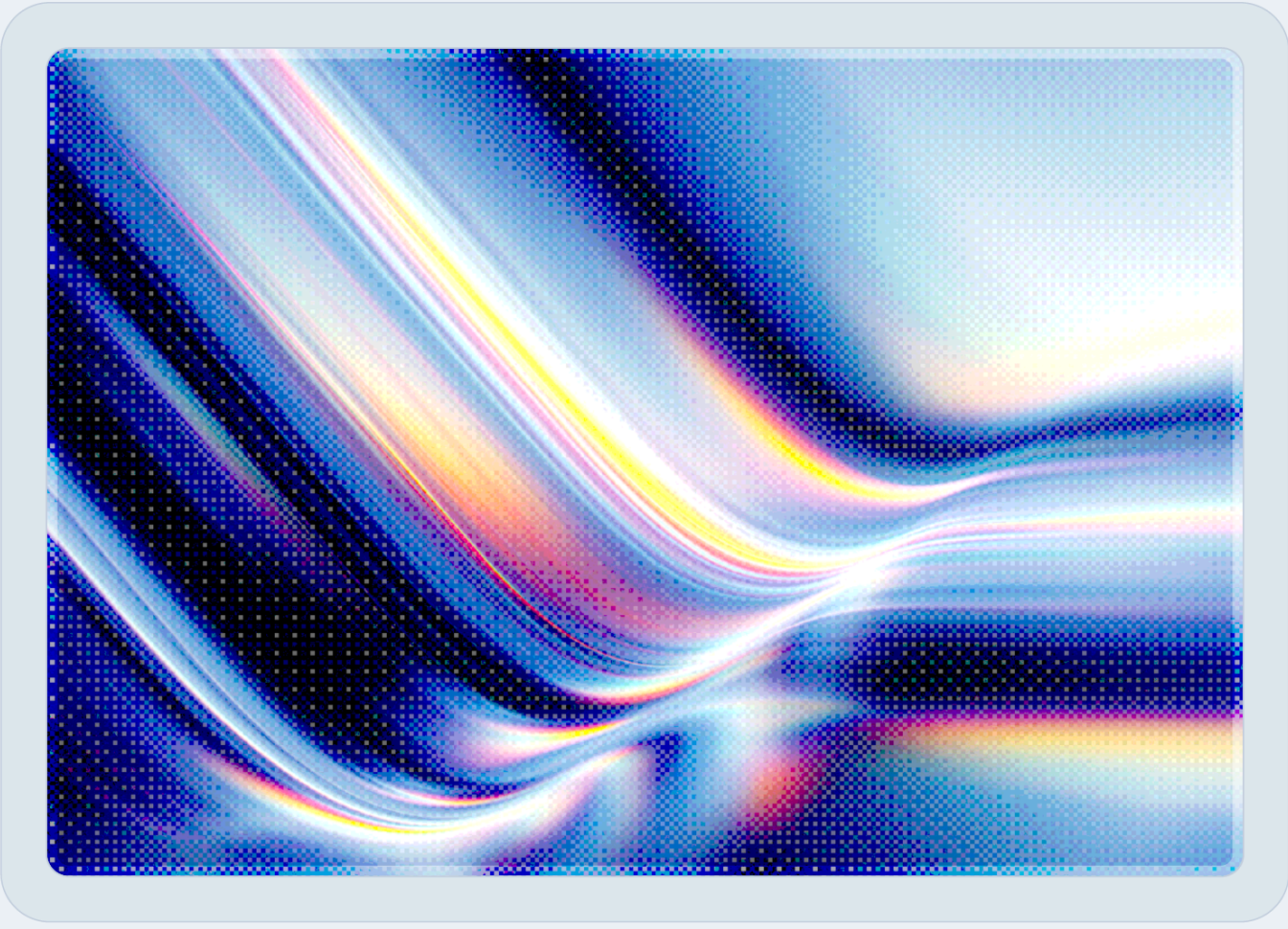
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: AI Supply Chain Worm Hits npm and PyPI

TeamPCP's Self-Propagating Campaign Compromises AI Developer Tooling and CI/CD Infrastructure

2026-05-13

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- TeamPCP, a financially motivated threat group also tracked as PCPcat and ShellForce, launched the "Mini Shai-Hulud" campaign in March 2026, with a major escalation on May 10–11, 2026, compromising approximately 170 npm packages and multiple PyPI libraries — including AI-specific tools from Mistral AI, Guardrails AI, and LiteLLM [1][2].
- The campaign represents the first documented npm worm to produce validly-signed SLSA Build Level 3 provenance attestations, meaning standard integrity checks passed despite the packages containing malicious payloads [10][11].
- Attackers harvested stolen GitHub OIDC tokens from CI/CD pipelines to publish directly to npm registries under legitimate project namespaces, reaching an estimated 518 million cumulative downloads across affected package versions [11].
- The malware installs persistence mechanisms inside Claude Code and VS Code, creating token-monitoring services and injecting malicious GitHub Actions workflows — effectively converting developer workstations into long-lived credential-exfiltration nodes [5].
- Organizations using AI developer SDKs, particularly TypeScript and Python tooling for LLM integration, should audit dependencies published between March 19 and May 12, 2026, and rotate all credentials stored in CI/CD environments that ran the affected package versions.

Background

Software supply chain attacks have grown in frequency and complexity since the SolarWinds incident of 2020, as documented in successive years of industry threat reporting, but 2026 has introduced a qualitatively new threat pattern: self-propagating worms that traverse package ecosystems autonomously while carrying valid cryptographic provenance signatures. The campaign known as Mini Shai-Hulud, attributed to the threat group TeamPCP, embodies this shift [1].

TeamPCP first appeared in security reporting in late 2025 and has since been tracked under multiple aliases including PCPcat, ShellForce, and DeadCatx3 [6]. The group is assessed as financially motivated, focusing on credential theft from developer environments, cloud infrastructure, and cryptocurrency

wallets rather than ransomware or nation-state espionage objectives. Their early campaigns in late 2025 targeted individual packages; by early 2026, the operation had expanded into a coordinated, multi-ecosystem campaign with significantly greater scope and technical complexity [6].

The Mini Shai-Hulud campaign began in earnest on March 19, 2026, when Aqua Security's Trivy scanner (version 0.69.4) was compromised — a choice that targeted security tooling itself and may reflect deliberate effort to undermine the integrity of defenses [6]. Within days, the worm had propagated across dozens of npm packages. Further attacks on March 23 compromised Checkmarx's KICS static analysis tool and the OpenVSX extension registry. On March 24, LiteLLM versions 1.82.7 and 1.82.8 were published to PyPI containing the malicious payload, followed on March 27 by Telnyx versions 4.87.1 and 4.87.2 [7]. April brought additional compromises including the Bitwarden CLI npm package and the xinference PyPI package for distributed LLM inference [6].

The campaign's most expansive action to date occurred on May 10–11, 2026, when TeamPCP executed a coordinated attack against the TanStack ecosystem on npm — a suite of popular JavaScript libraries with multi-million weekly download counts — alongside packages from Mistral AI, UiPath, and OpenSearch [1][2][4]. This phase drew widespread attention from the security research community and prompted coordinated incident response from npm, PyPI, and GitHub.

Security Analysis

Attack Architecture: Three-Stage Compromise Chain

The Mini Shai-Hulud attack chain operates in three coordinated phases that collectively bypass multiple layers of conventional software supply chain defenses [3][5].

In the first phase, attackers establish a payload staging environment by creating a GitHub fork — identified as the "voicproducoes" account — containing a 2.3 MB obfuscated JavaScript payload named `tanstack_runner.js` alongside a weaponized `package.json` [1][5]. The obfuscation leverages the Bun JavaScript runtime's bundling capabilities to produce a single executable blob designed to evade static analysis tooling that relies on signature-based detection of known JavaScript malware patterns [3]. This staging infrastructure is deliberately positioned inside GitHub's trusted domain, where it can be fetched by CI/CD systems without triggering network-level controls.

The second phase exploits the inherent trust relationship between GitHub Actions and npm's OIDC-based publishing workflow. GitHub Actions runners generate short-lived OpenID Connect tokens that can be exchanged for npm publish credentials without requiring stored secrets. TeamPCP harvests these

tokens by injecting itself into workflows during the first npm install cycle, then uses the stolen tokens to publish malicious package versions to legitimate namespaces [3]. The critical consequence of this approach is that the published packages carry valid SLSA Build Level 3 provenance attestations — the same attestations that compliance frameworks and automated verification tools treat as proof of a trusted build pipeline. An organization relying solely on SLSA attestation verification has no automated attestation-based mechanism to detect these packages as malicious — attestations verify process, not payload. Complementary behavioral controls are therefore essential.

The third phase, executed by the malicious payload itself, performs broad credential harvesting from the compromised developer environment. The payload scrapes memory from GitHub Actions Runner.Worker processes, queries AWS IMDS and ECS metadata service endpoints, reads from over 100 hardcoded file paths including SSH private keys, cloud provider credential files, and cryptocurrency wallet keyfiles, scans for Kubernetes service account tokens, and extracts secrets from HashiCorp Vault where the process has ambient access [1][5]. Exfiltrated data traverses three redundant channels: the Session Protocol (a decentralized communication layer used by certain AI coding assistants) [5], GitHub API dead drops using the stolen tokens, and attacker-controlled typosquat domains including `git-tanstack[.]com` [4][5].

Persistence in AI Developer Tooling

A particularly consequential aspect of the May campaign is the malware's targeted persistence within AI-integrated development environments. The payload installs hooks inside Claude Code and VS Code, creating token-monitoring services that run at the OS service level and survive reboots [5]. It also injects malicious GitHub Actions workflows into any repository the developer's compromised token can access, establishing lateral propagation paths that extend the worm's reach beyond the originally infected workstation into the developer's broader organizational footprint.

The inclusion of a dead-man's switch mechanism further distinguishes Mini Shai-Hulud from opportunistic supply chain attacks. If a developer detects the compromise and revokes the stolen npm publish token, the malware may trigger a destructive wipe of the infected system [5]. This mechanism meaningfully raises the cost of incident response, since forensic evidence may be destroyed before recovery begins. Security researchers analyzing the campaign have characterized its operational planning as more consistent with organized criminal activity than with opportunistic malware deployment [6].

Why AI Developer Tooling Is a High-Value Target

The campaign's deliberate selection of AI developer SDKs — including Mistral AI's TypeScript client, LiteLLM's Python abstraction layer, and Guardrails AI's validation library — suggests the group assessed AI developer tooling as high-value attack surface. Developers building LLM-integrated applications routinely execute these libraries in environments that hold API keys for AI platforms, cloud provider credentials, and access to sensitive data pipelines. A single compromised LiteLLM or Guardrails installation may yield credentials that grant access to AI workloads processing sensitive enterprise data, creating secondary exposure well beyond the developer's immediate workstation [7][8].

The targeting of security tooling (Trivy, KICS, Bitwarden CLI) in earlier campaign phases follows a parallel strategic logic: these tools run with elevated trust in CI/CD pipelines and are often configured with broad read access to secrets management systems precisely because they need to scan for secrets. Compromising a secret scanner provides a structurally privileged credential-harvesting position.

Scope and Impact Indicators

Security Boulevard's reporting aggregates the campaign's impact at over 1,800 developers potentially affected across npm, PyPI, and PHP ecosystems, with CVE-2026-45321 assigned a CVSS score of 9.6 [8][10][11]. The Hacker News tracked over 400 repositories created using stolen developer credentials, illustrating how initial workstation compromises translate into broad organizational exposure through GitHub API access [11]. Orca Security's analysis identified approximately 170 affected npm packages and multiple PyPI packages with 373 malicious versions published under legitimate namespaces during the campaign's full duration [9].

Recommendations

Immediate Actions

Organizations should immediately audit their dependency manifests for packages published in the March 19 through May 12, 2026 window from the following namespaces and libraries: the TanStack ecosystem (particularly `@tanstack/react-router`, `@tanstack/router-core`, `@tanstack/solid-router`, and `@tanstack/vue-router`), `@mistralai/mistralai`, LiteLLM versions 1.82.7–1.82.8, Telnyx versions 4.87.1–4.87.2, and Bitwarden CLI packages from April

2026 [1][2][7]. Any system that installed affected versions should be treated as potentially fully compromised. Prioritize credential rotation for cloud providers, AI platform API keys, npm tokens, and GitHub tokens confirmed to have been accessible on that system during the window of exposure [5].

Developers using Claude Code, VS Code, or GitHub-integrated workflows on systems that ran affected packages should inspect their IDE plugin directories and OS service registries for unexpected additions. The malware creates named services and install hooks that survive IDE restarts; simply uninstalling the affected package does not remove the persistence layer.

Credential revocation must be approached with awareness of the reported dead-man's switch: do not revoke npm tokens in isolation without first imaging the system and extracting the persistence payload, as token revocation may trigger a destructive wipe before forensic evidence can be collected [5]. Confirm with your incident response team that the persistence behaviors described in available analysis apply to the specific payload version recovered from your environment.

Short-Term Mitigations

Organizations should reconsider their reliance on SLSA attestation verification as a sufficient supply chain control in isolation. The Mini Shai-Hulud campaign demonstrates that SLSA Build Level 3 attestations can be produced by malicious actors who have compromised a GitHub Actions runner — attestations verify process integrity, not the integrity of the underlying code or credentials used. Complementary controls are essential: pin dependencies to commit-level hashes rather than mutable version tags, implement package-level behavior monitoring in CI/CD environments (detecting unexpected outbound network connections during builds), and validate that published packages match expected source code hashes against independently-verified build artifacts [3].

For AI developer SDKs specifically, organizations may benefit from a brief observation window — such as 48–72 hours — before upgrading widely-used LLM integration libraries, giving the security community time to surface anomalies in new releases before they propagate into production pipelines. The appropriate window will depend on each organization's risk tolerance and operational cadence.

GitHub Actions environments should be hardened to limit the blast radius of OIDC token theft. This means scoping npm publish permissions to the minimum necessary package namespaces, requiring pull request approval gates before workflows with publish permissions can execute, and implementing npm token scope restrictions that prevent a stolen token from being used to publish to packages outside the specific repository's namespace.

Strategic Considerations

The Mini Shai-Hulud campaign illustrates a structural tension in the evolution of software supply chain security: the same CI/CD automation and OIDC-based publishing trust chains that reduce friction for legitimate developers also create compound attack surfaces. As AI development tooling becomes more deeply integrated into developer workflows — with AI coding assistants reading local files, accessing cloud credentials, and executing code on behalf of developers — the attack surface available to a supply chain attacker expands correspondingly.

Organizations should incorporate AI developer tooling into their software supply chain risk inventory explicitly, treating AI SDK dependencies with the same scrutiny applied to security-sensitive libraries. The SLSA framework remains valuable for detecting many classes of supply chain attacks, but attestation verification should be paired with behavioral controls, anomaly detection in build logs, and cryptographic hash pinning to address the attack pattern demonstrated here.

CSA Resource Alignment

This incident maps directly to several existing CSA frameworks and guidance documents that provide controls addressing the attack vectors exploited by the Mini Shai-Hulud campaign.

CSA's MAESTRO framework (Multi-Agent Execution and Secure Trust for Reliable Operations) addresses agentic AI threat modeling and is directly applicable here: the malware's persistence inside Claude Code and VS Code exploits the elevated ambient access that AI coding agents require to function, turning that access against the developer. MAESTRO's infrastructure and data operations layers — covering agent authorization and credential isolation — are relevant mitigations.

The AI Controls Matrix (AICM), CSA's AI-specific extension of the Cloud Controls Matrix, addresses supply chain integrity for AI components. AICM controls in the AI Transparency and Explainability domain speak to provenance verification requirements, while controls in the AI Resilience domain address the recovery procedures relevant to a compromise of the scope described here.

CSA's DevSecOps Six Pillars framework, particularly Pillar 3 (Continuous Security Testing) and Pillar 4 (Compliance and Development Bridge), provides the procedural scaffolding for implementing the dependency scanning, behavioral monitoring, and attestation verification practices recommended above. The Six Pillars guidance on CI/CD pipeline integrity is directly applicable to defending against the GitHub OIDC token harvesting technique at the center of this campaign.

The Software Transparency guidance in CSA's corpus — including SBOM adoption and software component analysis practices — informs the dependency audit recommended as an immediate action. While SBOMs do not prevent supply chain attacks of this type, they can meaningfully accelerate the identification of affected systems once an attack is disclosed, reducing mean time to remediation across the organization.

References

[1] StepSecurity. "Mini Shai-Hulud Is Back: A Self-Spreading Supply Chain Attack Compromises TanStack npm Packages." StepSecurity Blog, May 11, 2026.

[2] Aikido Security. "Mini Shai-Hulud Is Back: npm Worm Hits over 160 Packages, including Mistral and Tanstack." Aikido Security Blog, May 11, 2026.

[3] StepSecurity. "A Mini Shai-Hulud Has Appeared: Obfuscated Bun Runtime Payloads Hit SAP-Related npm Packages." StepSecurity Blog, April 2026.

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

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

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

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

[8] Security Boulevard. "1,800 Developers Hit in Mini Shai-Hulud Supply Chain Attack Across PyPI, NPM, and PHP." Security Boulevard, May 2026.

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

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

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