

Miasma Mini Shai-Hulud Hits LeoPlatform npm Packages and Git...

 socket.dev/blog/miasma-mini-shai-hulud-hits-leoplatform-npm-packages-go-ecosystem

Socket Research Team

June 25, 2026



Summarize with AI

Latest wave affects LeoPlatform/RStreams npm packages, three llxlr-published npm packages, the Verana Blockchain Go module, and GitHub Actions/developer-tool workflows.

Socket Threat Research is tracking a new supply chain attack wave tied to the Mini Shai-Hulud, Miasma, and Hades malware family. The latest activity includes malicious npm releases affecting LeoPlatform and RStreams packages, GitHub Actions workflow abuse, and a related Go module compromise involving the Verana Blockchain project. While many of the affected npm packages were published through the `czirker` account, the activity is not limited to that publisher: three additional malicious packages, `hexo-deployer-wrangler`, `hexo-shoka-swiper`, and `prism-silq`, were published by the npm user [llxlr](#).

This wave combines npm registry poisoning, `binding.gyp` install-time execution, Bun-staged JavaScript malware, GitHub dead-drop infrastructure, GitHub Actions secret theft, AI coding assistant persistence, developer-tool execution hooks, and encrypted credential exfiltration.

The campaign overlaps with recent GitHub Actions compromises that use the same operational markers, including `RevokeAndItGoesKaboom`.

The Verana finding expands the campaign beyond npm, but the execution path is not Go-native. The malicious payload is staged through source-repository configuration. In this sample, the clearest observed trigger is a VS Code folder-open task that runs `node .claude/setup.mjs`; the included Claude `SessionStart` hook points to `.github/setup.js`, which is not present in the archive, so we treat it as a nonfunctional or leftover campaign template rather than a confirmed trigger.

The campaign continues the pattern seen across recent Mini Shai-Hulud, Miasma, and Hades waves: compromise developer or maintainer credentials, plant a small execution trigger, stage a larger obfuscated payload through Bun, steal secrets from developer and CI/CD environments, and use the stolen access to spread across package registries, repositories, and trusted developer workflows.

Socket has been tracking this broader Mini Shai-Hulud, Miasma, and Hades activity across prior campaigns, including earlier coverage "[Shai-Hulud Descends to Hades: Miasma Worm Campaign Spreads with New PyPI Wave](#)" and "[Mini Shai-Hulud, Miasma, and Hades Worms Target Bioinformatics and MCP Developers via Malicious PyPI Wheels](#)".

Socket AI Scanner's analysis of `leo-aws@2.0.4`, one of the malicious packages identified in the current Miasma Mini Shai-Hulud wave, flags the compromised release as confirmed malware with multiple detections across the package contents.

The Go security team acted quickly after we notified them, promptly reviewing the report and coordinating remediation. Socket notified Verana maintainers on GitHub to alert them to the compromise.

A fast publish burst across npm packages

The malicious npm releases were published in a tight window on June 24, 2026. The affected packages are part of the LeoPlatform and RStreams ecosystems, including SDK, CLI, AWS, cron, logging, connector, and serverless packages used in data pipeline and cloud integration workflows.

The package set includes the following affected versions:

1. [hexo-deployer-wrangler@1.0.4](#)
2. [hexo-shoka-swiper@0.1.10](#)
3. [leo-auth@4.0.6](#)
4. [leo-aws@2.0.4](#)

5. [leo-cache@1.0.2](#)
6. [leo-cdk-lib@0.0.2](#)
7. [leo-cli@3.0.3](#)
8. [leo-config@1.1.1](#)
9. [leo-connector-elasticsearch@2.0.6](#)
10. [leo-connector-mongo@3.0.8](#)
11. [leo-connector-mysql@3.0.3](#)
12. [leo-connector-oracle@2.0.1](#)
13. [leo-connector-redshift@3.0.6](#)
14. [leo-cron@2.0.2](#)
15. [leo-logger@1.0.8](#)
16. [leo-sdk@6.0.19](#)
17. [leo-streams@2.0.1](#)
18. [prism-silq@1.0.1](#)
19. [rstreams-metrics@2.0.2](#)
20. [rstreams-shard-util@1.0.1](#)
21. [serverless-convention@2.0.4](#)
22. [serverless-leo@3.0.14](#)
23. [solo-nav@1.0.1](#)

This remains an ongoing investigation, and we will continue to update our findings as new information comes to light. We are tracking the full campaign on a dedicated page, with all affected artifacts added as they are identified: <https://socket.dev/supply-chain-attacks/miasma-mini-shai-hulud-supply-chain-attack>.

The install trigger: `binding.gyp`

The current LeoPlatform wave uses the “Phantom Gyp” execution pattern that has become a defining feature of newer Miasma activity. Instead of relying on a visible `preinstall` or `postinstall` script in `package.json`, the malicious packages add a `binding.gyp` file. `npm` automatically invokes `node-gyp` when this file is present. The malicious `binding.gyp` uses command expansion to execute JavaScript during the build configuration phase.

A package with no obvious `preinstall` script can still execute arbitrary code during installation if `binding.gyp` is present and invokes a shell expansion. In the LeoPlatform packages, the trigger executes the package’s replaced `index.js`, which is no longer normal library code. It is a large one-line JavaScript loader.

ROT, AES-GCM, Bun, and obfuscated JavaScript

The loader follows the Miasma/Hades pattern. The first layer uses a Caesar-style letter shift and immediate `eval()` execution. The next layer decrypts embedded AES-GCM payloads. The final payload uses JavaScript-obfuscator-style string hiding, lookup tables, and runtime reconstruction of meaningful strings.

The loader also adds or relies on Bun. If Bun is not present, the malware attempts to download or install it, then runs the main payload through `bun run`. This continues a broader shift in the campaign toward Bun-staged malware, likely because many Node.js-focused security hooks and runtime controls do not observe Bun execution with the same depth.

The high-level execution chain is:

1. `npm install` sees `binding.gyp`
2. `node-gyp` executes the embedded command expansion
3. `index.js` decodes and evaluates the first-stage loader
4. AES-GCM encrypted blobs are decrypted
5. Bun is installed or resolved
6. the main Miasma payload runs under Bun
7. the malware steals secrets, stages exfiltration, and attempts propagation

Execution flow based on one malicious package from the latest Miasma Mini Shai-Hulud wave, showing the shared payload pattern: `binding.gyp` install-time execution, Bun-staged malware, developer and CI/CD secret theft, GitHub Actions abuse, IDE and AI-agent persistence, and encrypted GitHub API exfiltration.

Credential theft targets developer and CI/CD environments

The payload is designed for environments where source code, cloud identity, package publishing, and AI coding tools overlap. The current activity shows collection logic for `.env` files, npm and PyPI tokens, GitHub tokens, Slack tokens, Twilio tokens, SSH keys, Docker authentication files, Kubernetes configs, AWS credentials, Azure credentials, GCP credentials, Vault data, shell history, CI secrets, and IDE or AI-agent configuration paths.

The payload also performs security product checks for common EDR, endpoint, and fleet tooling, including CrowdStrike, SentinelOne, Microsoft Defender, Carbon Black, Cylance, osquery, Tanium, Qualys, and others. Like earlier Miasma activity, it includes a Russian locale guard.

The credential target list is not random. It reflects a worm built to move through software supply chains. Package registry credentials allow malicious republishes. GitHub tokens allow

repository poisoning. CI/CD secrets allow cloud and production access. AI-agent configuration files allow persistence on developer machines.

GitHub Actions is a primary target

This wave heavily targets GitHub Actions. The malware searches for workflows that publish packages, especially workflows using npm publishing, yarn publishing, GitHub OIDC, or package registry tokens. In CI environments, it attempts to collect secrets directly from the runner context and from runner memory. It also uses GitHub API behavior for staging and exfiltration, including repository creation and content upload paths.

A recurring workflow template in this family is named `Run Copilot`. Its purpose is not to run Copilot. It is designed to blend in with AI-assisted development workflows while dumping GitHub Actions secrets into an uploaded artifact.

Separately, the LeoPlatform compromise included repository-level poisoning. Public reporting describes orphan `snapshot-*` branches pushed to LeoPlatform repositories, with a fake dependency-update workflow and a large `_index.js` payload. The workflow was named to look like Dependabot activity and requested GitHub Actions permissions relevant to publishing.

The important point for defenders is that this is not only an npm install problem. If the malware has a GitHub token with sufficient scope, it can alter repositories, add workflows, poison branches, and plant persistence hooks that fire later.

RevokeAndItGoesKaboom connects the LeoPlatform wave to GitHub Actions compromises

One of the strongest campaign-level markers is `RevokeAndItGoesKaboom`. This marker appears in the LeoPlatform/Miasma activity and in the `codfish/semantic-release-action` compromise documented by StepSecurity. In the `codfish` case, the malicious action searched GitHub commits for `RevokeAndItGoesKaboom` messages and used them as an operator token dead-drop channel.

The same marker now appears in GitHub commit search results associated with repositories created during the “Alright Lets See If This Works” wave. This links the npm package compromise, GitHub dead-drop behavior, and GitHub Actions compromises into the same operational cluster or tooling lineage.

codfish/semantic-release-action: adjacent compromise, same tradecraft

The codfish/semantic-release-action compromise is important context for this wave. In that incident, attackers force-pushed malicious commits and repointed version tags so downstream workflows using mutable tags executed attacker-controlled code inside GitHub Actions runners. The malicious action switched execution toward Bun and ran obfuscated JavaScript from the action context.

The same broader tradecraft appears again: Bun runtime staging, GitHub token theft, encrypted collection, GitHub API exfiltration, AI coding assistant persistence, and Russian locale checks.

One additional investigative lead is the project's workflow hardening after the compromise. A merged fix changed a validation workflow away from `pull_request_target`, while the prior workflow combined `pull_request_target` with checkout of the pull request head SHA. That pattern is a known "pwn request" risk because it can execute untrusted pull request code in a privileged base-repository context.

Commit that fixes the "pwn request" vulnerability in `validate.yml` workflow from the codfish/semantic-release-action.

Compromise of this action has a potential to cause additional cascading infections of the dependent GitHub repositories. Official GitHub numbers state that 1,442 repositories depend on this action, which should be a reason to monitor this campaign in the upcoming days.

AI coding assistant persistence continues

Miasma's AI-agent targeting remains one of its clearest differentiators. The malware plants hooks for developer tools and coding agents, including Claude, VS Code, Cursor, Gemini, Copilot-related configuration paths, and other agent or IDE ecosystems. These hooks are designed to execute the payload when a developer opens a repository, starts an agent session, or triggers a folder-open task.

This turns a poisoned repository into a delayed execution surface. A developer may clone or pull a repository after the original npm compromise has been remediated, open it in an IDE or AI coding tool, and trigger the malware locally.

This is why cleanup cannot stop at removing malicious package versions. Teams also need to audit repositories for injected configuration files, suspicious folder-open tasks, Claude or Gemini session hooks, Cursor rules, and `.github/setup.js` or `_index.js` payloads.

Go module and source-repository poisoning

Socket also identified the same payload family in a Go module/source archive for github.com/verana-labs/verana-blockchain@v0.10.1-dev.20, associated with the Verana Blockchain project. Verana is a Cosmos SDK-based Layer 1 implementation of a Verifiable Public Registry for decentralized trust ecosystems.

This finding expands the campaign beyond npm package installation. The archive contains a large obfuscated payload at [.claude/index.js](#), Bun launcher scripts at `.claude/setup.mjs` and `.vscode/setup.mjs`, and a VS Code folder-open task that executes `node .claude/setup.mjs`. The included Claude SessionStart hook points to `.github/setup.js`, which is not present in this archive. Based on this sample, the viable observed trigger is the VS Code folder-open path; normal Go module resolution or Go build logic does not appear to execute the payload.

The payload follows the same Miasma execution pattern observed in malicious npm packages: ROT-style decoding, immediate `eval()`, AES-GCM-decrypted embedded stages, Bun-staged execution, broad developer and CI/CD secret collection, GitHub Actions and OIDC abuse, encrypted exfiltration, AI/IDE hook persistence, and EDR/security tooling checks.

Unlike the npm packages, this sample does not rely on `binding.gyp`. The risk is source-repository execution: a developer who clones or opens the repository in a trusted IDE or AI coding assistant environment may trigger the payload through project configuration. This reinforces the larger campaign theme: Miasma is moving across package ecosystems by targeting developer workflows, not just package-manager install hooks.

Socket AI Scanner flags github.com/verana-labs/verana-blockchain@v0.10.1-dev.20 as known malware, identifying `.claude/index.js` as a high-confidence decode-and-eval JavaScript loader staged through source-repository configuration and a VS Code folder-open execution path.

Defensive guidance

Teams that installed any affected package version should treat the installing environment as compromised until reviewed.

Recommended response:

1. Preserve forensic artifacts before cleanup where possible.
2. Identify every developer machine, CI runner, and build container that installed affected package versions.
3. Remove affected versions and rebuild from a known-good lockfile.

4. Rotate npm, GitHub, PyPI, RubyGems, cloud, Vault, Kubernetes, Docker, SSH, Slack, Twilio, and CI/CD secrets exposed to affected environments.
5. Rotate from a clean machine, not from the potentially infected host.
6. Audit repositories for injected workflows, AI-agent hooks, `.github/setup.js`, `_index.js`, orphan branches, suspicious Dependabot-like commits, and unexplained Bun usage.
7. Review GitHub Actions runs for Bun downloads, unexpected repository creation, artifact uploads containing secret material, and GitHub API content-upload calls.
8. Review use of `pull_request_target`, especially workflows that check out pull request head code or run build/test commands on untrusted pull request content.
9. Pin GitHub Actions to immutable full-length commit SHAs where possible and monitor for tag drift.
10. Restrict npm trusted publishing and GitHub OIDC permissions to workflows and branches that require them.

Indicators of compromise

Mini Shai-Hulud, Miasma, and Hades affected packages

We are tracking the full campaign on a dedicated page, with all affected artifacts added as they are identified: <https://socket.dev/supply-chain-attacks/miasma-mini-shai-hulud-supply-chain-attack>

SHA-256 hashes

- Confirmed LeoPlatform/RStreams set — `binding.gyp`:
32d1bc728d8e504952083a6adc488c309a401c7df4dc8f47b382ce32e4aebe21
- `leo-logger@1.0.8` — `index.js`:
57ba86f6f0caaa580c1dccdf4ed7873d1470e5ea2f8e9ca7a989dc04899f13c0
- `leo-logger@1.0.8` — `package.json`:
4a0aa78757958683155a7b9289427fb829abcaad1bf5ee6399eb73e8409b0bc11
- `leo-sdk@6.0.19` — `index.js`:
026588d39b7c650b5c0dfbba6c6fcc0e7ec8e3b72ba8639012e7f71c708f2c3b
- `leo-auth@4.0.6` — `index.js`:
df9ea0c71574e11c93141ad2f018a63a5375cd6d69ca2f744732ad7814170657
- `leo-aws@2.0.4` — `index.js`:
1a3b9ed0b377f56f49b9a703612cf45e86ab7d100587e1e7a476d809fe337a8c
- `leo-sdk@6.0.19` — npm tarball:
f565988f281bf77bcad26ea7f543617e53da4b62f5df63d4f7a89bae1729cf81

- leo-auth@4.0.6 — npm tarball:
a934a5bcf692b9d01e8129bf264be23809dfee464df471d75a9f3fa1bcede343
- leo-aws@2.0.4 — npm tarball:
f7c47be306351ffacd46584d2067f7be676dbfe17cd89ab4880632decfe18f3d
- leo-cli@3.0.3 — npm tarball:
3da2ca129c9920d9acd2e3477aee8f46b5a5f0e9537ad6e7b6ab1df1007adad1

File and package structure indicators

- binding.gyp added to packages that previously did not require native build behavior
- index.js replaced with a very large single-line obfuscated payload
- new bun dependency in package.json
- _index.js payloads in GitHub repositories
- .github/setup.js payloads in poisoned repositories
- .claude/settings.json
- .claude/setup.mjs
- .gemini/settings.json
- .cursor/rules/setup.mdc
- .vscode/tasks.json with folder-open execution behavior
- suspicious node-gyp rebuild activity in packages that should be pure JavaScript

Campaign strings

- Alright Lets See If This Works
- RevokeAndItGoesKaboom
- TheBeautifulSandsOfTime
- thebeautifulmarchoftime
- thebeautifulsnadsoftime

Go Source-Repository Artifacts

SHA-256 hashes

- verana-blockchain-v0.10.1-dev.20.zip:
b3e217f4354e8a4383038b99b0bcaefff191a79df58e7a1f2355a79aac2faf13
- .claude/index.js:
15b415ae41df72acf1f7e9e67569531d41dee62d089d34b4c0fab0c7fe5cc14f
- .claude/setup.mjs:
6cb3fc3650355973b8a1ed86619a3f412fb0700f29c1c3a736cada4c2c76a9f7

- .vscode/setup.mjs:
6cb3fc3650355973b8a1ed86619a3f412fb0700f29c1c3a736cada4c2c76a9f7
- .claude/settings.json:
6a861a479f45fe53f067091414332248bc027ffc396116811d12e57a6ff71250
- .vscode/tasks.json:
927387d0cfac1118df4b383decc2ea6ba49c9d2f98b47098bcbcbba1efc026e1f
- decoded first-stage JavaScript:
1a0e1daea87cab5610a3cc2aa72e7c6f1abfe55959a156368bcfa6585fa6ce
- decrypted Bun bootstrap payload:
ceff7c51d70832c3ec8dd2744b606a23b3c924ef664ae23439b9b742ea154108
- decrypted main payload:
9f93d77d32833a515bc406c46da477142bb1ac2babeecb6aa42f98669a6db015

Subscribe to our newsletter

Get notified when we publish new security blog posts!

Enter your email