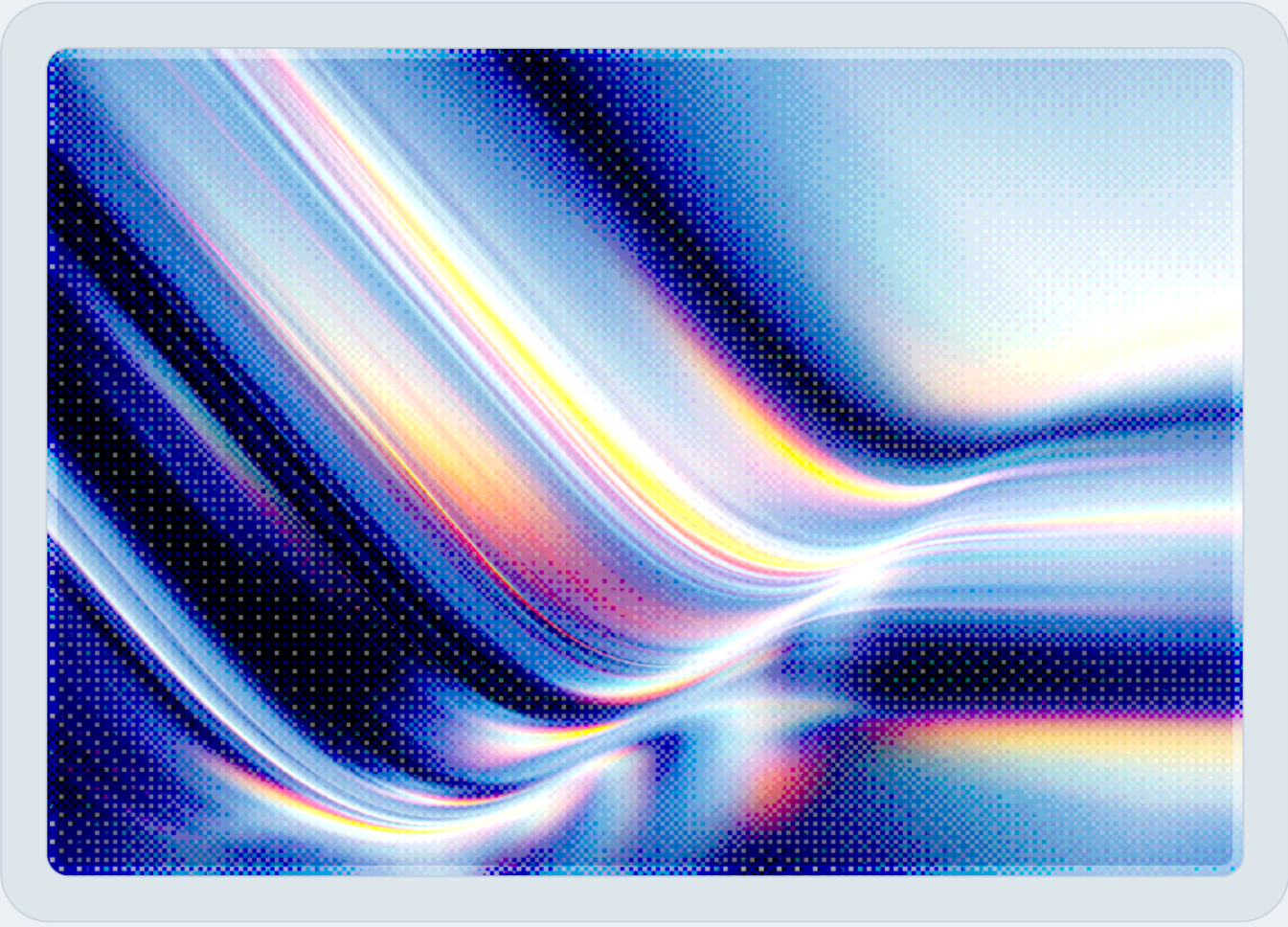
CSAI Foundation | Cloud Security Alliance

Mini Shai-Hulud Escalates: AI Packages and GitHub Targeted

TeamPCP's Worm Hits AI SDK Infrastructure, Compromises GitHub, and Spawns Copycat Variants

2026-05-20

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- On May 11, 2026, TeamPCP executed the largest wave of the Mini Shai-Hulud campaign, publishing 373 malicious package-version entries across 172 npm and PyPI packages, affecting packages with a combined download count exceeding 518 million [1][2][20].
- In what StepSecurity and Vectra AI describe as the first documented npm worm to produce packages bearing valid SLSA Build Level 3 provenance attestations, TeamPCP extracted OIDC tokens from GitHub Actions runner memory to obtain valid cryptographic signing certificates — meaning standard provenance checks would pass on malicious artifacts [2][12] [17].
- AI infrastructure packages were primary targets: the Mistral AI Python SDK, Guardrails AI, and LiteLLM — a gateway used by developers to access over 100 large language model providers with more than 95 million monthly downloads — were all compromised [1][7][21].
- On May 12–13, TeamPCP published the Shai-Hulud worm's full source code to GitHub alongside explicit instructions encouraging other actors to deploy it in new campaigns [3][4] [5]; GitHub removed the repositories, but forks proliferated, and OX Security confirmed the first copycat campaigns within days [6].
- On May 20, GitHub disclosed that an employee device was compromised via a poisoned VS Code extension attributed to TeamPCP, resulting in the exfiltration of approximately 3,800 internal repositories, which were subsequently listed for sale at $50,000 [9][10].

Background

The Shai-Hulud family of supply chain attacks has evolved rapidly since its first documented appearance targeting npm in late 2025. TeamPCP — tracked by vendors and threat intelligence platforms under the aliases DeadCatx3, PCPcat, ShellForce, and UNC6780 [10][18] — established a pattern early on: compromise widely trusted open-source tools, implant credential-stealing payloads in their build pipelines, and use the resulting package versions as vehicles for lateral movement through victims' developer infrastructure. The group's primary worm tool, CanisterWorm, provides the self-propagating

mechanism underlying the campaign [18][19]. Prior CSA reporting covered the worm's initial npm propagation mechanics and its April 2026 expansion into multi-ecosystem targeting of SAP developer packages [20].

The May 11 wave marked a qualitative shift. Where earlier campaigns operated with measured precision against specific package namespaces, this phase weaponized the trust relationships embedded in CI/CD infrastructure at a scale and speed that would challenge most security teams' standard incident response timelines. The choice of targets — TanStack for initial entry, then Mistral AI, Guardrails AI, OpenSearch, UiPath, and LiteLLM — is consistent with deliberate selection of packages deeply embedded in modern AI development workflows, suggesting targeting priorities shaped by AI infrastructure exposure rather than general developer tool prevalence. The campaign's subsequent acts — publishing the worm's source code and breaching GitHub itself — suggest either a group operating beyond containment risk calculation, or one that has already achieved its primary objectives and is transitioning to a disruption phase.

The threat actor gained initial access to TanStack's build pipeline by exploiting a GitHub Actions workflow misconfiguration in how TanStack's CI handled pull requests from forked repositories [16]. The workflow in question used the `pull_request_target` trigger with write access to the base repository's cache. An attacker-controlled pull request from a fork allowed TeamPCP to inject malicious code into a shared cache partition. When a legitimate maintainer later triggered a release workflow — a routine operation with no visible anomaly — the release process consumed the poisoned cache and executed the attacker's payload. This "Pwn Request" pattern, combined with cache poisoning across the fork-to-base trust boundary, is well-documented in CI/CD threat research but remains undermitigated across a large number of open-source projects with GitHub Actions workflows [13][14][18][19].

Security Analysis

SLSA Provenance as a False Assurance

Perhaps the most technically significant aspect of the May 11 wave was not its scale but its subversion of the software supply chain's primary integrity mechanism. After gaining control of TanStack's release workflow, the attacker's code extracted an OIDC token from the GitHub Actions runner process's memory during execution [2][17]. This token was then exchanged with Sigstore's Fulcio certificate authority to obtain a valid cryptographic signing certificate, which was used to produce a SLSA Build Level 3 provenance attestation for the malicious packages.

SLSA Build Level 3 is widely promoted as a strong guarantee that a package was built in a secure, controlled environment and that the build process was not tampered with. In this case, that guarantee remained technically accurate — the build environment was not tampered with externally; the malicious code arrived as a legitimate cache artifact from within the pipeline's own execution context. The attestation correctly recorded which workflow ran and which repository produced the artifact, while saying nothing about what content the cache had already deposited into the build [12][17]. Consumers relying solely on provenance attestation as their primary supply chain control would have had no signal from that control alone to detect the compromise. Attestation verifies the build provenance but says nothing about cache content deposited earlier in the pipeline.

CVE-2026-45321, assigned to this attack chain with a CVSS score of 9.6, covers the combined abuse of the `pull_request_target` trigger, cross-fork cache poisoning, and OIDC token extraction as an attack sequence [2]. The vulnerability affects any GitHub Actions workflow that combines these three characteristics and should be treated as a critical configuration risk regardless of whether TanStack specifically is a dependency.

AI SDK Infrastructure as a High-Value Target

TeamPCP's selection of AI-specific packages was not incidental, and the campaign's foothold in AI infrastructure predated May 11. LiteLLM had been compromised in an earlier campaign phase dating to March 2026, with versions 1.82.7 and 1.82.8 the affected releases; the attack reached LiteLLM not through a direct breach of its source repository but through its build pipeline's dependency on Trivy, an open-source vulnerability scanner that TeamPCP had previously compromised [7][8][19]. This cascading compromise — a security tool used in the CI pipeline of an AI infrastructure tool becoming the attack vector — illustrates the compound risk profile that emerges when security tooling is treated as inherently trusted within development workflows.

The significance of LiteLLM's compromise extends beyond its download count of over 95 million per month [7][21]. LiteLLM functions as a unified API gateway to more than 100 large language model providers, meaning that a single successful payload execution against an environment using LiteLLM may expose API credentials for OpenAI, Anthropic, Azure OpenAI Service, and dozens of other providers simultaneously. The malicious version 1.82.8 was particularly persistent: it installed a Python `.pth` file that executed the credential-stealing payload on every Python interpreter startup, regardless of whether LiteLLM itself was ever imported [7]. This means a developer who installed the malicious version, then uninstalled it while retaining the affected Python environment, would have continued silently exfiltrating secrets.

The Mistral AI Python SDK version 2.4.6 and Guardrails AI version 0.10.1 carried payloads that operated similarly: both downloaded a remote Python artifact on import and executed it without integrity verification [1][15]. The choice of these specific AI packages reflects accurate threat intelligence on TeamPCP's part — organizations building AI applications frequently install SDKs from model providers and validation frameworks as initial project dependencies, giving these packages broad presence in developer environments. Research note coverage from CSA Labs on May 17 first identified the AI SDK targeting pattern; this analysis extends that coverage to include the LiteLLM compromise details and the post-compromise escalation [20].

Unit 42 and SANS documented TeamPCP's compromise of security tooling including Trivy and KICS — infrastructure that served as stepping stones into AI infrastructure packages — in reporting published in late May 2026 [18][19]. The LiteLLM 1.82.7 and 1.82.8 compromise, which SANS places at March 24, 2026, reflects a systematic mapping of dependency relationships between security scanning tools and AI infrastructure packages, exploited to achieve maximum downstream impact [19].

The malicious payloads across the npm wave also exhibited a new persistence vector: they targeted AI coding agent configuration files, specifically `.claude/settings.json` [20]. By inserting malicious commands into session startup hooks within these files, compromised packages could cause AI coding assistants to silently exfiltrate additional context — including repository contents, additional secrets visible at runtime, and potentially agent session outputs — each time a developer initiated an AI coding session. This represents an AI-native persistence mechanism that most traditional endpoint security tooling is not configured to detect by default, as effective detection requires monitoring for unauthorized modifications to AI coding assistant configuration files rather than standard malware indicators.

Source Code Publication and the Copycat Proliferation Risk

On the evening of May 12, 2026, TeamPCP published the full Shai-Hulud worm source code, including detailed usage instructions, to GitHub repositories [3][4][5]. GitHub removed the repositories promptly, but the code had already been forked widely, with multiple mirrors appearing on alternative platforms. SecurityWeek reported that TeamPCP accompanied the release with explicit encouragement for other actors to use the code in supply chain attacks, along with offers of monetary compensation for successful deployments [3].

OX Security confirmed the first copycat deployments within days of the source code release [6]. The observed copycat packages — chalk-tempalte, @deadcode09284814/axios-util, axois-utils, and color-style-utils — operated as typosquatting campaigns targeting Axios users, with payloads that varied in what data was exfiltrated: some targeted cloud credentials and environment variables, one directed victim machines to participate in a DDoS botnet [6]. The C2 infrastructure used by the first confirmed

copycat cluster communicated with `87e0bbc636999b[.]lhr[.]life`, and compromised repositories created by this cluster contain the string "A Mini Sha1-Hulud has Appeared" [6]. Defenders should treat both the domain and that repository string as high-confidence indicators of compromise.

The strategic consequence of open-sourcing the worm extends beyond the immediate copycat activity. The OIDC token extraction technique is now documented and packaged for use by actors without the operational sophistication to develop it independently. While specific indicators of compromise will differ with every new variant, the underlying technique — combining `pull_request_target` misconfiguration, cache poisoning, and OIDC memory extraction — will recur until the technique's enabling conditions are systematically remediated across the open-source ecosystem.

GitHub Breach: A Campaign Culmination

On May 20, 2026, GitHub disclosed unauthorized access to its internal repositories tracing to a compromised employee device [9][10][11]. The compromise originated from a poisoned Visual Studio Code extension attributed to TeamPCP by security researchers based on infrastructure and tooling overlaps with the broader campaign [9][10], consistent with the group's established practice of using developer tooling as initial access vectors. GitHub's assessment placed the exfiltration at approximately 3,800 internal repositories — a figure GitHub described as directionally consistent with its internal investigation findings [9]. TeamPCP listed the exfiltrated source code for sale at no less than $50,000 on cybercrime forums [9].

GitHub stated that its current evidence does not indicate impact to customer-owned repositories, enterprises, organizations, or user data stored outside GitHub's internal systems, and that critical secrets were rotated as a first response priority [9][10]. The investigation remains ongoing as of this writing. Nevertheless, the breach of GitHub's own internal codebase by the same actor responsible for the supply chain campaign represents a significant compounding risk: if internal GitHub tooling, workflow code, or secrets from internal repositories are used in ways that touch the broader GitHub Actions or package ecosystem, follow-on exposure is plausible even if unconfirmed.

The VS Code extension attack vector is consistent with a pattern that TeamPCP had already demonstrated against GitHub employees: developer tooling at the intersection of trust and productivity is systematically targeted because developers frequently grant it elevated permissions without the scrutiny applied to other software. The extension-based compromise also demonstrates that TeamPCP's targeting is not limited to package registries; it extends beyond registries to encompass developer tooling such as IDE extensions — a vector that receives less systematic scrutiny than package dependencies.

Recommendations

Immediate Actions

Organizations using the identified affected packages should audit their environments immediately. Any environment that installed TanStack packages between May 11 and May 12, Mistral AI Python SDK version 2.4.6, Guardrails AI version 0.10.1, or LiteLLM versions 1.82.7 or 1.82.8 should be treated as compromised until proven otherwise. Rotating all secrets present in those environments — including CI/CD tokens, cloud credentials, API keys for AI providers, SSH keys, and Kubernetes service account tokens — should precede any other remediation work. The presence of a `.pth` file executing on Python startup should specifically be checked in affected environments, as package uninstallation does not remove this artifact.

Security teams should scan GitHub repositories for any commit messages or files containing the string "A Mini Sha1-Hulud has Appeared" and block outbound connections to `87e0bbc636999b[.]lhr[.]life` at the network layer. Any installed VS Code extensions of uncertain provenance on developer machines should be reviewed; TeamPCP's entry into GitHub's environment via an extension indicates this vector is actively exploited.

Short-Term Mitigations

Organizations maintaining CI/CD pipelines should audit all GitHub Actions workflows for use of the `pull_request_target` trigger, particularly in workflows that also use Actions cache, write tokens, or OIDC permissions. Where `pull_request_target` is required for functional reasons, the associated workflow should be reviewed against published hardening guidance to ensure cache write access is scoped appropriately and that fork-initiated workflows cannot interact with base-branch secrets [18][19]. Pinning Actions to specific commit SHAs rather than mutable version tags reduces the risk of dependency substitution within workflows.

For Python environments specifically, organizations should enumerate installed `.pth` files in site-packages directories and verify that none execute unexpected external code. Software composition analysis tooling should be configured to alert on newly published versions of AI infrastructure packages pending provenance verification, with special attention to packages in active campaigns. Organizations should consult the official LiteLLM project changelog and any available security advisories to identify the first confirmed clean version; version guidance from secondary sources should be verified against the project's own release notes before acting on it.

SLSA provenance attestation should not be treated as a sufficient standalone control for detecting supply chain compromise. The May 11 wave demonstrated that a valid SLSA Build Level 3 attestation is compatible with a payload-carrying package when the attack arrives through the build pipeline's own cache. Attestation should be combined with dependency pinning, runtime behavioral monitoring, and anomaly detection on package behavior at install time.

Strategic Considerations

The deliberate open-sourcing of the Shai-Hulud codebase signals that TeamPCP is either no longer dependent on operational security for this technique or has calculated that widespread proliferation serves its interests — potentially by creating noise that complicates attribution of its own continued activity. Either interpretation suggests that the OIDC-based CI/CD attack technique should now be treated as a commodity capability available to a broad range of threat actors, not a sophisticated targeted attack.

Organizations with significant AI infrastructure should evaluate whether their use of AI API gateways such as LiteLLM creates a single point of credential exposure. A single compromised gateway installation can yield keys for every AI provider the organization uses. Secrets for AI providers should be treated with the same rotation cadence and access control rigor applied to cloud credentials. API keys for LLM providers should be generated with the minimal scope necessary for each use case and rotated on a schedule that accounts for silent credential exfiltration as an attack scenario.

The GitHub breach warrants particular attention from organizations that rely on GitHub for CI/CD integrity. If the exfiltrated internal repositories contain any code or configuration used in GitHub's own build or actions infrastructure, there is a non-zero risk of follow-on supply chain exposure through the platform itself. Organizations should monitor GitHub's incident disclosure channel for updated findings and treat any unusual behavior in GitHub Actions workflows — including unexpected cache reads, new workflow file additions, or anomalous permission grants — as potentially related until the investigation is complete.

CSA Resource Alignment

This campaign engages multiple layers of CSA's MAESTRO threat model for agentic AI systems. At Layer 7 (AI Ecosystem), the compromise of LiteLLM, Mistral AI SDK, and Guardrails AI represents exactly the third-party dependency risk that MAESTRO identifies as a core threat surface for AI applications: the model-serving, validation, and gateway infrastructure upon which AI agents depend is

not inherently trustworthy and must be treated as a potential attack vector. At Layer 1 (Foundation Models), the credential theft targeting AI provider API keys directly threatens the confidentiality and integrity of foundation model access, enabling attackers to impersonate legitimate users with AI providers, exfiltrate prompts and outputs, or consume quota at organizational expense.

CSA's AI Controls Matrix addresses software supply chain risk through controls requiring software composition analysis, software bill of materials maintenance, and third-party component verification. The Shai-Hulud campaign demonstrates a gap between what these controls can detect in a conventional deployment — where packages are evaluated at intake — and what they can detect when the attack arrives via a legitimate CI pipeline producing a validly attested artifact. Organizations should review whether their SCA controls are positioned to detect behavioral anomalies at install time and not only at static analysis time.

The CSA STAR program's third-party risk assessment guidance applies directly to the CI/CD tooling supply chain. The compromise of Trivy as a stepping stone to LiteLLM illustrates that security-designated tools within CI pipelines carry the same supply chain risk as production dependencies and should be evaluated accordingly in STAR-aligned third-party risk programs. CSA's Zero Trust guidance for cloud-native environments — which establishes that no process, tool, or artifact should be granted implicit trust based on its position in the pipeline — applies here: the poisoned cache artifact was trusted because it was present in the build environment, not because its contents were verified.

References

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

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

[3] SecurityWeek. "TeamPCP Ups the Game, Releases Shai-Hulud Worm's Source Code." SecurityWeek, May 2026.

[4] The Register. "Malware crew TeamPCP open-sources its Shai-Hulud worm on GitHub." The Register, May 13, 2026.

[5] OX Security. "Shai-Hulud Goes Open Source: Malware Creators Leak Their Own Code to GitHub." OX Security Blog, May 2026.

[6] OX Security. "New Actors Deploy Shai-Hulud Clones: TeamPCP Copycats Are Here." OX Security Blog, May 2026.

[7] SiliconANGLE. "Forcepoint details TeamPCP supply chain attack that turned LiteLLM into a credential stealer." SiliconANGLE, May 18, 2026.

[8] Trend Micro. "Your AI Gateway Was a Backdoor: Inside the LiteLLM Supply Chain Compromise." Trend Micro Research, 2026.

[9] The Hacker News. "GitHub Breached – Employee Device Hack Led to Exfiltration of 3,800+ Internal Repos." The Hacker News, May 2026.

[10] Help Net Security. "TeamPCP breached GitHub's internal codebase via poisoned VS Code extension." Help Net Security, May 20, 2026.

[11] BleepingComputer. "GitHub investigates internal repositories breach claimed by TeamPCP." BleepingComputer, May 2026.

[12] Akamai. "Mini Shai-Hulud: The Worm Returns and Goes Public." Akamai Security Research Blog, May 2026.

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

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

[15] Phoenix Security. "Mini Shai-Hulud: TeamPCP's Self-Propagating npm Worm Hits TanStack, OpenSearch, and Mistral AI Across 170 Packages." Phoenix Security, May 2026.

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

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

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

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

[20] CSA AI Safety Initiative. "Mini Shai-Hulud: npm Worm Targets AI Developer Supply Chain." CSA Labs, May 17, 2026.

[21] The Hacker News. "TeamPCP Backdoors LiteLLM Versions 1.82.7 and 1.82.8 in Supply Chain Attack." The Hacker News, March 2026.