

Shai-Hulud strikes again: CHAINDROP worm hits 400+ npm packages

 elastic.co/security-labs/shai-hulud-chaindrop-npm-supply-chain

Elastic Security Labs

August 6, 2026

Elastic Security Labs identified the return of Shai-Hulud. Attackers compromised the keyv maintainer and deployed CHAINDROP, a worm that uses stolen npm credentials to backdoor co-owned packages totaling over 1.3 billion monthly downloads.

5 min read [Threat Intelligence](#), [Malware Analysis](#)

On August 4, 2026, Elastic Security Labs identified a new Shai-Hulud campaign targeting the maintainer of keyv, a widely used key-value storage library. The attackers trojanized the monorepo and embedded a self-propagating worm called CHAINDROP that uses stolen npm credentials to automatically backdoor every other package the maintainer had publish rights to. The reach of this compromise is significant: keyv alone received over 600 million downloads last month, with related packages compounding the exposure: flat-cache near 580 million, cacheable-request at over 137 million, cacheable at over 30 million, and cache-manager at over 16 million monthly downloads.

Our team was alerted via Slack notification around 5:39 AM EST. We received a hit from a previously released supply-chain monitoring [project](#) used to detect malicious npm packages. At the time of this writing, over 400 unique npm packages have been compromised.

@timestamp	package.name	service.type	supply_chain.verdict.label	supply_chain.verdict.description	url.full	supply_chain.verdict.analysis
Aug 4, 2026 @ 05:20:55.747	@cacheable/utils	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/@cacheable/utils/v2.5.1	Verdict: malicious
Aug 4, 2026 @ 05:20:27.194	cache-manager	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/cache-manager/v7.2.10	This package diff exhibits multiple, unambiguous indicator... Verdict: malicious
Aug 4, 2026 @ 05:19:21.848	file-entry-cache	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/file-entry-cache/v11.1.6	This package diff shows textbook npm supply-chain malware ... Verdict: malicious
Aug 4, 2026 @ 05:13:26.727	flat-cache	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/flat-cache/v6.1.24	This package diff shows textbook indicators of a supply ... Verdict: malicious
Aug 4, 2026 @ 05:12:57.680	@cacheable/node-cache	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/@cacheable/node-cache/v3.1.2	This package diff shows extremely strong evidence of a ... Verdict: malicious
Aug 4, 2026 @ 05:12:08.779	cacheable	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/cacheable/v2.5.1	This package diff shows the classic signature of a supply ... Verdict: malicious
Aug 4, 2026 @ 04:38:11.354	keyv	npm	malicious	LLM found one or more indicators of compromise	https://www.npmjs.com/package/keyv/v6.0.0	This diff shows clear, severe indicators of a supply chain... Verdict: malicious
						This package diff shows unmistakable indicators of a suppl...

Supply Chain Monitor initial notifications

How the CHAINDROP worm works

Execution is triggered via a preinstall hook in package.json. This abuses a legitimate npm feature that will run arbitrary commands before a package is installed, requiring no further interaction from the victim after the installation.

```
...
"scripts": {
  "build": "tsdown",
  "preinstall": "node setup.mjs",
  "lint": "biome check --write --error-on-warnings",
  "lint:ci": "biome check --error-on-warnings",
  "test": "pnpm lint && vitest run --coverage",
  "test:ci": "pnpm lint:ci && vitest --run --sequence.setupFiles=list --coverage",
  "clean": "rimraf ./node_modules ./coverage ./test/testdb.sqlite ./dist"
}
```

Preinstall hook in package.json

Every subpackage in the keyv repo was backdoored with a dropper (setup.mjs) that delivers the same underlying payload. The malware is cross-platform, targeting Linux, macOS, and Windows platforms, giving it broad impact.

The payload appears under two filenames: `Math_Symbol.js` in packages compromised directly from the `keyv` monorepo, and `math_init.js` in packages trojanized during worm propagation to other maintainers. Both files share the same SHA-256 hash, making the filename a reliable indicator of infection generation. Worm-generated commits can also be identified by the author name `claude` and the commit message `chore: update config`.

This PC > Local Disk (C:) > tmp > keyv-6.0.0 > core > keyv >

Name	Date modified	Type	Size
src	8/4/2026 4:54 AM	File folder	
test	8/4/2026 4:54 AM	File folder	
LICENSE	8/4/2026 4:54 AM	File	1 KB
Math_Symbol.js	8/4/2026 4:54 AM	JavaScript File	711 KB
package.json	8/4/2026 4:54 AM	JSON File	2 KB
README.md	8/4/2026 4:54 AM	MD File	50 KB
setup.mjs	8/4/2026 4:54 AM	MJS File	30 KB
tsconfig.json	8/4/2026 4:54 AM	JSON File	1 KB
tsdown.config.ts	8/4/2026 4:54 AM	TS File	1 KB
vitest.config.ts	8/4/2026 4:54 AM	TS File	1 KB

Trojanized files

There are two other execution paths that can be triggered by users using Claude Code or VS Code. The threat actor has placed a `SessionStart` hook in `settings.json` under the Claude directory that will run `node .claude/setup.mjs` if a new Claude session is started. For VS Code, a `folderOpen` task is used under `tasks.json` that will run `node .vscode/setup.mjs` when an infected repository is opened. When a GitHub App token is among the stolen credentials, the worm extends this further by committing malicious hooks to up to 50 branches per accessible repository, injecting `.claude/settings.json` and `.vscode/tasks.json` into each. This means a developer can still be infected by just opening the repository.

The dropper (`setup.mjs`) checks if the JavaScript runtime (`bun`) is already installed. If not, the dropper will detect the platform/architecture of the machine and download `bun v1.3.13` directly from the official release [page](#). It will extract `bun`, then use it to execute the payload (`Math_Symbol.js`). After execution, the `bun` temporary directory is deleted to cover its tracks.

The payload is heavily obfuscated at 711 kilobytes, employing control-flow flattening with a string encoding scheme using Base91. The payload contains strings with Dune-themed references and has similarities to previous [Shai-Hulud campaigns](#).

`laza`, `kanly`, `ghola`, `mentat`, `lasgun`, `sietch`, `fedaykin`, `tleilaxu`, `sandworm`, `sardaukar`, `ornithopter`, `navigator`

The payload has a component called `collector` that functions as a credential harvester, scanning over 300 unique patterns across many credential stores found on a developer machine. The more notable targeting is aimed at AI tooling credentials such as Anthropic, Claude, Codex, Cursor, OpenAI, and Gemini. The malware also targets cloud provider credentials from AWS, GCP, Azure, and Alibaba Cloud. Many other credentials are targeted, such as GitHub (PATs, JWT, session tokens), HashiCorp Vault tokens, SSH private keys, Kubernetes service account tokens, and npm tokens.

```

25131     "~/cache/helm/repository/*",
25132     "~/config/helm/repositories.yaml",
25133     "~/config/openai/auth.json",
25134     "~/openai/auth.json",
25135     "~/anthropic/auth.json",
25136     "~/config/anthropic/auth.json",
25137     "~/anthropic/.env",
25138     "~/claude/credentials.json",
25139     "~/claude/settings.json",
25140     "~/cursor/config.json",
25141     "~/openclaw/openclaw.json",
25142     "~/config/opencode/opencode.json",
25143     "~/codex/auth.json",
25144     "~/hermes/.env",
25145     "~/gemini/.env",
25146     "~/tccli/*",
25147     "~/hcloud/*",

```

AI tooling targeted by credential stealer

After the credentials are stolen, they are gzip-compressed and then encrypted with a randomly generated AES-256-GCM key. That AES key is then RSA-encrypted with the attacker's hardcoded public key. Only the attacker who holds the corresponding private key can decrypt the data.

CHAINDROP does not hardcode a C2 domain; instead, it queries an Ethereum smart contract at address 0xE1f2395ee43e45A1556EC6438a88c31B83493103 to retrieve the current exfiltration endpoint at runtime. It uses multiple RPC providers as fallbacks. The practical effect is that the attacker can rotate C2 infrastructure by updating the smart contract without touching the payload. At the time of our detonation runs, we observed the following dead-drop domains (awqhnjewqjkl[.]icu, npm-cache[.]com). If the contract discovery fails, CHAINDROP searches GitHub commit history for a cryptographically signed marker (thebeautifulmarchoftime) and accepts a C2 domain only after validating it against an embedded RSA public key.

If these two previous paths fail, CHAINDROP will exfiltrate the data via a public GitHub repository based on Dune names using the compromised victim's account with a description: Shai-Hulud: Here We Go Again.

The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with 'main' selected, '1 Branch', and '0 Tags'. A search bar says 'Go to file'. There are buttons for 'Add file' and 'Code'. Below this, a commit summary shows a commit by a user (6242ba4) 2 hours ago with 3 commits. The file list shows 'results' (added 2 hours ago) and 'README.md' (initial commit, 2 hours ago). The README content is displayed below, featuring the title 'atreides-futar-498' and the description 'Shai-Hulud: Here We Go Again'.

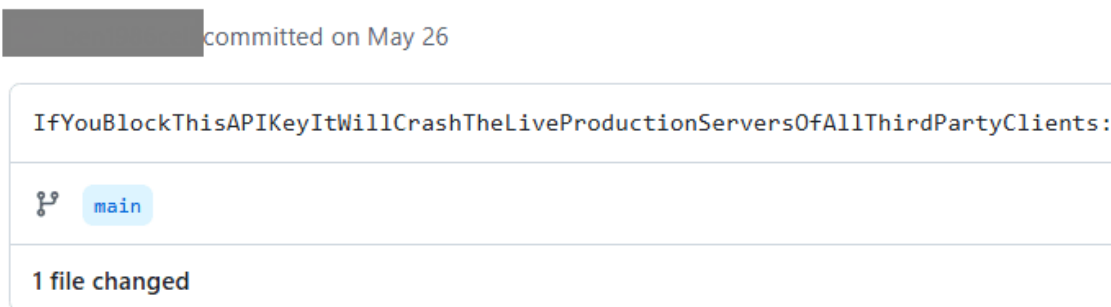
GitHub repo propagation with Shai-Hulud: Here We Go Again

The worm component activates when the credential sweep turns up an npm token, but only if that token meets two specific requirements: package write permissions and the ability to publish without two-factor authentication (bypass_2fa). Once a qualifying token is found, the worm enumerates every package the victim has publish rights to. For each writable package, it performs the following steps:

1. Downloads the latest published tarball from the npm registry
2. Extracts it to a temporary directory
3. Injects the malicious components (math_init.js/Math_Symbol.js, setup.mjs)
4. Patches the package.json to add the preinstall hook and bumps the patch version
5. Re-packages and publishes back to the npm registry

In this campaign, the commit message contains the stolen GitHub token alongside the string `IfYouBlockThisAPIKeyItWillCrashTheLiveProductionServersOfAllThirdPartyClients`. The wording is an attempt to intimidate developers not to revoke the GitHub token.

Commit 8f75879

 committed on May 26

`IfYouBlockThisAPIKeyItWillCrashTheLiveProductionServersOfAllThirdPartyClients:`

 main

1 file changed

Commit intimidation string

Elastic Defend's protection and detection coverage

Elastic Defend has coverage for this supply-chain compromise through the following rules:

Trojanized npm package install running its dropper script

- [Node.js Pre or Post-Install Script Execution](#)
- [Elastic Defend Alert from Package Manager Install Ancestry](#)

Node.js dropper shelling out to download and extract the bun runtime

- [Suspicious Windows Powershell Arguments](#)
- [Suspicious Curl Execution via NodeJS](#)
- [Curl or Wget Spawned via Node.js](#)

Bun executing the obfuscated JavaScript payload and reaching out to infrastructure

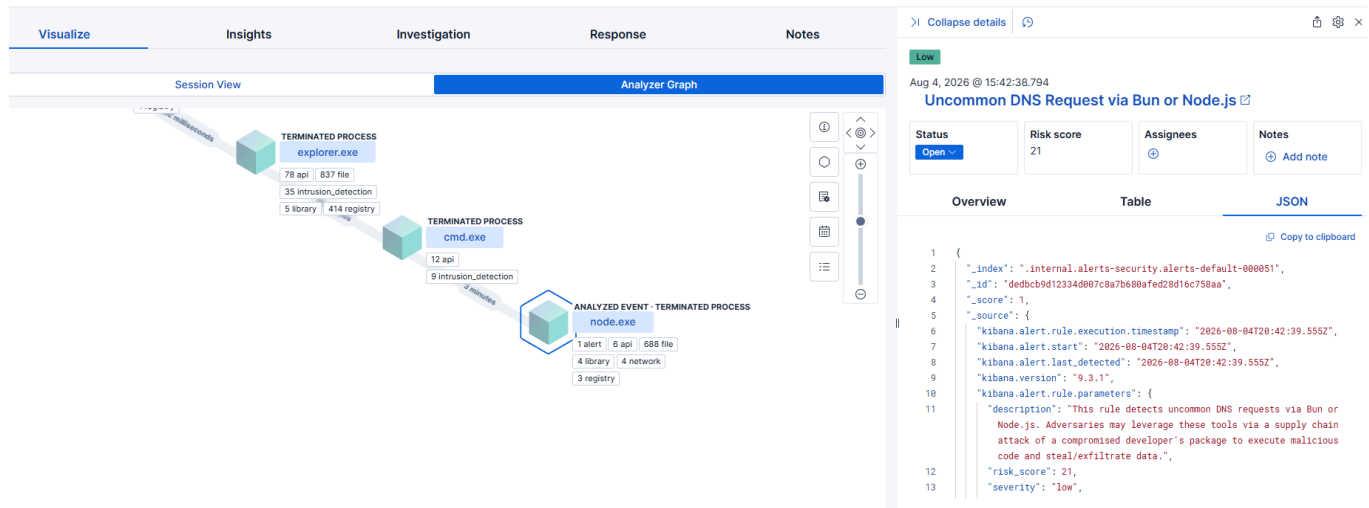
- [Uncommon DNS Request via Bun or Node.js](#)
- [DNS Request to Crypto/DHT Services](#)
- [Suspicious Instance Metadata Service \(IMDS\) API Request](#)

Staged script execution followed by deletion of the temporary directory

[Node Script Execution and Immediate Deletion](#)

Enumeration/discovery

[System Owner/User Discovery Linux](#)



Elastic Defend - Uncommon DNS Request via Bun alert

Hunting queries

The following hunting queries can be used to find this associated activity:

```
FROM logs-endpoint.events.process-*
| WHERE process.name IN ("node", "node.exe")
| WHERE process.command_line IN ("node setup.mjs", "node setup.mjs")

FROM logs-endpoint.events.process-*
| WHERE process.name IN ("bun", "bun.exe")
| WHERE process.command_line LIKE "*Math_Symbol.js*" OR process.command_line LIKE "*math_init.js*"

FROM logs-endpoint.events.network-*
| WHERE process.name IN ("bun", "bun.exe")
| WHERE dns.question.name IN ("go.getblock.io", "eth.llnarnpc.com", "npm-cache.com", "eth-mainnet.nodereal.io", "awqhnjewqjkl.icu")
)
```

Recommendations: How to respond to the Shai-Hulud compromise

- Don't pull package updates immediately; add a soak period where new versions sit before your builds adopt them. This gives the community time to catch compromises before they reach your pipelines and developer machines (previously mentioned [Navigating the Shai-Hulud Worm 2.0: Elastic's updated response to npm supply chain compromise](#) and [How we caught the Axios supply chain attack](#))
- Revoke all GitHub tokens (PATs, session tokens) for any impacted machines

- Revoke and regenerate npm tokens found on affected machines, especially automation tokens with write permission and `bypass_2fa`
- Check your GitHub repositories for unauthorized commits with the message `chore: update config from claude@users.noreply.github.com`
- Rotate any impacted credentials from cloud services, Kubernetes configs, Vault tokens, AI tooling, CI/CD secrets
- Upgrade to npm 12 or later, which blocks `preinstall` hooks by default
- Enable 2FA on all npm accounts and avoid automation tokens with `bypass_2fa: true` where possible

Indicators of Compromise

The following observables were discussed in this research.

Observable	Type	Name	Reference
9fc2570b7cef51c1b8df116d144d11ff4096357be7d2c4c6367cfc2509cf1bcc	SHA-256	Math_Symbol.js	
fd3ca4007b225fdf8de7af4345a19179d5efa8c4bb9205f88cda806e5684b1eb	SHA-256	setup.mjs	
54dc7ea54a1317cca0e890a2770630cf7fa6c97813e0cb9d2caa93012b350668	SHA-256	setup.mjs	
npm-cache[.]com	domain		
0xE1f2395ee43e45A1556EC6438a88c31B83493103	Ethereum contract		C2 resolver
npm-cache[.]com	domain-name		C2 server
awqhnjewqjkl[.]icu	domain-name		C2 server

Jump to section

- [How the CHAINDROP worm works](#)
- [Elastic Defend's protection and detection coverage](#)
- [Trojanized npm package install running its dropper script](#)
- [Node.js dropper shelling out to download and extract the bun runtime](#)
- [Bun executing the obfuscated JavaScript payload and reaching out to infrastructure](#)
- [Staged script execution followed by deletion of the temporary directory](#)
- [Enumeration/discovery](#)
- [Hunting queries](#)
- [Recommendations: How to respond to the Shai-Hulud compromise](#)
- [Indicators of Compromise](#)