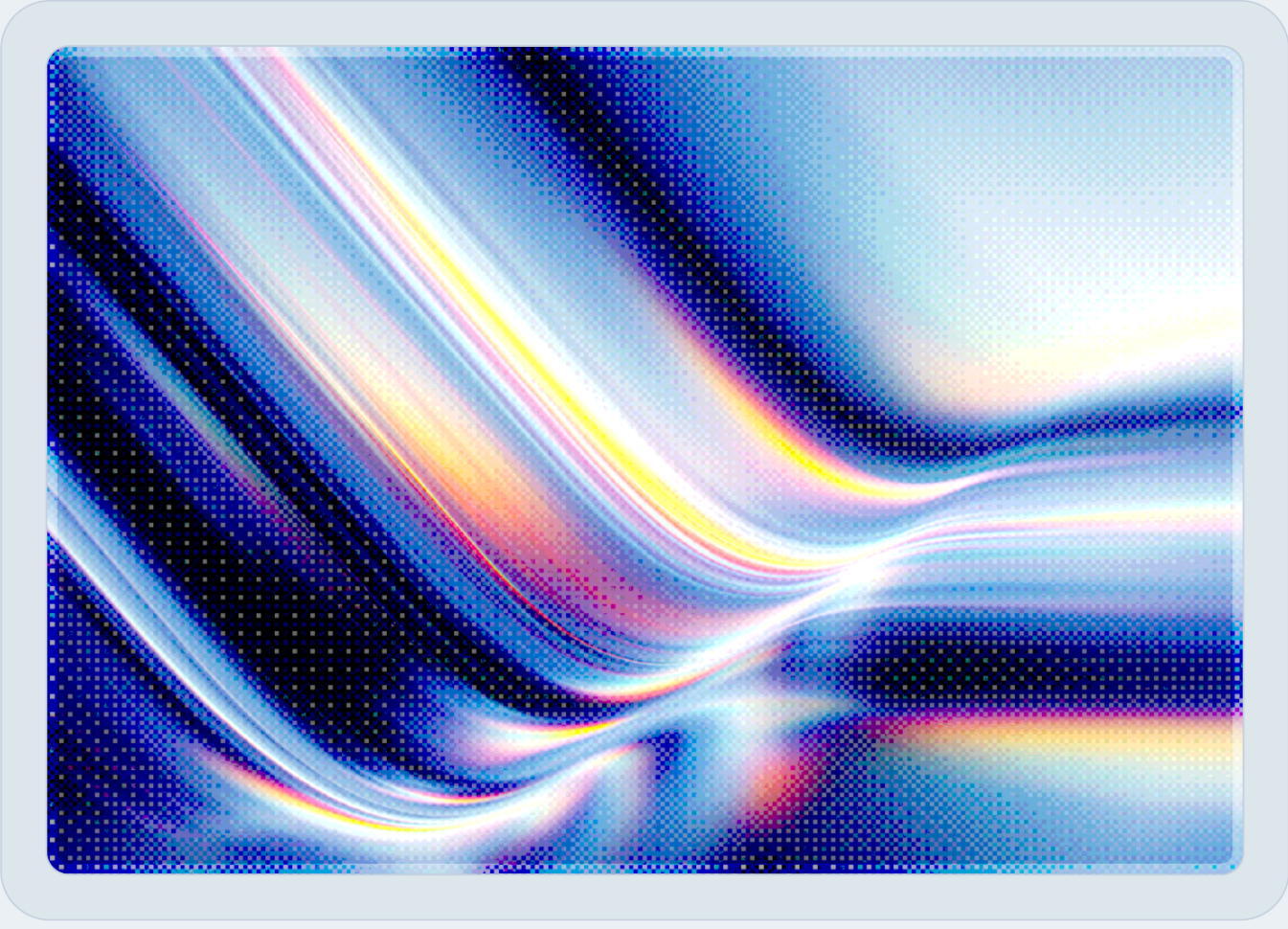
CSAI Foundation | Cloud Security Alliance

Shai-Hulud Goes Open Source: AI Package Supply Chain Under Siege

2026-05-19

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

The Shai-Hulud worm that swept the npm ecosystem in September 2025 has now entered a third, more consequential phase. Between March and May 2026, the threat actor cluster known as TeamPCP compromised the official client packages for Mistral AI, the LiteLLM model-routing library, PyTorch Lightning, Guardrails AI, and the TanStack React routing ecosystem, publishing 403 malicious versions across 172 distinct npm and PyPI packages with a cumulative download volume exceeding 518 million [1][2][3][4]. The May 11, 2026 wave, tracked as CVE-2026-45321 with a CVSSv3.1 base score of 9.6, was published in under six minutes [5].

On May 12, 2026, TeamPCP published the full source code and deployment instructions for the worm to GitHub under several likely-compromised user accounts, accompanied by a "supply chain challenge" advertised on BreachForums that promised monetary rewards to operators who could prove downstream impact [6][7][8]. GitHub removed the original repositories, but multiple forks propagated and Datadog and Reversing Labs both reported that independent groups had begun modifying the code within hours [9][10].

The salient implications for organizations consuming AI/ML packages are threefold. First, the AI developer toolchain is now an explicit, named target rather than an opportunistic one. Second, the worm's use of GitHub Actions OIDC token theft demonstrates that SLSA Build Level 3 provenance attestations and Sigstore signatures cannot, on their own, confirm maintainer authorization when the workflow that issues them is reachable through a misconfigured trigger and an orphaned commit. Signed packages from affected repositories were cryptographically indistinguishable from legitimate releases at the moment of publication [1][11]. Third, the release of the offensive framework into the public domain has compressed the time horizon for copycat campaigns from months to days, which warrants an immediate review of CI/CD trust boundaries, npm and PyPI publishing credentials, and the runtime privileges held by AI workloads that pull dependencies from public registries.

Background

The Original Shai-Hulud Wave

The Shai-Hulud worm first surfaced in September 2025 when researchers identified more than 180 npm packages whose maintainers had pushed updates containing a malicious pre-install hook [12][13]. The hook downloaded a payload that ran TruffleHog, the legitimate open-source secrets scanner from Truffle Security, against the developer machine or CI runner [14]. Any credentials it recovered — npm tokens, GitHub personal access tokens, AWS keys, GCP service account JSON, Kubernetes configurations — were exfiltrated to attacker-controlled infrastructure and, in many cases, to a freshly created public GitHub repository named "Shai-Hulud" or similar variants under the victim's own account.

The worm component used the stolen npm credentials to enumerate other packages owned by the compromised maintainer and to publish backdoored versions of those packages, repeating the cycle. Datadog Security Labs and Sysdig both characterized this as the first widely observed self-propagating supply chain worm against npm at scale [10][13]. A second wave in December 2025, labeled "Shai-Hulud 2.0" or "Sha1-Hulud: The Second Coming," compromised 796 unique npm packages and exposed an estimated 33,185 unique secrets across 20,649 scanned repositories before propagation was contained [15][16].

Mini Shai-Hulud and the Pivot to AI/ML Targets

The 2026 campaign, which researchers initially labeled "Mini Shai-Hulud" because the payload was leaner and more modular than the December 2025 version, shifted its targeting profile in a manner that is consequential for the AI/ML community. The first known wave, in late April 2026, hit four SAP Cloud Application Programming Model packages [17]. Within two weeks the campaign expanded to AI-developer-facing libraries: Mistral AI's official TypeScript client `@mistralai/mistralai` and Python client `mistralai`, the LiteLLM routing library, PyTorch Lightning, Guardrails AI's guardrail framework, OpenSearch client libraries, UiPath automation packages, and the TanStack React routing and data-fetching ecosystem [1][2][3][18].

The LiteLLM compromise, disclosed on March 24, 2026, is instructive on its own. LiteLLM is a Python package downloaded approximately 95 million times per month [19] and used by AI teams to abstract calls across OpenAI, Anthropic, Mistral, AWS Bedrock, Azure OpenAI, and other model providers. Malicious versions 1.82.7 and 1.82.8 were published using credentials harvested from the LiteLLM maintainer and remained live on PyPI for roughly forty minutes before quarantine, during which they accumulated tens of thousands of downloads [3][19]. The payload was delivered through a Python

`.pth` file, an interpreter-hook mechanism that auto-executes code every time the Python interpreter starts, without requiring an explicit import — a loading path that many dependency-scanning tools do not inspect by default. The exfiltration scope on installation included cloud credentials, SSH keys, and Kubernetes service-account tokens — the exact credential classes that an AI inference platform requires to operate [19].

The PyTorch Lightning compromise on April 30, 2026 affected versions 2.6.2 and 2.6.3 and used a different but similarly subtle execution path, embedding an obfuscated JavaScript payload inside a hidden `_runtime` directory inside the Python package and executing it through a wrapper invoked at module import [20]. Lightning is a deep-learning training framework whose dependency graph reaches into the build environments of model trainers across academia and industry.

Security Analysis

How the May 11 Wave Worked

The TanStack compromise on May 11, 2026 is the most technically significant event in the campaign because it inverted several of the controls that the supply-chain-security community has spent the past two years building. According to incident reconstructions published by Wiz, Snyk, Phoenix Security, and the TanStack maintainers themselves, TeamPCP identified an orphaned commit in a TanStack repository that remained reachable through a GitHub Actions workflow [1][5][21]. The workflow used the `pull_request_target` trigger and had OIDC trust federation configured with npm. By triggering a workflow run against the attacker-controlled commit, the operators were able to cause a GitHub-hosted runner to mint an OIDC token bearing the TanStack workflow's identity, then extract that token from the `Runner.Worker` process memory using a Python helper that scanned `/proc` for the masked secret structures the Actions runtime would otherwise scrub from logs [1][21].

That OIDC token was then exchanged at npm's federation endpoint for a full publishing credential under TanStack's identity. Because the credential was federated rather than a static token, no npm secret was technically "stolen" in the sense that incident response playbooks usually contemplate, and because the resulting publish was driven by a GitHub Actions workflow run, the released packages carried valid SLSA Build Level 3 provenance attestations, valid Sigstore signatures, and legitimate GitHub Actions signature blocks [1][11][21]. From a downstream consumer's perspective, including any consumer who had implemented npm provenance verification, the malicious versions were cryptographically indistinguishable from legitimate releases at the moment of publication.

Eighty-four malicious versions across forty-two `@tanstack/*` packages were published in approximately six minutes (19:20 to 19:26 UTC) [5]. The campaign then propagated to other namespaces via stolen npm and GitHub credentials harvested from the first wave of installs, producing the cumulative 172-package, 403-version tally reported by Mend and OX Security over the subsequent 48 hours [1][2].

The Destructive Payload

The May 2026 payload added a class of behavior the earlier waves did not exhibit. Beyond the recursive credential harvest, the worm installs a persistent process named `gh-token-monitor` as a macOS LaunchAgent or Linux systemd unit. The monitor polls GitHub's API every sixty seconds using the harvested GitHub token. On receiving an HTTP 401 or 403, which would indicate that the token has been revoked, the monitor invokes `rm -rf ~/` on the developer's machine. The daemon exits automatically after twenty-four hours [5][22].

Wiz researchers documented a geofenced branch in the payload that, when system locale or IP geolocation indicates Israel or Iran, applies a one-in-six probabilistic execution of `rm -rf /` against the root filesystem [1]; a separate branch suppresses execution entirely when the host appears to be in a Russian-language environment, a finding subsequently confirmed by independent reporting [22][23]. These behaviors push the campaign past the pure data-theft profile of earlier supply-chain worms into territory that organizations would conventionally classify as destructive malware, and they should be treated as such in incident-response severity scoring.

Why the AI/ML Ecosystem Is Disproportionately Exposed

Three structural features of the AI/ML development life cycle increase exposure relative to a traditional web application stack. The first is credential concentration. An AI inference platform needs API keys for one or more model providers, a vector store, an observability backend, and, in many production deployments, cloud credentials with broad permissions to scale GPUs on demand. A single compromise of the LiteLLM package on a developer laptop or a model-serving CI runner yields a credential set that may span the full production inference path — model-provider API keys, cloud scaling credentials, and observability tokens — in environments where these are stored on the compromised host. CSA's State of Non-Human Identity and AI Security report found that fifty-two percent of surveyed organizations use generic workload identity for their AI agents and forty-three percent rely on shared service accounts, both of which increase the blast radius of a single stolen token [24].

The second is the velocity of the AI dependency graph. Frameworks like LiteLLM, LangChain, PyTorch Lightning, Mistral's clients, and Guardrails AI release frequently; teams actively prototyping with these libraries often use floating version constraints that make it difficult to avoid automatic exposure to newly

published versions. A worm-published version that survives the registry for forty minutes can be pulled by thousands of `pip install` or `npm install` invocations in that window, including by automated training jobs whose retry behavior may pin them to the malicious release.

The third is the relative novelty of the AI tooling layer. AI-developer library maintainers who adopted OIDC federation and SLSA attestations were following current best practice. The May 11 incident exploited a misconfiguration in the workflow trigger — specifically, the combination of the `pull_request_target` trigger, a reachable orphaned commit, and OIDC federation — rather than a weakness in the signing mechanisms themselves. It demonstrated that the assurance provenance attestations provide is bounded entirely by the integrity of the workflow definition that issues them, and that a single misconfigured trigger is sufficient to inherit the supply chain's full publishing trust.

What the Open-Source Release Changes

The May 12 publication of the worm's source code, deployment instructions, and the BreachForums "supply chain challenge" with monetary incentives changes the threat model in three ways. First, the operational sophistication required to mount a Shai-Hulud-class attack has been removed; clones and modifications were observed propagating within hours of the initial drop [9][10]. Second, the framework now supports modular payload swaps, meaning future variants will not necessarily carry the destructive daemon or geofenced branches that defenders have built signatures around — the destructive and geofenced components are the most signature-rich features of the current payload and therefore the most likely to be replaced in subsequent variants. Third, the BreachForums challenge structure provides a coordination layer that — based on the independent clone propagation already observed within hours of the initial drop — appears likely to recruit a long tail of independent operators rather than consolidate into a single attributable campaign, complicating both indicator sharing and attribution-driven legal remedies.

Reversing Labs and OX Security characterized the public release as "open season" for supply-chain attacks against open-source registries and noted that the historical baseline for malicious package publishing on npm and PyPI is likely to step upward rather than return to pre-incident levels even after the immediate campaign subsides [7][10].

Recommendations

Immediate Actions (0-7 days)

Organizations that pulled any version of `@mistralai/mistralai`, `mistralai` (Python), `litellm`, `lightning` (PyTorch Lightning), `guardrails-ai`, `@tanstack/*`, or any of the packages enumerated in the StepSecurity, Wiz, and OX Security advisories between March 20 and May 14, 2026 should treat affected developer workstations and CI/CD runners as compromised and rotate all credentials reachable from those hosts [1][2][3]. This rotation should include npm publish tokens, GitHub personal access tokens and fine-grained tokens, GitHub Actions repository and organization secrets, GitHub Apps installation tokens, cloud credentials across AWS, Azure, and GCP, Kubernetes service-account tokens stored on disk, and any model-provider API keys that were present in environment variables or `.env` files.

Software composition analysis tools and registry advisories from npm and PyPI should be cross-referenced against current `package-lock.json`, `pnpm-lock.yaml`, `poetry.lock`, and `uv.lock` artifacts in every repository, with particular attention to AI/ML services that pin loosely. Snyk, Wiz, Datadog, and StepSecurity have published current IOC and affected-version lists [1][10][11][32]; Microsoft's Shai-Hulud 2.0 guidance contains a complementary detection set for the December 2025 wave that remains useful baseline reading [16].

Where GitHub Actions workflows on critical repositories use the `pull_request_target` trigger, OIDC trust federation, or any combination of workflow-issued tokens with publish-class privileges to npm, PyPI, or container registries, the workflow definitions should be audited for unreachable references, orphaned commits, and trigger paths that allow attacker-supplied SHAs to inherit the workflow identity. GitHub's own guidance on hardening `pull_request_target` is the appropriate starting point [25].

Short-Term Mitigations (7-90 days)

Publish pipelines for npm, PyPI, and container registries should be moved behind a publish gate that requires explicit human approval and a hardware-backed signing step that is not reachable from the same workflow that builds the artifact. The MAESTRO threat-modeling framework's layer covering the model-supply chain is consistent with treating the build pipeline and the publish pipeline as separate trust zones [26]; the May 11 incident illustrates why that separation should be enforced operationally rather than aspirationally.

Internal mirrors should be configured for the AI/ML dependencies an organization considers production-critical. Both npm and PyPI support pull-through caches, and organizations operating model-training or inference platforms at scale should pin to internal mirrors with a deliberate promotion process from upstream rather than installing directly from public registries during build. This does not prevent compromise of an upstream maintainer, but it interposes a quarantine window during which malicious versions can be identified before they reach production builds.

Credentials held by AI services should be moved to short-lived, federated identities wherever the cloud and platform stack supports it. The CSA State of Non-Human Identity and AI Security findings make clear that the majority of organizations remain on long-lived service-account credentials [24]; the LiteLLM and TanStack incidents are both cases where the absence of short-lived, federated credentials extended the window during which harvested tokens remained valid and usable.

Strategic Considerations

The AI Controls Matrix domain on Supply Chain Management, Transparency, and Accountability defines obligations that map directly onto the Shai-Hulud threat model, including provenance verification, third-party risk management for model and library suppliers, and incident-disclosure obligations between model providers, orchestrated service providers, and application providers [27][28]. Organizations that have not yet completed an AICM-aligned assessment of their AI supply chain should treat the May 2026 campaign as evidence that the relevant controls are now load-bearing rather than aspirational.

The CCM v4.1 controls covering software-supply-chain integrity, vulnerability management, and threat intelligence remain authoritative for the underlying cloud and platform layer [29], and STAR for AI provides an assurance mechanism by which downstream consumers can assess whether their AI service providers have attested to implementing these controls, with STAR's higher assurance tiers providing third-party validation [30]. For organizations that operate as both consumers and producers of AI packages — a population that includes most large enterprises with internal AI platforms — STAR for AI's mapping between provider obligations and consumer verification is a useful structural reference.

A separate strategic consideration is software bill of materials maturity for AI workloads. The notion of an AI Bill of Materials, which extends conventional SBOMs to cover model weights, training data lineage, and evaluation artifacts, has been discussed for several years but is not yet standard practice. The Shai-Hulud campaign affects the conventional SBOM dimension first, and an AIBOM cannot substitute for that. But the campaign's targeting of AI/ML libraries argues for accelerating both: an organization that does not currently know which versions of `litellm` or `lightning` are present in its model-serving environments cannot answer the most basic question an incident commander will ask in the next variant of this campaign.

CSA Resource Alignment

This research note maps to several active CSA frameworks and publications. The AI Controls Matrix supplies the canonical control set for AI supply chain assurance, with particular relevance for the Supply Chain Management, Transparency, and Accountability domain and the Identity & Access Management domain that governs how publishing credentials and runtime tokens are scoped [27][28]. The MAESTRO threat-modeling framework provides a layered model in which the build pipeline, the deployed model, and the orchestrated agent are distinct attack surfaces; the Shai-Hulud campaign primarily exercises the build-pipeline layer but propagates into the deployed-model and orchestration layers through stolen credentials [26].

CSA's State of Non-Human Identity and AI Security report supplies the empirical baseline against which the credential-rotation recommendations above should be sized, including the prevalence of generic workload identity, shared service accounts, and long-lived tokens in production AI deployments [24]. The Agentic AI Red Teaming Guide describes test patterns relevant to validating that a compromised dependency does not yield meaningful blast radius against an agentic system, and should be used to scope post-incident assurance exercises [31]. STAR for AI provides the consumer-side verification mechanism through which an organization can assess whether an AI service provider has attested to implementing the controls discussed here, with higher assurance tiers providing third-party validation [30].

For organizations consuming AI services that themselves depend on the affected packages, the Shared Security Responsibility Model embedded in the AICM clarifies which incident-response and disclosure obligations rest with the model provider, the orchestrated service provider, and the application provider [28]. The May 2026 incidents have been instructive in revealing where these boundaries remain ambiguous in practice and where contractual language has not yet caught up with the operational reality.

References

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

[2] OX Security. "'Shai-Hulud' Malware Hits 170+ npm & PyPi Packages." OX Security, May 2026.

[3] Trend Micro. "Your AI Stack Just Handed Over Your Root Keys: Inside the litellm PyPI Breach." Trend Micro Research, March 2026.

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

[5] Lilting Channel. "Mini Shai-Hulud hits TanStack & Mistral npm: CVE-2026-45321 (CVSS 9.6), TeamPCP campaign chain." Lilting Channel, May 2026.

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

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

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

[9] Dark Reading. "Shai-Hulud Worm Clones Spread After Code Release." Dark Reading, May 2026.

[10] Datadog Security Labs. "Shai-Hulud Goes Open Source." Datadog Security Labs, May 2026.

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

[12] Krebs on Security. "Self-Replicating Worm Hits 180+ Software Packages." Krebs on Security, September 2025.

[13] Sysdig. "Shai-Hulud: The novel self-replicating worm infecting hundreds of NPM packages." Sysdig Blog, September 2025.

[14] Unit 42 (Palo Alto Networks). "'Shai-Hulud' Worm Compromises npm Ecosystem in Supply Chain Attack." Unit 42, September 2025 (updated November 26, 2025).

[15] Datadog Security Labs. "The Shai-Hulud 2.0 npm worm: analysis, and what you need to know." Datadog Security Labs, December 2025.

[16] Microsoft Security Blog. "Shai-Hulud 2.0: Guidance for detecting, investigating, and defending against the supply chain attack." Microsoft Security, December 9, 2025.

[17] Sophos. "'Mini Shai-Hulud' supply chain attack targets SAP npm packages." Sophos News, April 2026.

[18] Hackread. "TeamPCP Used Mini Shai-Hulud Worm to Poison Over 400 npm and PyPI Packages." Hackread, May 2026.

[19] Arthur AI. "LiteLLM Supply Chain Attack: What AI Teams Need to Know Now." Arthur AI, March 2026.

[20] Socket. "PyTorch Lightning PyPI Package Compromised in Supply Chain Attack." Socket Blog, May 2026.

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

[22] Tom's Hardware. "Compromised Mistral AI and TanStack packages may have exposed GitHub, cloud and CI/CD credentials in 'mini Shai Hulud' malware infection." Tom's Hardware, May 2026.

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

[24] Cloud Security Alliance. "The State of Non-Human Identity and AI Security." CSA, 2026.

[25] GitHub Docs. "Security hardening for GitHub Actions." GitHub, 2026.

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

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

[28] Cloud Security Alliance. "Introductory Guidance to AICM." CSA, 2025.

[29] Cloud Security Alliance. "CCM v4.1: Strengthening Cloud Security." CSA, December 2, 2025.

[30] Cloud Security Alliance. "STAR for AI." CSA, 2026.

[31] Cloud Security Alliance. "Agentic AI Red Teaming Guide." CSA, May 2025.

[32] StepSecurity. "Mini Shai-Hulud is Back: A Self-Spreading Supply Chain Attack Hits the npm Ecosystem." StepSecurity, May 2026.