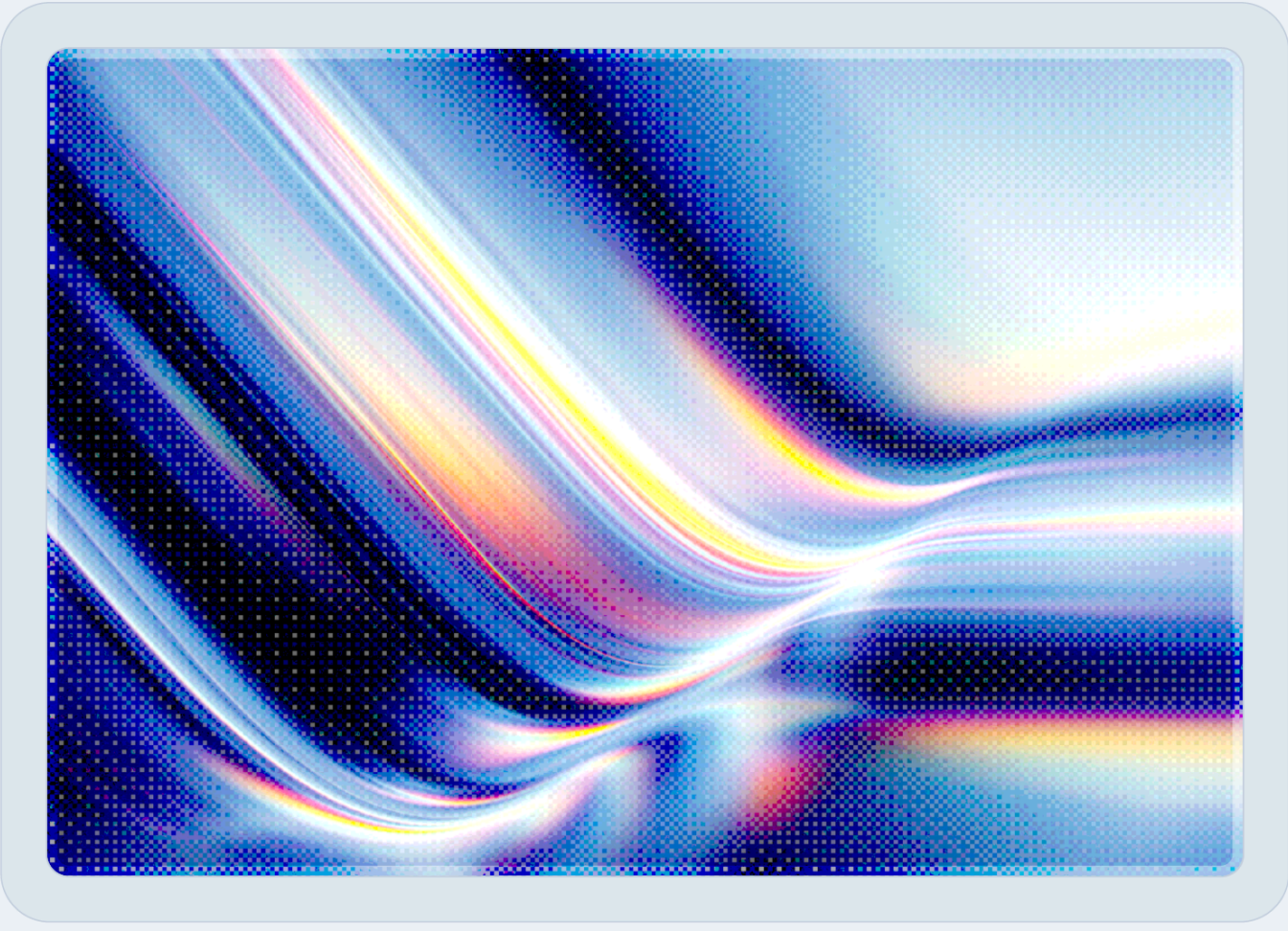
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: AI Developer npm Supply Chain Worm

Signed Malicious Packages Target TanStack, Mistral AI, and 170+ Ecosystems

2026-05-14

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- On May 11, 2026, threat group TeamPCP published 404 malicious package versions across 172 npm packages and 2 PyPI packages in under six hours, compromising major AI developer ecosystems including TanStack, Mistral AI, UiPath, OpenSearch, and Guardrails AI [1][2].
- For the first time in documented history, a malicious npm worm produced packages bearing valid SLSA Build Level 3 provenance attestations—defeating a foundational supply chain integrity control—by hijacking the legitimate GitHub Actions release pipeline rather than exfiltrating credentials externally [3][4].
- The attack, designated CVE-2026-45321 (CVSS 9.6), chained three GitHub Actions vulnerabilities: a "Pwn Request" via `pull_request_target`, cross-fork cache poisoning, and runtime extraction of OIDC tokens from runner process memory [5].
- Security researchers reportedly identified a geofenced destructive branch that — based on static code analysis — carries a one-in-six probability trigger for executing a full filesystem wipe on systems geolocated to Israel or Iran [13].
- Any developer or CI/CD pipeline that installed affected TanStack, Mistral AI SDK, UiPath, OpenSearch JavaScript, or Guardrails AI packages on or after May 11, 2026 should treat all secrets reachable from that environment as compromised and rotate them immediately [5] [7].

Background

The name "Mini Shai-Hulud" derives from Frank Herbert's *Dune*, where Shai-Hulud names the giant sandworms that travel unseen beneath the desert and surface with devastating consequence. The threat actor known as TeamPCP adopted this name for a self-replicating worm that propagates through open source package registries—invisible until it erupts through trusted publishing infrastructure.

The Shai-Hulud lineage predates TeamPCP's involvement. Earlier waves in September and November 2025 established the worm's foundational technique: after compromising one maintainer's credentials, the malware automatically enumerates every package that maintainer controls and publishes infected versions of each, using each new infection as a springboard to the next [8]. The November 2025 wave

alone reportedly exposed more than 26,000 GitHub repositories within a 24-hour window [8]. TeamPCP entered this lineage with a more aggressive operational tempo and a pattern consistent with a systematic strategy of escalating sophistication across successive campaign waves.

In March 2026, the group backdoored the axios npm package—one of the most widely downloaded JavaScript libraries in existence, with approximately 100 million weekly downloads—by compromising a maintainer account [9]. CISA issued an advisory confirming the scope of that compromise [10]. In April 2026, a fresh Mini Shai-Hulud wave targeted SAP-related npm packages, introducing obfuscated Bun runtime payloads to complicate forensic analysis [11][12]. That same month, security researchers attributed the compromise of Aqua Security's Trivy container scanner and the Bitwarden CLI npm package to TeamPCP, suggesting a pattern of targeting tooling embedded within security and DevOps pipelines [8].

The May 2026 wave represents an operational shift. Prior waves relied on stolen developer credentials—a technique that some organizations had begun to defend against through hardware security keys and short-lived token policies. TeamPCP responded by engineering an attack that bypasses credential theft entirely, instead hijacking the trusted build pipeline itself. The target selection suggests deliberate intent: TanStack is embedded in React and Vue applications throughout the JavaScript ecosystem, with @tanstack/react-router alone recording more than 12.7 million weekly downloads [1]. Mistral AI's SDK suite is the official client library for developers integrating Mistral's large language models. Guardrails AI is a Python framework specifically designed to enforce safety validation and structured output in LLM pipelines. Together, these targets represent a strike at the infrastructure through which AI applications are built and governed.

Security Analysis

The Three-Stage Attack Chain

TanStack's official postmortem documents a six-minute window—19:20 to 19:26 UTC on May 11, 2026—during which 84 malicious artifacts were published across 42 packages in the @tanstack namespace [5][7]. The attack chained three distinct vulnerabilities in GitHub Actions in sequence, each exploiting permissions or trust boundaries that the prior stage had positioned the attacker to abuse.

The first stage exploited the `pull_request_target` workflow trigger, a GitHub Actions configuration pattern documented among security practitioners as a "Pwn Request." Unlike the standard `pull_request` trigger, `pull_request_target` executes in the context of the *base repository* even when the triggering pull request originates from a fork, and it inherits access to the base

repository's secrets and write permissions by default. TeamPCP created a fork of TanStack/router and submitted a pull request designed solely to invoke this workflow with attacker-controlled code. The fork's changes were never merged; the trigger event alone was sufficient to initiate the attack chain [3][5].

The second stage exploited GitHub Actions' cross-fork cache boundary. Pull requests from forks are permitted to read cache entries written by base-branch workflows under certain scoping conditions. TeamPCP's forked workflow wrote a poisoned pnpm dependency store into the GitHub Actions cache. When TanStack maintainers subsequently merged legitimate pull requests to the main branch and the release workflow executed, it restored build tooling from this poisoned cache—silently replacing legitimate binaries with attacker-controlled code that was structurally identical to the expected toolchain [3][5].

The third stage involved runtime extraction of an OIDC token from the GitHub Actions runner's process memory via `/proc/*/mem`. GitHub's OIDC integration allows workflow runners to obtain short-lived identity tokens that can be exchanged for publishing credentials without storing any long-lived secret. TeamPCP's injected binary read the OIDC token from runner memory at the moment it was generated, then exchanged it for a valid npm publish token authenticated under TanStack's own trusted release identity [3][5]. The resulting packages were indistinguishable from legitimate releases by any inspection of publishing credentials alone — a precise statement of the attack's success against provenance-based verification.

The Defeat of SLSA Provenance Attestation

SLSA (Supply chain Levels for Software Artifacts) Build Level 3 provenance is designed to ensure that a package was built by a specific repository's GitHub Actions workflow on a certified build system, providing a cryptographic certificate to that effect. This property meaningfully raises the bar against attacks that rely on stolen npm publish credentials alone. In practice, many adopters have treated this build-origin guarantee as a sufficient indicator of overall trustworthiness — an assumption the Mini Shai-Hulud campaign directly refutes.

TeamPCP's campaign exposes a precise boundary condition in this reasoning. SLSA provenance accurately attests that a package was built by a specific repository's GitHub Actions workflow run. It does not—and by design cannot—attest that the workflow was authorized to execute, that it ran from a protected branch, that the pull request triggering it was legitimate, or that the build toolchain it invoked had not been poisoned upstream in the same workflow [3][4]. Because TeamPCP hijacked the legitimate release workflow from within, rather than presenting fabricated credentials from outside, the resulting packages carry provenance that is cryptographically valid, technically accurate as a factual record of the

build event, and misleading as an indicator of trustworthiness. Security researchers have confirmed this as the first documented instance of a malicious npm package carrying valid SLSA Build Level 3 provenance [3][4].

The practical implication for organizations that have adopted SLSA attestation verification as a supply chain control is significant: provenance narrows the attack surface by raising the bar for credential-only attacks, but it does not prevent attacks that compromise the pipeline itself. The security guarantee is narrower than some implementations appear to have assumed, based on the prevalence of SLSA adoption as a standalone control.

Worm Propagation, Credential Harvesting, and Destructive Payload

Mini Shai-Hulud earns its worm designation through its self-propagation mechanism. After the initial TanStack compromise, the malware used the stolen OIDC token to enumerate every package the compromised maintainer identity controlled, publishing infected versions of each without any additional exploitation step. Possession of a valid publish token is sufficient for lateral movement within the npm registry. Within hours of the initial TanStack infection, the worm had spread to Mistral AI's SDK suite (Mistral AI published advisory MAI-2026-002 for this incident [14]), 65 UiPath automation npm packages, OpenSearch JavaScript clients, and Guardrails AI's PyPI distribution [1][2][6].

The credential harvesting scope is deliberately broad. The malware targets GitHub tokens and Actions secrets, npm authentication tokens, AWS IAM credentials (including instance metadata role tokens), GCP service account keys, Azure managed identity tokens, Kubernetes service account tokens, HashiCorp Vault tokens, and SSH private keys [1][2][5]. Exfiltration proceeds through three redundant channels simultaneously: a typosquatting domain registered as `git-tanstack[.]com`, the Session decentralized messenger network, and GitHub API dead drops that use the stolen token to write data to attacker-controlled repositories [6]. The triple-channel architecture suggests deliberate operational resilience against network-layer blocking.

Security researchers also identified a geofenced destructive branch within the payload. On systems where locale fingerprinting—using timezone settings, locale strings, and IP-based geolocation—identifies the host as being in Israel or Iran, the malware reportedly carries a one-in-six probability — based on static analysis of the payload's targeting module — of executing a recursive filesystem wipe rather than proceeding with credential collection alone [13]. The combination of targeted geofencing, financially oriented credential theft, and a destructive payload suggests the campaign serves objectives beyond conventional financially motivated cybercrime.

Recommendations

Immediate Actions

Any organization whose developers or CI/CD systems installed affected versions of TanStack, Mistral AI SDK, UiPath automation packages, OpenSearch JavaScript clients, or Guardrails AI between May 11 and May 13, 2026 should treat their build environment as fully compromised. The first priority is credential rotation: every secret reachable from an affected install host must be revoked and reissued before investigation proceeds. This includes AWS IAM credentials, GCP service account keys, Azure credentials, Kubernetes configs, GitHub tokens, npm tokens, HashiCorp Vault tokens, and SSH keys. The window between infection and active exploitation of stolen credentials is uncertain and may already have elapsed; rotation must not wait on the completion of forensic analysis [5][7].

Security teams should audit npm package lock files, `package-lock.json` entries, and PyPI `requirements.txt` pins in every CI/CD pipeline that ran during the affected window. The npm security database and the OSSF malicious packages dataset maintain the full list of affected versions. Organizations using automated software composition analysis should verify that their scanners have ingested the most recent malicious package advisories, including CVE-2026-45321 for the TanStack ecosystem and MAI-2026-002 for the Mistral AI incident [14]. Developers who installed affected packages locally should also rotate credentials exposed to the local install context, including credentials accessible via `~/.aws/credentials`, `~/.kube/config`, `.env` files, or environment variables.

Short-Term Mitigations

The `pull_request_target` trigger is a well-documented but persistently misconfigured GitHub Actions pattern that has now enabled multiple high-severity supply chain attacks. Engineering and platform security teams should audit all GitHub Actions workflows across their GitHub organizations for any use of `pull_request_target` in combination with secrets access or publish permissions. Where the trigger cannot be replaced with the safer `pull_request` alternative, explicit conditional checks must gate secrets access on the triggering branch matching a protected base-branch pattern. The StepSecurity Harden-Runner tool provides runtime network access restrictions and can block process-level reads of credential files during CI workflow execution [3].

GitHub Actions cache poisoning via cross-fork cache reads can be mitigated by scoping cache keys to `github.ref` or `github.sha` context values, preventing fork-sourced workflows from reading cache entries written by base-branch workflows. Organizations should enforce that package publishing

workflows—particularly those using OIDC-based authentication—execute only on protected branches with manual approval gates, ensuring that no publish operation can be initiated by an untrusted pull request event.

For AI developer tooling specifically, dependency pins for packages such as Mistral AI SDK, Guardrails AI, and similar LLM infrastructure libraries should use exact version specifications with integrity hash verification (`npm ci` with lockfile enforcement; `pip install --require-hashes`). These packages frequently appear in CI/CD environments with elevated credentials, and they often execute in contexts where they can influence model inference behavior or exfiltrate prompt data.

Strategic Considerations

The Mini Shai-Hulud campaign demonstrates that SLSA Build Level 3 provenance attestation, while a meaningful raising of the bar for supply chain attacks, does not provide the scope of protection that some implementations appear to have assumed. Organizations that rely heavily on SLSA attestation verification should layer complementary defenses: runtime behavioral analysis of packages at install time, network egress filtering that restricts outbound connections during `npm install` or `pip install` execution, and isolation of package installation environments from credential-bearing contexts. Provenance can confirm where a package was built; it cannot confirm that the build toolchain itself was uncompromised.

The worm's self-propagation architecture also warrants a reassessment of the npm maintainer trust model. A package whose provenance traces to a legitimate maintainer identity may still be malicious if that identity's release pipeline was hijacked mid-workflow. Trust in a package's signing identity must be distinguished from trust in a package's contents — a distinction that most conventional software composition analysis tools do not routinely enforce at publish time.

Organizations developing or deploying AI applications should incorporate software supply chain risk into their AI risk management programs. The deliberate targeting of Mistral AI's inference SDK and Guardrails AI's safety validation framework illustrates that the AI layer of a deployment stack introduces supply chain exposure at a particularly sensitive boundary. The recovered payload demonstrates that a compromised LLM SDK can exfiltrate API keys; a malicious SDK could additionally tamper with inference requests or disable safety controls in ways that bypass application-layer monitoring. Supply chain integrity is not a peripheral concern for AI systems — it is an AI safety concern.

CSA Resource Alignment

This incident maps to several layers of CSA's AI security frameworks. Within the MAESTRO threat modeling framework for agentic AI systems, the Mini Shai-Hulud attack pattern corresponds to Layer 3 (Agent Tools and External Integrations), where dependencies on external software libraries—including AI inference SDKs and safety guardrail packages—introduce transitive risk that agents inherit when they invoke those libraries. MAESTRO's emphasis on validating the integrity of tools and external integrations before agent execution is directly relevant: runtime verification of package integrity cannot be delegated solely to registry-level provenance signals, as this incident demonstrates.

The AI Controls Matrix (AICM), as a superset of the Cloud Controls Matrix that extends coverage to AI-specific risk domains, provides applicable guidance under its Software Development Lifecycle and Supply Chain Management control families. The defeat of SLSA provenance in this campaign reinforces that controls addressing CI/CD pipeline security—including workflow permission scoping, branch protection enforcement, and ephemeral credential policies—represent core AICM implementation requirements for any organization deploying AI-adjacent developer tooling in automated pipelines.

CSA's STAR for AI program provides a mechanism for AI service providers to publish and verify their security postures through registry submissions. Organizations procuring AI developer tooling—including managed inference APIs whose official client SDKs were compromised in this incident—should use STAR for AI assessments as a baseline for evaluating provider-side supply chain security practices, including the integrity controls applied to official client library release pipelines. The May 2026 campaign demonstrates that these libraries are active targets of sophisticated, persistent threat actors, and current STAR for AI submissions may not yet fully address CI/CD pipeline integrity as an assessment domain – a gap that warrants attention as the framework evolves.

CSA's Zero Trust guidance applies directly to the CI/CD credential architecture exposed in this attack. The successful extraction of OIDC tokens from runner process memory succeeded in part because the runner process operated with broader runtime access than a strict least-privilege model would permit. Treating each build step as an untrusted execution context—with network-isolated package installation, ephemeral scoped credentials, and per-step secret access controls—reflects the Zero Trust posture that the Mini Shai-Hulud incident makes operationally urgent.

References

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

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

[5] TanStack. "Postmortem: TanStack npm supply-chain compromise." TanStack Blog, May 2026.

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

[7] TanStack. "Hardening TanStack After the npm Compromise." TanStack Blog, May 2026.

[8] Phoenix Security. "Sha1-Hulud / Shai-Hulud: Full Technical Dissection of TeamPCP's Self-Propagating Supply Chain Worm." Phoenix Security, 2026.

[9] Huntress. "Tradecraft Tuesday Recap: axios npm Supply Chain Compromise." Huntress Blog, April 2026.

[10] CISA. "Supply Chain Compromise Impacts Axios Node Package Manager." CISA Alert, April 20, 2026.

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

[12] Wiz. "Supply Chain Campaign Targets SAP npm Packages with Credential-Stealing Malware." Wiz Blog, April 2026.

[13] Upwind. "Shai-Hulud: Here We Go Again – Dissecting a Supply Chain Worm Across the TanStack Ecosystem." Upwind Blog, May 2026.

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

Further Reading

The following sources were consulted during research and are provided as supplementary reading:

- BleepingComputer. "Shai Hulud attack ships signed malicious TanStack, Mistral npm packages." BleepingComputer, May 2026.
- Endor Labs. "Shai-Hulud compromises the @tanstack ecosystem: 80+ packages compromised." Endor Labs Blog, May 2026.
- Orca Security. "TanStack and 160+ npm/PyPI Packages Compromised in Supply Chain Worm Attack." Orca Security Blog, May 2026.
- NHS England Digital. "Supply Chain Attack Affecting Numerous npm and PyPI Packages." NHS Digital Cyber Alert CC-4781, May 2026.
- SafeDep. "Mass Supply Chain Attack Hits TanStack, Mistral AI npm and PyPI Packages." SafeDep Blog, May 2026.
- VentureBeat. "Protect your enterprise now from the Shai-Hulud worm and npm vulnerability in 6 actionable steps." VentureBeat, May 2026.