QUIMERA X
INTELLIGENCE

TLP:GREEN

RELATÓRIO DE

INTELIGÊNCIA

CYBER THREAT INTELLIGENCE

Cliente

Shai-Hulud August Wave - NPM Supply Chain Attack

Data
4 de agosto de 2026

Classificação
TLP:GREEN

São Paulo - SP
Brasil

Conteúdo

Lista de Figuras

Este documento foi elaborado com base na necessidade e das informações compartilhadas com o time de CTI do QuimeraX, com referência ao seu contexto, escopo e limitações.

Os resultados nele contidos são desenvolvidos a partir de nossos métodos, processos, técnicas e know-how. O cliente é o único proprietário destes resultados.

Este documento é confidencial em sua forma e conteúdo.

1. Sumário Executivo

Em **4 de agosto de 2026**, a equipe de ***Cyber Threat Intelligence*** (***CTI***) do **QuimeraX** consolidou a análise da nova onda da campanha de cadeia de suprimentos conhecida como **Shai-Hulud**, que comprometeu inicialmente o pacote **`keyv@6.0.0`** e, em seguida, propagou-se automaticamente por centenas de outros pacotes *npm* pertencentes a diferentes mantenedores e organizações. O primeiro pacote com carga maliciosa confirmado foi publicado às **09:35 UTC**, após alterações maliciosas no repositório **`jaredwray/keyv`** e da introdução de mecanismos de execução automática por **`preinstall`**.

Com base em eventos do registro *npm* e em inventários públicos correlacionados pela equipe **QuimeraX**, o maior levantamento consolidado nesta entrega atribui à campanha **444 nomes de pacotes** e **2.234 versões envenenadas**, distribuídas por pelo menos **12 grupos organizacionais**. Outros inventários contemporâneos chegaram a **434 pacotes / 1.381 versões** e mais de **428 pacotes / 1.700 versões**. Essas diferenças refletem horários de coleta, remoções rápidas do registro, republicações múltiplas e metodologias distintas. Os números devem ser tratados como fotografias de um incidente ainda em evolução, e não como contagem final consolidada.

A cadeia técnica principal observada foi:

```
pacote comprometido
    - script preinstall executa setup.mjs
    - loader baixa Bun 1.3.13
    - Bun executa Math_Symbol.js ou math_init.js
    - coleta credenciais e segredos
    - compromete GitHub, npm, nuvem e CI/CD
    - altera e publica novos pacotes npm
    - cria ou modifica repositórios e workflows GitHub
```

A carga buscava credenciais do *npm* e do GitHub, credenciais de AWS, Kubernetes e Vault, segredos de *workflows*, chaves SSH, arquivos de ambiente e *tokens* presentes em sistemas de desenvolvimento e *runners* de CI/CD. Ela também implementava um *worm* de publicação: enumerava pacotes aos quais a vítima tinha permissão, alterava *tarballs*, adicionava o *hook* malicioso, incrementava versões e republicava no registro.

A principal cadeia transitiva identificada foi:

```
eslint
    file-entry-cache@11.1.6
        flat-cache@6.1.24
            keyv@6.0.0
```

Isso tornou a campanha especialmente perigosa para ambientes de desenvolvimento e CI: um projeto não precisava declarar keyv diretamente para resolver uma versão maliciosa. A exposição exata depende do *lockfile*, do horário da instalação, dos intervalos *semver* e de o gerenciador ter executado *lifecycle scripts*.

Conclusão operacional: qualquer sistema que tenha executado a instalação de uma das versões afetadas com *scripts* habilitados deve ser tratado como **potencialmente comprometido**, e não apenas como possuidor de uma dependência vulnerável. A resposta precisa incluir análise forense, reconstrução do ambiente e rotação de todas as credenciais acessíveis ao processo, não somente a substituição do pacote.

1.1. Contexto

A família **Shai-Hulud** já havia sido observada em ondas anteriores de comprometimento da cadeia de suprimentos *npm*, com coleta de segredos, propagação por republicação de pacotes e abuso de repositórios GitHub. Este relatório trata **exclusivamente da onda de 4 de agosto de 2026**. Menções a campanhas anteriores aparecem apenas como contexto técnico de linhagem, sem detalhamento operacional daquelas ondas.

O incidente inicia-se com o comprometimento do ecossistema **Keyv / Cacheable** (mantenedor associado a Jared Wray) e evolui rapidamente para um *worm* automatizado capaz de republicar versões maliciosas sob dezenas de escopos corporativos e de mantenedores, incluindo Ornikar, OneReach, ServiceTitan, Qlik, Deliveroo, Picsart e outros.

O escopo desta entrega cobre:

- identificação do núcleo inicial de pacotes confirmados;
- mapeamento da propagação secundária e das janelas temporais;
- cadeia de execução (`preinstall` → Bun → *payload*);
- indicadores de comprometimento (*IOCs*), regras defensivas e sinais de caça;
- avaliação de risco e recomendações de resposta.

Fora de escopo: atribuição conclusiva do operador humano; contagem validada de máquinas ou organizações efetivamente invadidas; auditoria forense de ambientes específicos de clientes.

1.2. Principais Descobertas

- **keyv@6.0.0** é o primeiro *tarball* malicioso confirmado (publicação **09:35:00 UTC**), com `preinstall: node setup.mjs`, `setup.mjs` e `Math_Symbol.js`.
- Em seguida, ao menos **dez pacotes irmãos** do ecossistema Cacheable / Keyv foram publicados com carga maliciosa entre **10:09** e **10:28 UTC**.
- A campanha se comportou como *worm* de publicação *npm*, gerando rajadas de dezenas a centenas de versões em minutos sob múltiplos escopos.
- Contagens públicas divergem (**444 / 2.234**, **434 / 1.381** e **>428 / >1.700** em inventários distintos), coerentes com remoções rápidas e *snapshots* distintos.
- Exposição transitiva crítica via `eslint` → `file-entry-cache` → `flat-cache` → `keyv`.
- *Provenance npm* aparentemente válida (GitHub Actions) **não** garante integridade do repositório, do *pipeline* ou da identidade de publicação.
- Exfiltração / *staging* via repositórios GitHub públicos com descrição `Shai-Hulud: Here We Go Again` (busca QuimeraX: da ordem de **~3.000** repositórios; não equivale a milhares de vítimas distintas).
- Domínio `npm-cache.com` aparece em parte das análises como C2 / *fallback*; outras amostras examinadas não apresentam C2 tradicional (indicador conflitante).
- Persistência descrita como *dead-man switch* de *tokens* GitHub (`gh-token-monitor`) deve ser caçada, mas sua execução universal não deve ser presumida.
- Pacotes `@keyv/*@6.0.0` permanecem em classificação **suspeita / conflitante** entre inventários e *tarballs* inicialmente obtidos.

Questão	Estado da evidência
Identidade do operador e forma exata de acesso inicial	Não verificado. Há forte associação técnica à família Shai-Hulud, mas sem atribuição pública conclusiva do operador de agosto.
Pacotes `@keyv/*@6.0.0`	Conflitante. Parte dos *tarballs* inicialmente publicados não continha o `preinstall` malicioso; outros inventários os associaram à campanha. Tratar como suspeitos até validação por *hash* e conteúdo.
C2 em `npm-cache.com`	Presente em parte das análises (destino dinâmico ou *fallback*); outras amostras não apresentaram C2 tradicional. Pode representar diferença entre variantes, estágios ou momentos da campanha.
Dead-man switch de *tokens* GitHub	Código e caminhos de instalação/persistência foram observados; em pelo menos um caminho de execução analisado não houve confirmação de chamada ativa.
Número de máquinas ou organizações efetivamente invadidas	Não verificado. Contagens de pacotes, *downloads* mensais e repositórios de exfiltração não equivalem a vítimas executando a carga.

Figura 1: Pontos não verificados ou conflitantes

2. Detalhamento Técnico

2.1. Metodologia de Investigação

A análise seguiu metodologia de *CTI* para incidente de cadeia de suprimentos *npm*, com foco em triangulação de fontes abertas e preservação de indicadores:

1. Correlação de *timelines* de *commits* no repositório `jaredwray/keyv` com eventos de publicação no registro *npm*;
2. Consolidação de listas públicas de pacotes/versões e inventários de *IOCs* disponíveis em fontes abertas;
3. Comparação de *diffs* entre `keyv@6.0.0` e `keyv@6.0.0-rc.1` (código `dist` equivalente; alterações concentradas em manifesto e arquivos auxiliares);
4. Extração de *hashes*, *lifecycle hooks*, caminhos de persistência e marcadores de exfiltração;
5. Mapeamento da cadeia transitiva e do impacto em ambientes de desenvolvimento / CI;
6. Separação explícita entre fatos confirmados, indícios conflitantes e exposição potencial;
7. Elaboração de recomendações de contenção, forense e prevenção contínua.

Fontes prioritárias consultadas: registro *npm*, repositório GitHub `jaredwray/keyv`, listas públicas de *IOCs* e inventários técnicos correlacionados pela equipe **QuimeraX**.

2.1.1. Coleta de Dados

Fonte	Artefato / conteúdo	Uso na análise
GitHub `jaredwray/keyv`	*Commits* `ee2681a9…`, `d8c850c…`, `f97eabc…`, `174f6a5…`	Cadeia inicial de inserção da carga
Registro *npm*	Metadados e *tarball* de `keyv@6.0.0` (quando disponível)	Confirmação de publicação, *hashes* e *provenance*
Lista pública de *IOCs*	CSV com combinações pacote/versão (`keyv-packages.csv`)	Lista inicial de 42 combinações pacote/versão
Inventário ampliado (OSINT)	Relatórios técnicos e eventos históricos do registro	Contagem 444 / 2.234 e agrupamento por família
Análise de *tarball*	Inspeção de manifesto, `scripts` e arquivos auxiliares	Confirmação do núcleo e classificação de `@keyv/*`
Correlação de variantes	Contagens, C2 e caminhos de persistência	Triangulação de divergências (C2, *dead-man switch*)

Figura 2: Fontes e artefatos consultados

As listas públicas de pacotes/versões e inventários de *IOCs* correlacionados **não** devem ser usadas isoladamente como *allowlist* ou *denylist* definitiva. Aproximadamente **550** versões do inventário ampliado já não estavam disponíveis pela consulta normal ao registro e precisaram ser recuperadas por eventos históricos. Consultas atuais ao *endpoint* de uma versão podem retornar **404** mesmo quando aquela versão foi efetivamente publicada e instalada por algum consumidor.

2.1.2. Análise de Indicadores

Os indicadores foram classificados por confiança e por consenso entre fontes:

- **Alta confiança:** *hashes* de `setup.mjs`, `Math_Symbol.js` / `math_init.js` e do *tarball* key `v-6.0.0.tgz`; presença do *hook* `preinstall`; *commits* documentados no repositório Keyv;
- **Média / alta:** caminhos de persistência `gh-token-monitor`, arquivos `.vscode` / `.claude` maliciosos, *workflow* `codeql_analysis.yml` com nome `Run Copilot`;
- **Conflitante / contextual:** C2 `npm-cache.com`, classificação de `@keyv/*@6.0.0`, ativação universal do *dead-man switch*.

A análise priorizou evidência de **execução** (processo `node setup.mjs`, Bun executando o *payload*, artefatos em `/tmp/bun-dl-*`) sobre mera presença de nome de pacote em inventários.

2.2. Escopo e Identificação dos Pacotes Comprometidos

2.2.1. Núcleo inicial confirmado

As **11** versões abaixo foram confirmadas por análise de *tarballs*, registros do *npm* e comparação com versões anteriores. O conjunto representa a primeira onda associada ao mantenedor e aos projetos Keyv / Cacheable.

Pacote	Versão comprometida	Publicação UTC	Estado no levantamento inicial	Orientação imediata
`keyv`	`6.0.0`	09:35:00	Malicioso confirmado	Remover; `5.6.0` validada como limpa; `6.0.0-rc.1` sem o *hook*
`@cacheable/net`	`2.1.1`	10:09:44	Malicioso confirmado	Restaurar versão anterior conhecida pelo *lockfile* e validar o *tarball*
`@cacheable/node-cache`	`3.1.2`	10:10:34	Malicioso confirmado	Remover e reconstruir o ambiente
`cacheable`	`2.5.1`	10:10:44	Malicioso confirmado	Restaurar versão anterior verificada
`flat-cache`	`6.1.24`	10:10:55	Malicioso; removido rapidamente	Fixar `6.1.23` ou outra versão interna previamente validada
`cacheable-request`	`13.0.20`	10:11:24	Malicioso; removido rapidamente	Fixar `13.0.19` ou versão anterior validada

`@cacheable/memory`	`2.2.1`	10:11:29	Malicioso confirmado	Remover e validar versão anterior
`file-entry-cache`	`11.1.6`	10:13:02	Malicioso confirmado	Restaurar versão anterior conhecida pelo *lockfile*
`@cacheable/utils`	`2.5.1`	10:14:21	Malicioso confirmado	Remover e reconstruir
`cache-manager`	`7.2.10`	10:14:41	Malicioso; removido rapidamente	Fixar `7.2.9` ou versão anterior validada
`ecto`	`5.0.1`	10:28:01	Malicioso confirmado	Remover e validar versão anterior

Figura 3: Núcleo inicial de pacotes maliciosos confirmados (04/08/2026, UTC)

"Restaurar versão anterior" **não** significa simplesmente executar um *downgrade* sobre o mesmo `node_modules`. Como a carga já pode ter roubado credenciais e modificado repositórios, é necessário reinstalar em *host* ou imagem limpa, limpar *caches* e verificar o *lockfile*.

2.2.2. Pacotes da propagação secundária

Após keyv, foram observadas publicações em diversos escopos e contas. Exemplos com versões exatas documentadas:

Organização ou família	Exemplos de pacotes e versões	Natureza do risco
ThienNQ	`@thiennq/docs-viewer@1.6.2`	Uma das primeiras propagações, minutos depois de keyv
HubSync	`@hubsync/web-sdk-react@6.3.7` e versões subsequentes	Grande sequência de versões atribuída à propagação automática
Ornikar	`@ornikar/babel-preset-base@6.0.3`, `@ornikar/babel-preset-react@6.1.4`, `@ornikar/eslint-config-react@24.0.1`, `@ornikar/eslint-config-typescript@24.0.1`, `@ornikar/renovate-config@9.0.2`, `@ornikar/stylelint-config@14.0.3`	Ferramentas de *build*, *lint* e configuração, frequentes em estações de desenvolvimento e CI
Deliveroo	`@deliveroo/reevent@1.0.1`, `@deliveroo/determinator@0.2.1`	Pacotes sob escopo corporativo; publicação não comprova comprometimento de sistemas de produção
Picsart	`@picsart/ai-sdk@3.32.2`	SDK de aplicação, com possibilidade de instalação direta por consumidores
OneReach	`@or-sdk/invitations@1.4.9` e amplo conjunto sob `@or-sdk` / `@onereach`	Uma das maiores ondas por número de nomes e versões

Qlik	`@qlik/embed-runtime@1.6.4`, `@qlik/embed-react@2.5.3`, `@qlik/embed-web-components@1.7.3`, `@qlik/runtime-module-loader@1.5.1`	Componentes de *embedding* e *runtime*
Nebula / Picasso	`@nebula.js/nucleus@0.5.1`, `picasso-plugin-hammer@2.11.6`, `picasso-plugin-q@2.11.6`, `picasso.js@2.11.6`	Ecossistema de visualização relacionado à Qlik
ServiceTitan	Centenas de versões sob `@servicetitan` e pacotes relacionados	Maior onda em número de versões no inventário consolidado
ARV Bedrock	Pacotes sob `@arv-bedrock`, incluindo versões sucessivas	Propagação automática por credenciais *npm* recuperadas
Umacloud	Conjunto de nove nomes/versões	Onda posterior, perto de 13:18 UTC
AdminIDE / Workbench	Pacotes sob `@adminide-stack` e `@workbench-stack`	Ferramentas e infraestrutura de desenvolvimento

Figura 4: Exemplos de propagação secundária por organização ou família

2.2.3. Famílias e horários de propagação

O agrupamento abaixo demonstra a velocidade do *worm*. Os números correspondem a um *snapshot* consolidado a partir de eventos do registro e não devem ser somados como contagem definitiva sem deduplicação adicional.

Primeiro evento UTC	Identidade ou escopo afetado	Nomes de pacotes reportados	Versões reportadas
09:35	Jared Wray / família Keyv	11	11
09:38:13	`@thiennq`	1	3
10:12:34	`@hubsync`	1	27
10:19:10	Ornikar e pacotes relacionados	47	537
10:32:09	`@arv-bedrock`	5	10
10:33:58	`@deliveroo`	2	2
10:37:46	Picsart	2	2
10:39:06	OneReach, `@onereach` e `@or-sdk`	155	487
10:41:16	ServiceTitan e pacotes relacionados	147	1.082
10:46:36	Qlik, Nebula e Picasso	59	59
11:00:30	AdminIDE e Workbench	5	5
13:18:05	Umacloud	9	9

Figura 5: Primeiro evento por família (04/08/2026 UTC)

2.2.4. Dependências transitivas

A cadeia mais relevante documentada foi a inclusão de `keyv` por meio de ferramentas de *cache* utilizadas pelo ESLint:

```
Projeto consumidor
    - eslint
        - flat-cache
            - keyv
                - preinstall: node setup.mjs
                    - Bun 1.3.13:
                        - Math_Symbol.js
```

A sequência `eslint` → `file-entry-cache` → `flat-cache` → `keyv` ilustra como o pacote inicial poderia chegar de forma indireta a projetos que nunca declararam `keyv`. A versão efetivamente resolvida varia conforme `package-lock.json`, `yarn.lock`, `pnpm-lock.yaml`, *caches* e o instante da resolução *semver*.

Há ainda uma segunda forma de exposição indireta: pacotes de configuração, *presets* Babel, *plugins* ESLint, *loaders* CSS, configurações Renovate e ferramentas de testes são normalmente classificados como dependências de desenvolvimento, mas são executados justamente nos ambientes que concentram *tokens* de publicação e credenciais de CI. Por isso, `devDependency` **não** reduz materialmente o risco desta campanha.

2.2.5. Pacotes @keyv/*: classificação conflitante

Parte dos *tarballs* de pacotes `@keyv/*@6.0.0`, publicados aproximadamente entre **09:30** e **09:32 UTC**, não continha o `preinstall` nem os arquivos maliciosos nas amostras inicialmente obtidas. Em contraste, inventários ampliados os classificaram como afetados ou associados à campanha.

Pacote / conjunto	Classificação	Orientação
`keyv@6.0.0`	Malicioso confirmado	Remoção imediata e tratamento forense
`@keyv/*@6.0.0`	Suspeito / conflitante	Inspeção do *tarball* e do campo `scripts`
Qualquer versão posterior publicada na janela da campanha	Suspeita até validação	Validar por *hash*, *provenance* e conteúdo

Figura 6: Classificação recomendada para o ecossistema Keyv

2.3. Evidências Técnicas e Cadeia de Ataque

2.3.1. Commits e alterações no repositório Keyv

Repositório primário: `https://github.com/jaredwray/keyv`

Timestamp UTC	Commit	Evidência
09:02:37	`ee2681a9b62f3637b0eb5133c36c864d3376cc5b`	Adição do *loader* e *payload* ao pacote central, alteração do fluxo de publicação e mensagem associada ao lançamento `v6.0.0`. *Commit* não assinado segundo a análise.
09:04:30	`d8c850c7800e…`	Introdução de arquivos em `.vscode` e `.claude`, com centenas de linhas e mecanismos de execução no contexto do editor/assistente. GitHub mostrava o *commit* como “Verified”, apesar da identidade semelhante a *bot* e do coautor forjado.
09:23:50	`f97eabcdd057105f1fce3f05d6c029dac3f2ac78`	Remoção de um teste relacionado ao `preinstall`, reduzindo evidências visíveis sem necessariamente remover a carga do pacote.
09:39:45	`174f6a55690b0812a69adef47260ba8714a9be48`	Preparação ou disseminação dos arquivos maliciosos para pacotes irmãos sob o mesmo *monorepo*.

Figura 7: Commits relevantes documentados (04/08/2026, UTC)

URLs de referência:

- https://github.com/jaredwray/keyv/commit/ee2681a9b62f3637b0eb5133c36c864d3376cc5b
- https://github.com/jaredwray/keyv/commit/d8c850c7800e
- https://github.com/jaredwray/keyv/commit/f97eabcdd057105f1fce3f05d6c029dac3f2ac78
- https://github.com/jaredwray/keyv/commit/174f6a55690b0812a69adef47260ba8714a9be48
- https://github.com/jaredwray/keyv/blob/1f79edd843e2a99ba1634874bf5e5b73cbefb74c/core/keyv/package.json

O *commit* d8c850c… adicionou cinco arquivos sob diretórios de configuração de IDE/agentes, com aproximadamente **670** linhas, incluindo *loaders* capazes de buscar Bun e executar JavaScript. Isso cria uma rota de execução distinta da instalação *npm*: um desenvolvedor que clonasse ou abrisse o repositório em um ambiente que aceitasse tarefas ou *hooks* também poderia ficar exposto.

2.3.2. Diff do pacote publicado

A comparação de `keyv@6.0.0` com `keyv@6.0.0-rc.1` mostrou que o código distribuído em `dist` era byte a byte equivalente; as alterações relevantes estavam no manifesto e nos arquivos adicionais. Isso tornou o comprometimento menos perceptível em uma revisão superficial do código compilado.

O *diff* essencial foi equivalente a:

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

O *hook* `preinstall` é particularmente grave porque roda antes da instalação normal do pacote, inclusive quando o pacote é transitivo, desde que os *lifecycle scripts* estejam habilitados.

2.3.3. Registros e hashes do npm

Artefato	Algoritmo	Hash ou integridade
`keyv-6.0.0.tgz`	SHA-256	`d584f9b6af48b7ed1f93713944f033783bf149e1c25e1643eb8c0e9df5dc7782`
`keyv-6.0.0.tgz`	SHA-512 (hex)	`37f9f847e9c3e520b47d83a9029e199dbc30c6a195a1d80467d0fb1a6fd5068728ad4d1a422995ca578359263afdac5d3b4fed7a6b9befad27de19cca6966952`
Integridade *npm*	SHA-512 (Base64)	`N/n4R+nD5SC0fYOpAp4ZnbwwxqGVod-gEZ9D7Gm/VBocorU0aQimVy-leDWSY6/axdO0/temub760n3hnMppZpUg==`
`setup.mjs` (primeira onda)	SHA-256	`54dc7ea54a1317cca0e890a2770630cf7fa6c97813e0cb9d2caa93012b350668`
`setup.mjs` (variante de propagação)	SHA-256	`fd3ca4007b225fdf8de7af4345a19179d5efa8c4bb9205f88cda806e5684b1eb`
`Math_Symbol.js` / `math_init.js`	SHA-256	`9fc2570b7cef51c1b8df116d144d11ff4096357be7d2c4c6367cfc2509cf1bcc`

Figura 8: Hashes e integridade documentados para keyv@6.0.0 e componentes da carga

Endpoints históricos relevantes:

- `https://registry.npmjs.org/keyv/6.0.0`
- `https://www.npmjs.com/package/keyv/v/6.0.0`

Esses *endpoints* podem deixar de apresentar o conteúdo depois de uma remoção. Por isso, *hashes* de *tarball*, *lockfiles* preservados, *caches* forenses e registros de *proxy* são importantes.

Um aspecto significativo é que `keyv@6.0.0` recebeu *provenance npm* aparentemente válida, produzida por GitHub Actions. A campanha demonstrou que uma *attestation* válida comprova a origem do *pipeline* que publicou o artefato, mas **não** comprova que o *pipeline*, o repositório ou a identidade responsável estavam íntegros.

2.3.4. Cadeia de execução

O *loader* `setup.mjs` identificava sistema operacional, arquitetura e, em Linux, diferenças entre *glibc* e *musl*/Alpine. Em vez de depender de um *runtime* já instalado, baixava **Bun 1.3.13** a partir dos *releases* legítimos do GitHub, executava `Math_Symbol.js` ou `math_init.js` e removia parte dos arquivos temporários. Diretórios com padrão `/tmp/bun-dl-*` foram associados a essa etapa.

Fluxo resumido:

```
Credencial ou repositório comprometido
    - Commit e configuração maliciosa
    - Publicação npm com provenance legítima
    - Instalação direta ou transitiva
    - preinstall executa setup.mjs
    - Detecção de SO e arquitetura
    - Download de Bun 1.3.13
    - Execução de Math_Symbol.js / math_init.js
    - Coleta de segredos
    - Exfiltração
    - Enumeração de pacotes npm
    - Injeção, incremento e republicação
    - Modificação de GitHub e workflows
```

2.3.5. Coleta de credenciais

As análises identificaram funções voltadas a:

- *tokens npm* e metadados de autenticação;
- *tokens* GitHub, incluindo prefixos ghp_ e gho_;
- credenciais AWS em variáveis, arquivos e *metadata services*;
- *tokens* Kubernetes e configurações de *clusters*;
- Vault;
- chaves SSH;
- arquivos `.env`, configurações de bancos de dados e diretórios de projetos;
- memória e processos associados a *runners* de CI;
- segredos expostos temporariamente em *workflows* GitHub Actions.

A simples ausência de um *token npm* no computador não elimina o risco: um *runner* com *token* GitHub capaz de alterar *workflows* poderia ser usado para extrair segredos adicionais de Actions.

2.3.6. Propagação automática pelo npm

A carga enumerava os pacotes mantidos pela conta *npm* comprometida, verificava permissões e condições como `bypass_2fa`, baixava o *tarball* mais recente, adicionava os arquivos maliciosos, substituía ou complementava *scripts*, incrementava a versão e submetia diretamente uma nova versão ao registro. Algumas análises também observaram geração de metadados DSSE / Fulcio / Rekor, fazendo com que versões geradas pelo *worm* parecessem ter *provenance* normal.

Isso explica as rajadas de dezenas ou centenas de versões em poucos minutos e diferencia o incidente de um comprometimento manual pacote a pacote.

2.3.7. Comprometimento de repositórios e CI/CD

Uma das variantes examinadas adicionava ou alterava:

```
.vscode/tasks.json
.vscode/setup.mjs
.claude/math_init.js
.claude/settings.json
.claude/setup.mjs
```

Também foram observados *commits* com mensagem `chore: update config`, autoria semelhante a *bots* e coautoria atribuída a Claude. A automação usava a API GraphQL do GitHub para alterar múltiplas *branches* e evitava algumas *branches* de *bots*, como Dependabot e Copilot.

Outra técnica consistia em criar temporariamente a *branch*:

```
dependabot/github_actions/format/setup-formatter
```

e adicionar:

```
.github/workflows/codeql_analysis.yml
```

O *workflow* aparecia com nome semelhante a `Run Copilot`, serializava o contexto de segredos, publicava o resultado como *artifact* e, depois, tentava apagar a *branch* e o *run*. A aparente autoria de `github-advanced-security[bot]` era usada para reduzir suspeitas.

2.3.8. Exfiltração e C2

Há consenso de que a campanha usou repositórios GitHub públicos como mecanismo de *staging* ou exfiltração, com descrição:

```
Shai-Hulud: Here We Go Again
```

Em monitoramento da equipe **QuimeraX**, uma busca no GitHub por `"Shai-hulud"` retornou da ordem de **~3.000 repositórios**, além de milhares de *issues* e *pull requests* associados. Os resultados incluem criação em massa sob contas como **`Dewforest`**, com nomes temáticos (universo de *Dune*) e a mesma descrição da campanha em todos os repositórios listados. Esse volume representa artefatos potenciais de exfiltração ou *staging*, e **não** equivale a milhares de vítimas distintas.

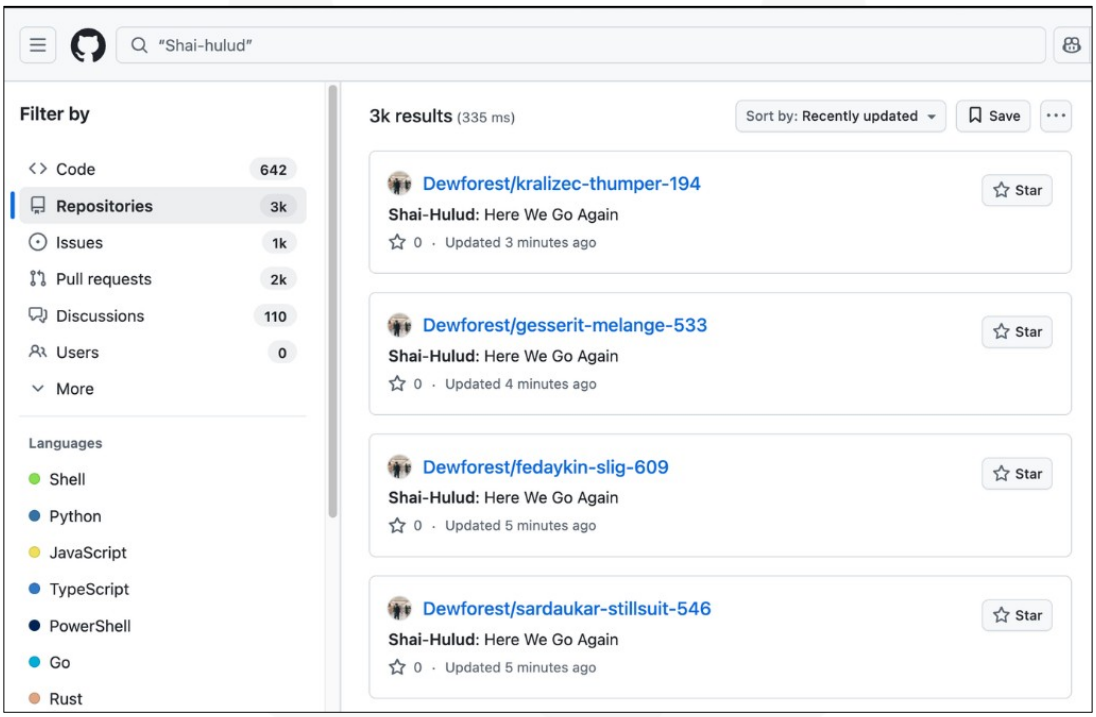

Figura 9: Busca no GitHub por Shai-hulud: cerca de 3.000 repositórios com descrição idêntica Shai-Hulud: Here We Go Again, incluindo criação em massa sob a conta Dewforest

Parte das análises associou `npm-cache.com` a infraestrutura de *fallback* ou a endereço recuperável dinamicamente, inclusive por meio de um contrato Ethereum. Outras amostras examinadas, porém, não apresentaram um *endpoint* C2 tradicional. Portanto, `npm-cache.com` deve ser bloqueado e investigado como indicador de alta relevância, mas sua presença **não** deve ser descrita como universal a todas as amostras.

2.4. Cronologia da Campanha de Agosto

Todos os horários abaixo são de **4 de agosto de 2026** em UTC. Para o horário de Brasília, subtraem-se três horas.

Hora UTC	Ator ou sistema	Evento e interpretação
~09:00	Operador não identificado usando identidade comprometida	Início de atividade no repositório Keyv e preparação de persistência em IDE/agentes. A forma de obtenção da conta permanece não verificada.
09:02:37	Repositório `jaredwray/keyv`	*Commit* `ee2681a9…` adiciona componentes da carga e altera lógica de publicação.
09:04:30	Repositório Keyv / GitHub	*Commit* `d8c850c…` adiciona configurações `.vscode` e `.claude`; *commit* exibido como verificado.
09:23:50	Repositório Keyv	*Commit* `f97eabc…` remove teste que tornava o *hook* mais evidente.
09:30–09:32	*npm* / escopo `@keyv`	Pacotes `@keyv/*@6.0.0` publicados; conteúdo malicioso não confirmado nos primeiros *tarballs* obtidos.
09:35:00	*npm* / `keyv`	`keyv@6.0.0` publicado com `preinstall`, `setup.mjs` e `Math_Symbol.js`.
~09:41	Comunidade / detecção precoce	Detecção pública cerca de seis minutos após a publicação de `keyv@6.0.0`.
09:38–09:39	Conta `@thiennq` e repositório Keyv	Primeira propagação para outro mantenedor e *commit* preparando pacotes irmãos.

09:49–09:51	Comunidade do projeto	Abertura das *issues* #2044, #2045 e #2046, sinalizando atividade anormal.
10:09:44–10:14:41	*npm* / família Cacheable	Nove publicações maliciosas em aproximadamente cinco minutos.
10:19–10:47	Contas corporativas e mantenedores	Propagação para Ornikar, ARV, Deliveroo, Picsart, OneReach, ServiceTitan e Qlik.
10:28:01	*npm* / `ecto`	Publicação de `ecto@5.0.1`, última das 11 versões do núcleo inicial enumerado.
11:16	Registro *npm*	*Snapshot* encontra oito versões ainda como `latest`; três já removidas do registro.
13:18:05	Umacloud	Onda adicional de pacotes.
13:37 CEST	Inventários públicos	Atualizações indicando ao menos 434 pacotes e 1.381 versões, com mais de dois bilhões de instalações mensais agregadas (exposição potencial, não número de execuções maliciosas).

Figura 10: Cronologia consolidada (commits, npm e detecções)

Campanhas anteriores Shai-Hulud são relevantes apenas como contexto técnico: a carga de agosto reutiliza conceitos de coleta de segredos, propagação por publicação de pacotes e abuso de GitHub observados em variantes anteriores, mas introduz ou amplia persistência em IDE/agentes, *provenance* produzida por *pipelines* legítimos e automação em escala muito maior. A atribuição à mesma família decorre da linhagem de código, dos marcadores e das técnicas, não de prova pública de que o mesmo indivíduo operou todas as ondas.

2.5. Técnicas, Táticas e Procedimentos (TTPs)

Tática	Técnica	Observação na campanha
Initial Access	T1195.001 Supply Chain Compromise: Compromise Software Dependencies and Development Tools	Publicação de versões maliciosas no registro *npm*
Execution	T1059 Command and Scripting Interpreter	`preinstall` → node `setup.mjs`; execução via Bun
Persistence	T1543 Create or Modify System Process / Launch Agent	`gh-token-monitor` (systemd / LaunchAgent), conforme amostras
Persistence	T1554 Compromise Host Software Binary / config abuse	Arquivos `.vscode` e `.claude` para execução em IDE/agentes
Credential Access	T1552 Unsecured Credentials	Leitura de `.npmrc`, `.env`, SSH, *kubeconfig*, Vault, etc.
Credential Access	T1555 Credentials from Password Stores / cloud metadata	Coleta AWS / *metadata services* e segredos de CI
Discovery	T1083 File and Directory Discovery	Enumeração de projetos, pacotes e permissões *npm*
Collection	T1005 Data from Local System	Coleta ampla de segredos locais e de *runners*
Exfiltration	T1567 Exfiltration Over Web Service	Repositórios GitHub públicos / *artifacts*; possível C2 auxiliar
Impact / Defense Evasion	T1070 Indicator Removal	Remoção de teste do `preinstall`, exclusão de *branch*/*run*, limpeza parcial de temporários
Resource Development	T1588 Obtain Capabilities	Download de Bun 1.3.13 a partir de *releases* legítimos do GitHub

Figura 11: Mapeamento MITRE ATT&CK (resumo operacional)

2.6. Indicadores de Comprometimento e Detecção

2.6.1. Indicadores de arquivo e conteúdo

Tipo	Indicador	Confiança
SHA-256	`54dc7ea54a1317cca0e890a2770630cf7fa6c97813e0cb9d2caa93012b350668`	Alta (*loader* inicial)
SHA-256	`fd3ca4007b225fdf8de7af4345a19179d5efa8c4bb9205f88cda806e5684b1eb`	Alta (*loader* de propagação)
SHA-256	`9fc2570b7cef51c1b8df116d144d11ff4096357be7d2c4c6367cfc2509cf1bcc`	Alta (*payload*)
SHA-256	`d584f9b6af48b7ed1f93713944f033783bf149e1c25e1643eb8c0e9df5dc7782`	Alta (*tarball* `keyv@6.0.0`)
Nome	`setup.mjs` junto a `Math_Symbol.js` ou `math_init.js`	Média/alta
Diretório	`/tmp/bun-dl-*`	Média
Persistência	`~/.config/gh-token-monitor/`	Alta quando não esperado
Persistência	`~/.local/bin/gh-token-monitor.sh`	Alta
Serviço	`gh-token-monitor.service`	Alta
LaunchAgent	`com.user.gh-token-monitor`	Alta
Lock de processo	`/tmp/tmp.dpkg_14527.lock`	Alta para a variante analisada
Chave PBKDF2 / *salt*	`svksjrhjkcejg` com 200.000 iterações	Alta nas amostras analisadas

Chave embutida	899d419bf1e9ecc25bd436832aff03b6b9f73af20b053cbb9ff27e43512378b3	Alta nas amostras analisadas
SHA-256 de chave RSA DER	dc1e6a7ddb29390dd53cf1e5aac40ad9204ea7c6b83ef5656e7cb7a796808b67	Alta
SHA-256 de chave RSA DER	166be2b7b58a440f7b17520ffb0368be5d89c76661704b4945417eb04b9ada65	Alta

Figura 12: IOCs de arquivo, hash e persistência

Os *hashes* e materiais criptográficos acima foram consolidados a partir da análise técnica das amostras e de inventários públicos correlacionados pela equipe **QuimeraX**.

2.6.2. Indicadores de rede

Indicador	Uso reportado	Classificação
`npm-cache.com`	C2, *fallback* ou *endpoint* dinâmico em parte das amostras	Alto interesse, mas conflitante entre amostras
`pypi-get.com`	Infraestrutura relacionada observada em inventários públicos	Investigar e bloquear se não houver uso legítimo
`js-mirror.com`	Infraestrutura relacionada observada em inventários públicos	Investigar e bloquear
GitHub *releases* de Bun 1.3.13	Download do *runtime* pelo *loader*	Comportamental; domínio GitHub é legítimo
`registry.npmjs.org/-/npm/v1/user` e *endpoints* de identidade/*token*	Verificação e abuso de credenciais *npm*	Comportamental
`169.254.169.254` e outros *metadata endpoints*	Coleta de credenciais de nuvem	Alto risco em processo Node/Bun inesperado

Endpoints RPC Ethereum	Recuperação de dados de contrato em parte das análises	Indicador contextual, não necessariamente malicioso isoladamente

Figura 13: IOCs e sinais de rede

A ausência de tráfego para `npm-cache.com` **não** descarta comprometimento, porque outras análises observaram exfiltração por GitHub e *artifacts* sem C2 tradicional.

2.6.3. Indicadores em GitHub

```
Descrição de repositório:
Shai-Hulud: Here We Go Again

Mensagem de commit:
chore: update config

Branch:
dependabot/github_actions/format/setup-formatter

Workflow:
.github/workflows/codeql_analysis.yml

Nome exibido do workflow:
Run Copilot

Identidade simulada:
github-advanced-security[bot]

Arquivos:
.vscode/setup.mjs
.vscode/tasks.json
.claude/setup.mjs
.claude/math_init.js
.claude/settings.json
```

Texto intimidatório observado em alguns artefatos, inclusive como mensagem de *commit* (exemplo abaixo, conta `nishantosc`, *branch* `main`), seguido de *payload* ofuscado em Base64:

```
IfYouBlockThisAPIKeyItWillCrashTheLiveProductionServersOfAllThirdPartyClients
```

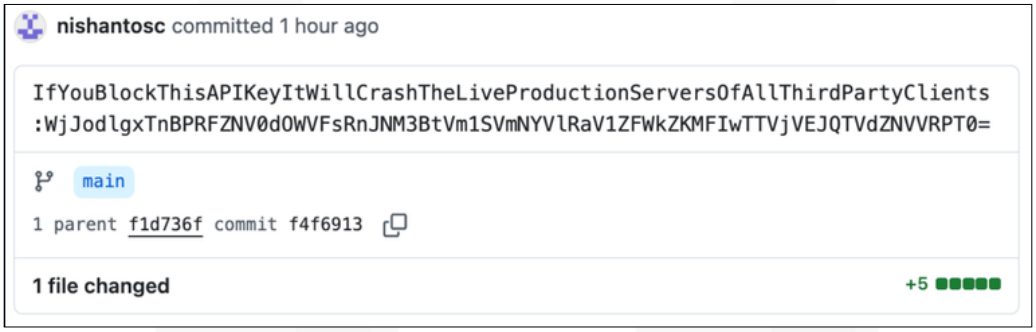

Figura 14: Commit malicioso com mensagem intimidatória IfYouBlockThisAPIKeyItWillCrashTheLiveProductionServersOfAllThirdPartyClients e payload ofuscado em Base64

Esses sinais devem ser correlacionados com *timestamps*, autoria, criação de repositórios, *downloads* de *artifacts* e alterações de *secrets*.

2.6.4. Regras defensivas (YARA e Sigma)

Regra YARA para *hashes* exatos:

```
import "hash"

rule NPM_ShaiHulud_Aug2026_Exact_Artifacts
{
    meta:
        description = "Detecta artefatos confirmados da campanha npm Shai-Hulud de agosto de 2026"
        author = "QuimeraX CTI - regra defensiva"
        date = "2026-08-04"
        confidence = "high"

    condition:
        filesize > 0 and
        (
            hash.sha256(0, filesize) ==
                "54dc7ea54a1317cca0e890a2770630cf7fa6c97813e0cb9d2caa93012b350668"
            or
            hash.sha256(0, filesize) ==
                "fd3ca4007b225fdf8de7af4345a19179d5efa8c4bb9205f88cda806e5684b1eb"
            or
            hash.sha256(0, filesize) ==
                "9fc2570b7cef51c1b8df116d144d11ff4096357be7d2c4c6367cfc2509cf1bcc"
            or
            hash.sha256(0, filesize) ==
                "d584f9b6af48b7ed1f93713944f033783bf149e1c25e1643eb8c0e9df5dc7782"
        )
}
```

Regra YARA heurística para o *loader*:

```
rule NPM_ShaiHulud_Aug2026_Loader_Heuristic
{
    meta:
        description = "Detecção heurística de loader setup.mjs associado à cam
            panha"
        author = "QuimeraX CTI - regra defensiva"
        date = "2026-08-04"
        confidence = "medium"
        warning = "Pode exigir ajuste para reduzir falsos positivos"

    strings:
        $bun_release = "bun-v1.3.13" ascii nocase
        $bun_tmp     = "bun-dl-" ascii
        $math_1      = "Math_Symbol.js" ascii
        $math_2      = "math_init.js" ascii
        $exec_1      = "execFileSync" ascii
        $exec_2      = "spawnSync" ascii
        $platform_1  = "process.platform" ascii
        $platform_2  = "process.arch" ascii

    condition:
        filesize < 200KB
        and $bun_release
        and $bun_tmp
        and 1 of ($math_*)
        and 1 of ($exec_*)
        and 1 of ($platform_*)
}
```

Regra Sigma para execução do *loader* ou *payload*:

```
title: Shai-Hulud August 2026 NPM Loader or Bun Payload Execution
id: 5f6b3ed4-18f0-4c9e-a130-b442fb6a8462
status: experimental
description: >
  Detecta Node executando setup.mjs ou Bun executando os payloads
  Math_Symbol.js e math_init.js associados à campanha Shai-Hulud.
author: QuimeraX CTI
date: 2026-08-04
logsource:
  category: process_creation
detection:
  selection_node:
    Image|endswith:
      - '/node'
      - '\node.exe'
    CommandLine|contains:
      - 'setup.mjs'
  selection_bun:
    Image|endswith:
      - '/bun'
      - '\bun.exe'
    CommandLine|contains:
      - 'Math_Symbol.js'
      - 'math_init.js'
  selection_paths:
    CommandLine|contains:
      - '/tmp/bun-dl-'
      - '\Temp\bun-dl-'
  condition: selection_node or selection_bun or selection_paths
falsepositives:
  - Aplicações internas que utilizam legitimamente arquivos chamados setup.mjs
  - Projetos legítimos que baixam e executam Bun em diretórios temporários
level: high
tags:
  - attack.execution
  - attack.t1059
  - attack.supply_chain
```

Regra Sigma para persistência de monitor de *token*:

```
title: Shai-Hulud GitHub Token Monitor Persistence
id: f16a31e2-7227-44d2-84e7-06d50ef829db
status: experimental
description: Detecta execução ou instalação do gh-token-monitor associado à ca
   mpanha.
author: QuimeraX CTI
date: 2026-08-04
logsource:
  category: process_creation
detection:
  selection:
    CommandLine|contains:
      - 'gh-token-monitor'
      - 'com.user.gh-token-monitor'
      - 'gh-token-monitor.service'
  condition: selection
falsepositives:
  - Ferramenta interna com nome coincidente, pouco provável
level: critical
tags:
  - attack.persistence
  - attack.credential_access
```

2.6.5. Sinais prioritários em CI/CD

Sinal	Prioridade
Processo node executando `setup.mjs` durante `preinstall` e criando processo Bun	Crítica
Download de Bun 1.3.13 por *job* que nunca utilizou Bun	Alta
Leitura de `.npmrc`, `.env`, `.ssh`, *kubeconfig* e credenciais AWS pelo mesmo processo	Crítica
Acesso a `/proc/*/mem` por Node/Bun em *runner*	Crítica
Acesso a *metadata service* de nuvem por *job* de *frontend*, *lint* ou testes	Crítica
Publicação de dezenas de versões *npm* em minutos	Crítica
Nova versão *patch* cujo `dist` não mudou, mas `scripts`/`files` mudaram	Crítica
Provenance válida ligada a *commit* ou *workflow* não revisado	Alta
Criação e exclusão rápida da *branch* `dependabot/github_actions/format/setup-formatter`	Crítica
Workflow Run `Copilot` usando serialização ampla do contexto de *secrets*	Crítica
Repositório público recém-criado com descrição da campanha	Crítica
Novos arquivos `.claude/setup.mjs` ou `.vscode/setup.mjs` em repositório que não os utilizava	Alta
Alteração de *dist-tag* `latest` seguida de *unpublish*	Alta

Figura 15: Sinais comportamentais de maior valor

O monitoramento contínuo deve correlacionar registro *npm*, GitHub *audit log*, *logs* de Actions, DNS/*proxy*, EDR, *cloud audit logs* e SBOMs. Nenhuma camada isolada é suficiente: a campanha conse-

guiu produzir versões com *provenance* aparentemente legítima, usar infraestrutura legítima do GitHub para *downloads* e operar por dependências transitivas que não eram visíveis no manifesto de nível superior.

2.7. Análise de Impacto

2.7.1. Impacto confirmado versus exposição potencial

O que está confirmado é a publicação de *tarballs* maliciosos, sua resolução por consumidores durante a janela de disponibilidade, alterações em repositórios e a criação de artefatos de exfiltração. Não há, até o corte deste relatório, uma contagem pública validada de estações, *runners* ou ambientes de produção nos quais a carga tenha sido efetivamente executada.

Os "mais de dois bilhões de *downloads* ou instalações mensais" associados aos pacotes representam a popularidade agregada de seus ecossistemas, e não dois bilhões de infecções. O risco real depende de:

1. o projeto ter resolvido uma versão exata comprometida;
2. a instalação ter ocorrido enquanto a versão estava disponível ou presente em *cache*;
3. *lifecycle scripts* terem sido executados;
4. o processo ter acesso a credenciais ou segredos úteis;
5. a carga ter concluído execução e exfiltração;
6. credenciais roubadas terem sido reutilizadas pelo operador.

Os escopos Deliveroo, Picsart, Qlik, Ornikar, OneReach e ServiceTitan tiveram pacotes publicados na onda. Isso comprova abuso de identidades ou permissões de publicação associadas aos respectivos pacotes, mas **não** comprova, por si só, invasão de sistemas de produção, clientes ou redes corporativas dessas empresas.

2.7.2. Riscos por tipo de ativo

Ativo	Consequência plausível
Conta *npm* de mantenedor	Publicação de novas versões maliciosas, alteração de *dist-tags*, comprometimento da base consumidora
GitHub pessoal ou organizacional	*Commits*, *branches* e *workflows* maliciosos; criação de repositórios públicos; extração de Actions *secrets*
Runner CI/CD	Roubo de *tokens* de publicação, nuvem, *registries*, *deploy* e *signing*; acesso à memória de processos
AWS e outras nuvens	Enumeração de identidade, acesso a *buckets*, *secrets managers* ou *workloads*, conforme permissões
Kubernetes	Uso de *service-account tokens* e *kubeconfigs* para acessar *clusters*
Vault	Leitura de segredos e expansão lateral do comprometimento
Estação de desenvolvedor	Roubo de SSH, `.npmrc`, *tokens* GitHub, arquivos `.env` e credenciais de múltiplos projetos
Repositório interno	Persistência em `.vscode`, `.claude` e *workflows*, permitindo reinfecção após corrigir o pacote
Registry ou *cache* interno	Redistribuição prolongada de versões já removidas do *npm* público

Figura 16: Consequências plausíveis por ativo

3. Conclusão

3.1. Síntese dos Achados

A onda de **4 de agosto de 2026** da campanha **Shai-Hulud** representa um incidente de cadeia de suprimentos *npm* de alta velocidade, iniciado em `keyv@6.0.0` e expandido por um *worm* de republicação automatizada. A combinação de dependência transitiva popular (ESLint / *cache*), execução via `preinstall`, *runtime* Bun baixado de fonte legítima e *provenance* aparentemente válida elevou tanto o alcance potencial quanto a dificuldade de triagem superficial.

Os principais fatos consolidados são:

- núcleo inicial de **11** versões maliciosas confirmadas no ecossistema Keyv / Cacheable;
- propagação para centenas de pacotes e milhares de versões sob múltiplos escopos;
- coleta ampla de credenciais e abuso de GitHub / CI/CD para persistência e exfiltração;
- divergências relevantes entre fontes quanto a C2, `@keyv/*` e ativação do *dead-man switch*;
- ausência, até o corte, de contagem pública validada de vítimas com execução confirmada.

3.2. Avaliação de Risco

Categoria	Condição observada	Risco	Ação
Execução confirmada	*Logs* mostram node `setup.mjs`, Bun executando `Math_Symbol.js` /`math_init.js`, ou *hashes* confirmados em `node_modules`	Crítico	Isolar, preservar evidência, remover persistência, rotacionar credenciais e reconstruir
Instalação com *scripts* habilitados	*Lockfile* ou *cache* contém versão comprometida e houve `npm install` / `npm ci` durante a janela	Crítico	Tratar como execução até prova em contrário

Runner CI efêmero	Versão comprometida instalada, mas *runner* foi destruído	Crítico para credenciais	Rotacionar *tokens* acessíveis; revisar publicações, *workflows* e *cloud logs*. Efemeridade não impede exfiltração
Versão presente, *scripts* bloqueados	Instalação com `--ignore-scripts` ou controle equivalente confirmado por *log*	Alto	Remover versão, validar que não houve execução por IDE ou comando manual e revisar *caches*
Repositório clonado/aberto	Presença de `.vscode` ou `.claude` malicioso, ainda que não haja pacote instalado	Alto	Verificar tarefas executadas, extensões, confiança do *workspace*, processos e *tokens*
Apenas referência sem instalação	*Lockfile* alterado, mas nenhum *build* ou *install* ocorreu	Médio	Corrigir *lockfile*, invalidar *caches* e impedir promoção do *build*
Pacote `@keyv/*@6.0.0` sem *hook* e *hash* validado	Divergência entre listas, *tarball* inspecionado como limpo	Médio / suspeito	Substituir preventivamente e manter evidência; não declarar comprometimento sem outros sinais
Versão anterior validada, sem artefatos	*Lockfile* e *cache* comprovadamente anteriores à onda	Baixo, não zero	Monitorar credenciais e mudanças inesperadas de repositório

Figura 17: Matriz de risco operacional

Avaliação geral: o risco residual para qualquer organização que tenha instalado dependências *npm* com *lifecycle scripts* habilitados na janela de **4 de agosto de 2026** é **alto a crítico**, até que a ausência de versões afetadas, de execução e de persistência seja demonstrada.

4. Ações Recomendadas

4.1. Ações Imediatas

A ordem das ações é importante. Foi descrito um mecanismo de monitoramento de *token* GitHub que poderia reagir à invalidação da credencial; sua ativação não foi confirmada em todas as amostras. Ainda assim, é prudente procurar e neutralizar essa persistência **antes** de revogar *tokens* a partir do *host* suspeito.

1. **Isolar** o sistema ou *runner*. Suspenda *pipelines*, bloqueie *egress* quando possível e preserve discos, *logs*, `node_modules`, *caches*, *lockfiles* e histórico de *shell*. Evite começar com limpeza destrutiva.
2. **Caçar o *dead-man switch* e outras persistências** nos caminhos abaixo, calcular *hashes* antes de apagar e inspecionar repositórios locais quanto a `.vscode`, `.claude` e `.github/workflows/codeql_analysis.yml`.
3. **Identificar dependências afetadas** com `npm ls`, `npm explain` e busca em *lockfiles* pelas versões do núcleo e pelas listas públicas de *IOCs*.
4. **Rotacionar credenciais a partir de estação limpa** (*npm*, GitHub, SSH, AWS e demais nuvens, Kubernetes, Vault, bancos, *registries*, *deploy* e *signing*).
5. **Reconstruir** em *host*/imagem limpa com `--ignore-scripts`, *overrides* testados e validação de *tarball*.
6. **Auditar** publicações *npm*, *dist-tags*, *maintainers*, *provenance*, repositórios/branches/workflows GitHub e *runs* excluídos.

Comandos úteis para caça e contenção:

```
# Linux: status e desativação do serviço associado

systemctl --user status gh-token-monitor.service 2>/dev/null
systemctl --user disable --now gh-token-monitor.service 2>/dev/null
loginctl show-user "$USER" -p Linger 2>/dev/null

find "$HOME" /tmp -type f \
  \( -name 'Math_Symbol.js' -o -name 'math_init.js' -o \
     -name 'setup.mjs' -o -name 'gh-token-monitor.sh' \) \
  -print 2>/dev/null

# macOS

launchctl list | grep -i gh-token-monitor
find "$HOME/Library/LaunchAgents" -type f \
  -name '*gh-token-monitor*' -print 2>/dev/null

# Hashes antes de apagar

sha256sum setup.mjs Math_Symbol.js math_init.js 2>/dev/null
shasum -a 256 setup.mjs Math_Symbol.js math_init.js 2>/dev/null

# Persistência em repositórios

find . -type f \( \
  -path '*/.vscode/tasks.json' -o \
  -path '*/.vscode/setup.mjs' -o \
  -path '*/.claude/setup.mjs' -o \
  -path '*/.claude/math_init.js' -o \
  -path '*/.claude/settings.json' -o \
  -path '*/.github/workflows/codeql_analysis.yml' \
\) -print
```

Identificação de dependências:

```
npm ls \
  keyv flat-cache file-entry-cache cacheable-request cacheable \
  @cacheable/memory @cacheable/net @cacheable/node-cache \
  @cacheable/utils cache-manager ecto \
  --all

npm explain keyv
npm explain flat-cache
npm explain file-entry-cache

rg -n \
  'keyv.*6\.0\.0|flat-cache.*6\.1\.24|file-entry-cache.*11\.1\.6|cacheable-request.*13\.0\.20|cacheable.*2\.5\.1|cache-manager.*7\.2\.10|ecto.*5\.0\.1' \
  package-lock.json npm-shrinkwrap.json yarn.lock pnpm-lock.yaml 2>/dev/null
```

Não confie apenas em `npm ls` executado depois da remoção do pacote: o diretório já pode ter sido alterado, e a versão pode continuar em *cache*, *registry* interno ou artefato de *build*.

A presença de `setup.mjs` isoladamente pode ser legítima. Classifique por *hash*, conteúdo, referência a Bun 1.3.13, execução de `Math_Symbol.js`/`math_init.js` e alterações contemporâneas.

Rotação recomendada (estação limpa):

1. Revogar todos os *tokens npm* acessíveis ao usuário ou *runner*;
2. Revogar *tokens* GitHub pessoais, OAuth, GitHub App e chaves SSH;
3. Revisar sessões ativas e autorizações de aplicações;
4. Rotacionar AWS *access keys* e credenciais de outras nuvens;
5. Invalidar *tokens* Kubernetes, *service accounts* e *kubeconfigs* quando aplicável;
6. Revogar *tokens* Vault e segredos obtidos por eles;
7. Rotacionar credenciais de bancos, *registries*, *deploy* e *signing*;
8. Trocar segredos presentes em `.env`, *logs*, *artifacts* ou memória de *runners*;
9. Revisar pacotes *npm* publicados, *dist-tags*, alterações de *maintainers* e *provenance*;
10. Revisar repositórios GitHub recém-criados, *forks*, *branches*, *workflows*, *artifacts* e *runs* excluídos.

A documentação do GitHub recomenda revogar ou rotacionar o segredo na fonte e revisar *logs* e usos posteriores; simplesmente apagar o segredo do repositório ou do histórico não o torna inválido.

4.2. Ações de Médio Prazo

Reconstrução segura e endurecimento do ciclo de dependências:

```
# Em host limpo e após preservar evidências:

rm -rf node_modules

# Limpar cache local; avalie impacto antes em ambientes compartilhados

npm cache clean --force

# Atualizar package.json/overrides e lockfile sem executar scripts

npm install --package-lock-only --ignore-scripts

# Instalação reprodutível sem lifecycle scripts

npm ci --ignore-scripts
```

Mitigação temporária sugerida para o núcleo Keyv / Cacheable (testar compatibilidade):

```
{
  "overrides": {
    "keyv": "5.6.0",
    "flat-cache": "6.1.23",
    "cacheable-request": "13.0.19",
    "cache-manager": "7.2.9"
  }
}
```

Para os demais pacotes, fixe a versão anterior presente em um *lockfile* conhecido como limpo e valide que o *tarball* não contém *lifecycle hooks* inesperados.

O *npm* 12 passou a não executar determinados *scripts* de instalação sem aprovação explícita por padrão. Isso reduz o risco de execução automática, mas não elimina comprometimento por IDE, execução manual, gerenciadores antigos ou *pipelines* configurados para permitir *scripts*.

Controle	Implementação recomendada
Lifecycle scripts	Usar `npm ci --ignore-scripts` como padrão e *allowlist* explícita para pacotes que realmente necessitam de *build*
Lockfiles	Exigir *lockfile* imutável em CI, revisar mudanças de integridade e impedir resolução *semver* fora de revisão
Quarentena de versões novas	Atrasar promoção de versões recém-publicadas em *proxies* internos, especialmente atualizações *patch* inesperadas
Diff de *tarballs*	Comparar `scripts`, arquivos adicionados, *entrypoints* e tamanho com a versão anterior antes de promover
Dependências transitivas	Gerar SBOM e consultar a árvore resolvida, não apenas `package.json`
Provenance	Validar *provenance*, mas correlacionar *commit*, *workflow*, revisão e identidade. *Provenance* isolada não é suficiente
Tokens npm	Preferir *trusted publishing*/OIDC, *tokens* de escopo mínimo e expiração curta; evitar *tokens* persistentes em estações
GitHub Actions	Usar *runners* efêmeros, ambientes protegidos, aprovação para *secrets* e permissões `GITHUB_TOKEN` mínimas
Proteção de *workflows*	CODEOWNERS e revisão obrigatória para `.github/workflows`, `package.json`, *lockfiles*, `.vscode` e `.claude`
IDEs e agentes	Desabilitar execução automática em *workspaces* não confiáveis; revisar *tasks* e configurações de agentes
Registry interno	Verificar e purgar versões removidas do *upstream*; manter *logs* históricos de *download* e *hash*
Detecção comportamental	Alertar para Node baixando Bun, acesso a *metadata services*, leitura de `/proc/*/mem` e publicação *npm* em massa
Resposta a segredos	Manter *playbooks* para revogação simultânea de *npm*, GitHub, nuvem, Kubernetes e Vault

Figura 18: Controles de prevenção contínua (médio prazo)

Na auditoria *npm*, procure: versões *patch* publicadas em rajadas; mudanças de `latest`; novos *maintainers*; *tokens* com `bypass_2fa`; publicações fora do horário normal; *provenance* válida originada de *workflow* ou *commit* inesperado; versões removidas rapidamente; pacotes republicados sem alterações funcionais em `dist`, mas com mudanças em `scripts` e `files`.

No GitHub, procure os marcadores da seção 2.6.3 e eventos de: `createRepository`; criação e exclusão rápida de *branches*; `workflow_dispatch` e *runs* removidos; *download* de *artifacts*; alterações em *secrets*; criação de PATs; *commits* em dezenas de *branches* em poucos minutos; *commits* "Verified" sem processo de revisão correspondente.

4.3. Ações de Longo Prazo

1. **Programa de segurança de cadeia de suprimentos de software (*SSCS*)** com SBOM contínuo, *diff* automatizado de *tarballs* e política de quarentena para versões novas.
2. **Identidade e publicação sem *token* persistente**: migrar publicação *npm* para OIDC / *trusted publishing*, com segregação de contas de mantenedor e revisão humana para mudanças de `scripts`/`files`.
3. **Endurecimento de CI/CD**: *runners* efêmeros, permissões mínimas, proteção de ambientes, revisão obrigatória de *workflows* e monitoramento de serialização de *secrets*.
4. **Controle de IDE/agentes**: política organizacional para desabilitar execução automática de *tasks* em repositórios não confiáveis e inventário de configurações `.vscode` / `.claude`.
5. **Detecção correlacionada**: unificar telemetria de EDR, DNS/*proxy*, GitHub *audit*, registro *npm* e *cloud audit* com as regras YARA/Sigma deste relatório.
6. **Exercícios de resposta** específicos para *supply chain npm*, incluindo rotação massiva de credenciais e reconstrução de estações/runners sem reutilização de `node_modules` ou *home* contaminados.
7. **Governança de *proxies* e *caches* internos**: retenção histórica de *hashes*, capacidade de *purge* rápido e bloqueio de reintrodução de versões removidas no *upstream*.