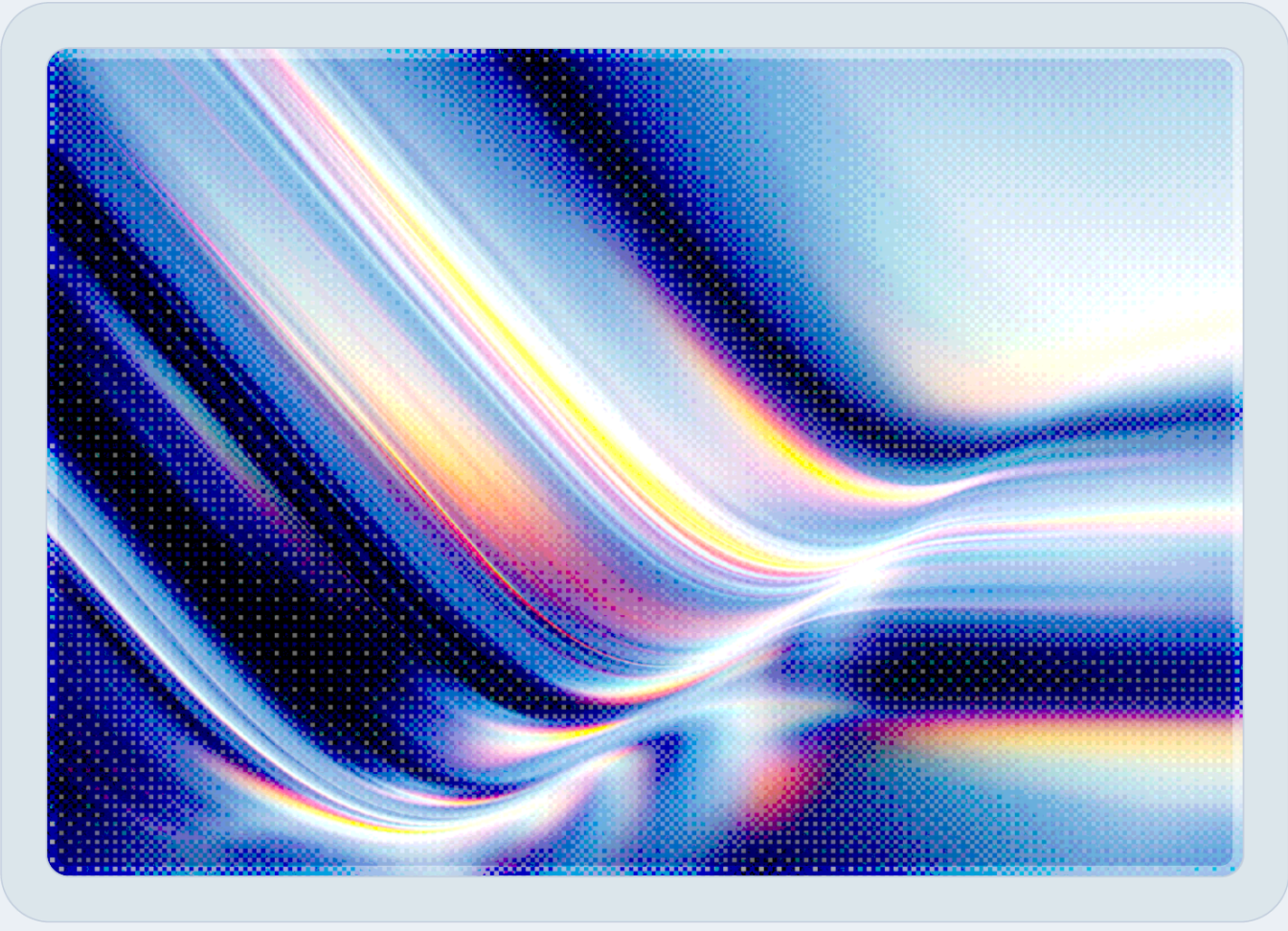
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud: TeamPCP's AI Package Ecosystem Campaign

Systematic Compromise of AI Developer Infrastructure Across npm and PyPI

2026-05-13

AI-assisted Rapid Research



CSAI

cloud security alliance®

Key Takeaways

- Between March and May 2026, a threat actor tracked as TeamPCP executed a three-phase supply chain campaign—dubbed "Mini Shai-Hulud"—that compromised over 170 npm and PyPI packages, including AI-specific libraries such as the Mistral AI SDK, LiteLLM, and Guardrails AI, as well as widely used developer tools like TanStack, UiPath, and the Trivy vulnerability scanner [1][2].
- The campaign introduced a previously undocumented persistence mechanism: malicious packages inject a modified `.claude/settings.json` file into developer repositories, exploiting Claude Code's `SessionStart` hook to re-execute the credential-stealing dropper every time an AI coding agent opens the project. To our knowledge, this is among the first documented supply chain attacks to weaponize AI coding agent configuration as a persistence vector — a technique not observed in prior supply chain attack reporting reviewed for this note [3][4].
- The worm component of Mini Shai-Hulud self-propagates autonomously—after stealing npm and GitHub tokens from a victim, it identifies all packages the victim maintains, injects its payload, increments version numbers, and republishes compromised releases, all without further attacker interaction [5].
- Organizations that installed affected package versions should assume credential compromise and initiate full secret rotation across AWS, GCP, Azure, Kubernetes, Vault, GitHub, npm, and SSH credentials from any machine that executed the malicious preinstall hooks, unless forensic investigation confirms that the preinstall hook did not execute or that exfiltration was blocked [6].
- The May 11 wave produced 373 malicious package versions carrying valid SLSA Build Level 3 provenance, demonstrating that modern supply-chain integrity attestations do not protect against attacks that compromise the CI/CD pipeline itself rather than tampering with artifacts after publication [7].

Background

The Mini Shai-Hulud campaign is the third documented generation of a supply chain worm family that threat intelligence researchers have tracked since late 2025. The original Shai-Hulud campaign, first identified by ReversingLabs, established a pattern of injecting credential-harvesting code into packages published by legitimate maintainers [8]. TeamPCP—a cloud-focused adversary that emerged in late 2025 [8]—adopted and iterated upon that model across three escalating phases in 2026, each adding capability: Phase Two introducing AI coding agent persistence, Phase Three weaponizing autonomous worm propagation.

The targeting pattern suggests a deliberate focus on developer infrastructure rather than opportunistic package selection—security scanners, AI proxies, framework libraries, and enterprise automation tooling. Compromising these categories would maximize the yield of secrets from high-privilege development environments: the developers who use Trivy to scan for vulnerabilities and LiteLLM to proxy AI model requests tend to operate in environments that hold cloud provider credentials, Kubernetes configuration files, and API keys at elevated privilege levels. The April addition of AI coding agent configuration as an attack surface represents a logical extension of this targeting pattern: as AI-assisted development becomes standard, the configuration files governing those agents become a new class of persistence target.

The campaign name derives from a signature embedded in the attacker's exfiltration infrastructure. Following each successful compromise, newly created GitHub repositories on victim accounts carried the repository description "A Mini Shai-Hulud has Appeared," a phrase borrowing from the sandworm mythology of Frank Herbert's *Dune* universe—an allusion that several security researchers noted suggests the attacker is aware of how their work is being tracked in the threat intelligence community [8].

Phase One: Weaponizing Security Tools (March 2026)

The campaign's first documented phase began on March 19, 2026, when TeamPCP exploited what investigators characterized as an incomplete credential rotation in the Aqua Security Trivy GitHub repository following a minor breach the previous month [9]. By stealing CI/CD secrets from the Trivy pipeline, the attacker deleted trusted release tags and force-pushed malicious binaries beginning with Trivy v0.69.4 [9]. Two days later, on March 21, the attacker used previously stolen secrets to compromise Checkmarx's KICS GitHub Actions workflows across two repositories—ast-github-action and kics-github-action—in a cascading pattern where each set of stolen credentials unlocked access to the next target [10].

On March 24, the campaign expanded to PyPI with the backdooring of LiteLLM versions 1.82.7 and 1.82.8. LiteLLM is a widely used AI proxy library that routes requests across model providers including OpenAI, Anthropic, Google, and others; it logs roughly 95 million monthly downloads [11]. The malicious versions harvested SSH keys, cloud credentials, Kubernetes secrets, database credentials, and environment variables before exfiltrating them to attacker-controlled infrastructure. Halcyon subsequently reported that the Trivy compromise entered an extortion phase as the "Vect" ransomware group published its first victim drawn from stolen Trivy CI/CD secrets—illustrating that credential theft from developer tooling can feed downstream ransomware operations [12].

Phase Two: SAP Ecosystem and the AI Agent Persistence Breakthrough (April 2026)

On April 29, 2026, four npm packages from the SAP JavaScript and cloud application development ecosystem received malicious updates: `mbt@1.2.48`, `@cap-js/sqlite@2.2.2`, `@cap-js/postgres@2.2.2`, and `@cap-js/db-service@2.10.1` [3][4]. These packages belong to SAP's Cloud Application Programming model and MTA build toolchain—infrastructure used extensively in enterprise SAP BTP deployments. Each compromised release added a `preinstall` hook that downloaded the Bun JavaScript runtime from GitHub Releases and used it to execute an obfuscated 11.6 MB credential stealer, with Bun serving as a fast, difficult-to-detect execution vehicle.

The April wave introduced the campaign's most novel capability: AI coding agent weaponization. After harvesting credentials and identifying all repositories the victim maintains, the payload commits two files into every accessible repository with a message forged as `chore: update dependencies` signed `claude@users.noreply.github.com`. The first file is a `.claude/settings.json` containing a `SessionStart` hook that fires automatically whenever Claude Code opens in the repository. The second is a `.vscode/tasks.json` with `"runOn": "folderOpen"` behavior that triggers in VS Code [3][4]. An organization that identifies and removes the malicious npm packages but fails to audit its repositories for these injected configuration files may find itself re-infected on the next developer session—transforming a transient dependency compromise into a persistent presence tied to the development environment on every machine that subsequently clones the repository.

The payload exfiltrates its harvest—SSH keys, AWS, Azure, and GCP credentials, Kubernetes configuration files, Vault and GitHub tokens, npm tokens, and AI tool configuration files—encrypted with RSA-OAEP-4096 and AES-256-GCM to attacker infrastructure [4].

Phase Three: The Self-Propagating Worm (May 11, 2026)

The campaign's third and largest phase launched on May 11, 2026, when TeamPCP executed a coordinated attack that compromised over 170 npm packages and 2 PyPI packages across 373 malicious published versions [5]. The scope of the May wave illustrates how the worm's self-propagating mechanism scaled the attacker's reach far beyond any manual operation could achieve.

TanStack's postmortem traced the initial entry point to a chained GitHub Actions attack exploiting the `pull_request_target` trigger combined with GitHub Actions cache poisoning and runtime memory extraction of an OIDC token from the GitHub Actions runner process [6]. A Python memory scraper reads directly from the `Runner.Worker` process on Linux-based GitHub Actions runners, extracting secrets that are masked in log output—the runner's log shows `***` while the malware reads the raw plaintext value from heap memory [7]. With the OIDC token hijacked, the attacker's release pipeline published malicious packages with valid SLSA Build Level 3 provenance, making attestation-based trust signals unreliable for detecting this class of attack [7].

Affected packages in the May wave included 42 TanStack router ecosystem packages, the official Mistral AI SDK on both npm and PyPI across three distribution channels (core SDK, Azure integration, GCP integration), 65 UiPath automation platform packages, the OpenSearch JavaScript client across four versions, and Guardrails AI on PyPI [2][13]. The malware exfiltrates harvested credentials through three redundant channels: a typosquat domain, the decentralized Session messenger network, and GitHub API dead drops using tokens stolen from victim environments [7]. On developer machines, it installs a persistent `gh-token-monitor` daemon—via macOS LaunchAgent or Linux systemd—that polls GitHub every 60 seconds and, upon detecting token revocation, attempts to execute `rm -rf ~/` before self-terminating after 24 hours [7].

Security Analysis

The Mini Shai-Hulud campaign exposes several structural vulnerabilities that extend well beyond any single compromised package.

CI/CD pipelines as the primary attack surface. Each phase of the campaign targeted the release pipeline rather than the package artifacts themselves. By compromising GitHub Actions workflows through credential theft, OIDC token hijacking, or cache poisoning, TeamPCP published malicious releases that passed signature verification and, in the May wave, even satisfied SLSA Build Level 3 attestation. This reflects a broader trend: supply chain integrity frameworks built around artifact signing and provenance attestation assume that the pipeline generating those artifacts remains trusted. When

the pipeline is compromised, attestations confirm that the CI/CD environment produced the artifact but provide no assurance about the integrity of the pipeline state at the time of production — an important limitation that attestation-based trust models must explicitly communicate [7][10].

The AI developer as a high-value credential target. TeamPCP's targeting of LiteLLM, Mistral AI, and Guardrails AI—alongside the Claude Code persistence mechanism—reflects a deliberate focus on organizations building AI-enabled applications. Development environments for AI systems frequently hold a concentration of high-value credentials, particularly in organizations that have not implemented secrets management practices for developer tooling: API keys for model providers, cloud credentials for training and inference infrastructure, and access tokens for managed AI services. Security controls applied to production AI systems may not extend equally to the developer environments where those systems are built, making the development toolchain a potential entry point into production AI infrastructure where security controls may be more mature.

Worm propagation multiplies impact without proportional attacker effort. The May 11 wave's scale —170+ packages, 373 malicious versions—was not achieved through 170 separate manual compromises. Once the worm established a foothold in a package maintainer's environment, it autonomously mapped that maintainer's entire package portfolio and published backdoored releases across all of them. This self-propagation dynamic means that the organizational exposure of a single compromised maintainer can cascade across an entire package namespace with no additional attacker action required [5].

AI agent configuration files as a new persistence class. The injection of malicious `.claude/settings.json` files into repositories represents a qualitatively new persistence mechanism. Unlike traditional malware that persists via OS-level mechanisms such as cron jobs or registry entries, this technique embeds itself in source-controlled project files that are committed, reviewed, and cloned as part of normal development workflows. A developer who cleans their local machine but clones the repository again—or a new developer who joins the project—may silently re-execute the dropper at the next Claude Code session start, provided the malicious configuration has not been identified and removed from the remote repository. Standard endpoint detection focused on process execution or file system changes may not flag this behavior because the hook executes through legitimate tooling in a context that appears normal [3][4].

The destructive fallback as deterrent against remediation. The `gh-token-monitor` daemon's `rm -rf ~/` payload may function as a deterrent against rapid token revocation — whether this reflects deliberate attacker design or incidental behavior, the practical effect is the same: an organization that discovers the compromise and immediately rotates GitHub tokens risks triggering the destructive handler on any developer machine still running the daemon [6]. This creates a response timing

dilemma—organizations must balance the speed of credential rotation against the risk of triggering destruction on compromised endpoints that have not yet been cleaned. Incident responders should treat endpoint investigation and credential rotation as coordinated, not sequential, activities.

Recommendations

Immediate Actions

Any organization that installed packages from the affected namespaces—TanStack, Mistral AI, UiPath, OpenSearch, Guardrails AI, SAP CAP (`@cap-js/*`), `mbt`, LiteLLM 1.82.7–1.82.8, or Trivy ≥v0.69.4—during the relevant windows should assume credential compromise and initiate full secret rotation across AWS, GCP, Azure, Kubernetes, Vault, GitHub, npm, and SSH credentials as a precautionary measure, unless forensic investigation confirms that the preinstall hook did not execute or that exfiltration was blocked. Rotation must be coordinated with endpoint investigation to avoid triggering the worm's destructive fallback on machines still running the `gh-token-monitor` daemon.

Before rotating credentials, security teams should scan all accessible repositories—including those the organization maintains as an npm or PyPI package publisher—for the presence of `.claude/settings.json`, `.vscode/tasks.json`, or any `preinstall` script containing Bun invocations that were not introduced through normal development workflow. These files may have been committed by the worm's self-propagation mechanism under disguised commit messages such as `chore: update dependencies`. Audit git history using `git log --all --author=claude@users.noreply.github.com` and inspect any commits from unfamiliar authors touching configuration directories [3].

Short-Term Mitigations

Organizations that rely on packages from npm or PyPI in CI/CD pipelines should implement version pinning with hash-based lockfile verification. The worm's propagation mechanism works by incrementing the version number of malicious releases—a pipeline pinned to a specific version hash rather than a version range is not automatically upgraded to the malicious release. Dependency review tooling should alert on any package version that was not present in the lockfile at the time of the prior successful build.

For teams using AI coding assistants, a formal policy governing `.claude/settings.json` and equivalent AI tool configuration files should be established and enforced through automated checks in CI. These files represent a relatively new class of security-relevant artifact whose contents can modify tool behavior at session start. Repository scanning tools should be configured to alert on unexpected changes to these files, and code review policy should require explicit security review for any modification to AI agent hook configurations. The principle of least privilege should also be applied to AI coding agent permissions: tools that do not require shell execution should not be configured with hooks that invoke shell commands.

GitHub Actions workflows in repositories that publish packages to npm or PyPI should be audited against three specific weaknesses identified in the TanStack postmortem: use of `pull_request_target` triggers that grant write access to external contributions, Actions cache configurations that could allow cache poisoning from untrusted branches, and OIDC token scope configurations that grant publication rights without job-level restrictions. StepSecurity's Harden Runner tool and similar workflow hardening utilities provide automated detection of these patterns [6].

Strategic Considerations

The Mini Shai-Hulud campaign suggests that the boundary between software supply chain security and AI security is narrowing — AI development libraries, proxies, and agent configurations are now active targets whose compromise yields the same credential classes as traditional developer tooling. Security programs that treat AI toolchain risk as separate from software supply chain risk should revisit that boundary.

Software Bill of Materials (SBOM) practices need to extend to development-environment dependencies, not just production application dependencies. The packages targeted by TeamPCP are primarily developer tools and SDK libraries; in many organizations they are not tracked in the same SBOM or dependency governance process as runtime dependencies. Extending SBOM coverage to CI/CD tooling and AI development libraries would have made the affected versions detectable through routine inventory scanning.

Finally, the campaign reinforces known limitations of attestation-based supply chain integrity. SLSA Build Level 3 provenance provides meaningful assurance that an artifact was built by a specific, hermetic pipeline—but it does not guarantee the integrity of the secrets and runtime environment of that pipeline. As CI/CD compromise becomes a routine attacker capability, supply chain security strategy must incorporate continuous monitoring of pipeline integrity, not just artifact-level attestation.

CSA Resource Alignment

The Mini Shai-Hulud campaign maps directly to threat models and control domains described in several CSA frameworks. Organizations can use these resources to structure their response and longer-term defensive posture.

CSA's **MAESTRO** (Multi-Agent Environment Security Taxonomy and Risk Observatory) framework addresses the risk of compromised tooling in agentic AI pipelines, including the threat class of malicious context injection at agent session initialization—the mechanism exploited by the `.claude/settings.json` `SessionStart` persistence technique. MAESTRO's Layer 3 (agent orchestration layer) threat models are particularly applicable to assessing the risk that compromised AI coding agent configurations pose to developer environments.

The **AI Controls Matrix (AICM) v1.0** provides guidance under its AI Supply Chain Security domain that applies directly to this campaign. AICM controls covering dependency verification, third-party component integrity, and CI/CD pipeline security should be reviewed against an organization's current posture in light of TeamPCP's demonstrated capability to compromise pipeline identity through OIDC token extraction.

CSA's **Cloud Controls Matrix (CCM)** addresses relevant supply chain risk under the Supply Chain Management, Transparency and Accountability (STA) domain and the Threat and Vulnerability Management (TVM) domain. The CCM's controls on software integrity verification (STA-09) and on automated vulnerability management processes (TVM-07 through TVM-09) are directly applicable to the challenge of detecting and responding to compromised package versions at scale.

CSA's **Zero Trust guidance** is relevant to the credential-theft objectives of this campaign. A Zero Trust architecture that enforces just-in-time, just-enough access for CI/CD pipeline identities—including restricting OIDC token scope to the minimum required for each workflow job—would limit the blast radius of the OIDC token hijacking technique at the core of the May 11 wave.

References

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

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

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

[4] Phoenix Security. "Mini Shai-Hulud: SAP CAP and mbt npm Packages Backdoored via Bun-Loaded Credential Stealer with Claude Code Persistence." Phoenix Security Blog, April 2026.

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

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

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

[8] ReversingLabs. "Team PCP's Mini Shai-Hulud tears at open-source trust." ReversingLabs Blog, May 2026.

[9] Wiz. "Trivy Compromised by 'TeamPCP'." Wiz Blog, March 2026.

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

[11] Endor Labs. "TeamPCP Isn't Done: Threat Actor Behind Trivy and KICS Compromises Now Hits LiteLLM's 95 Million Monthly Downloads on PyPI." Endor Labs Blog, March 2026.

[12] Halcyon. "Trivy Supply Chain Compromise Enters Extortion Phase as Vect Ransomware Publishes First Victim." Halcyon Blog, 2026.

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

Additional Resources

The following sources provide broader coverage of the Mini Shai-Hulud campaign and the TeamPCP threat actor. They are not directly cited in the analysis above but are recommended for organizations seeking additional technical detail and organizational guidance.

Mend.io. "Shai Hulud: SAP CAP Supply Chain Attack Via Claude Code." Mend.io Blog, April 2026.

Microsoft Security Blog. "Guidance for detecting, investigating, and defending against the Trivy supply chain compromise." Microsoft, March 24, 2026.

SANS Institute. "When the Security Scanner Became the Weapon: Inside the TeamPCP Supply Chain Campaign." SANS Blog, 2026.

Arctic Wolf. "TeamPCP Supply Chain Attack Campaign Targets Trivy, Checkmarx (KICS), and LiteLLM." Arctic Wolf Blog, March 2026.

JFrog Security Research. "Shai-Hulud: Here We Go Again – Worm by TeamPCP Hits NPM and PyPI." JFrog, May 2026.

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

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

NHS England Digital. "Supply Chain Attack Affecting Numerous npm and PyPI Packages." NHS England Cyber Alerts, May 2026.

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

CSA Labs. "Mini Shai-Hulud: Multi-Ecosystem Developer Supply Chain Attack." Cloud Security Alliance Labs, April–May 2026.

