Rooting For Secrets with TruffleHog

By BHIS

Published: 2024-01-18 · Archived: 2026-04-29 02:09:29 UTC

The potential leaking of confidential information can pose a significant security risk for any organization. When sensitive details (i.e., API keys, passwords, cryptographic keys, and other credentials) are unintentionally committed to version control systems like Git, they could lead to a compromise of systems, data, or other resources.

Leaking secrets can have severe repercussions for an organization, compromising data integrity, confidentiality, and system security. Exposed tokens can provide unauthorized access to sensitive information, enabling malicious actors to manipulate or steal data, disrupt services, and potentially escalate their attacks. Additionally, exploited tokens could also be leveraged to conduct sophisticated phishing campaigns or launch further cyberattacks.

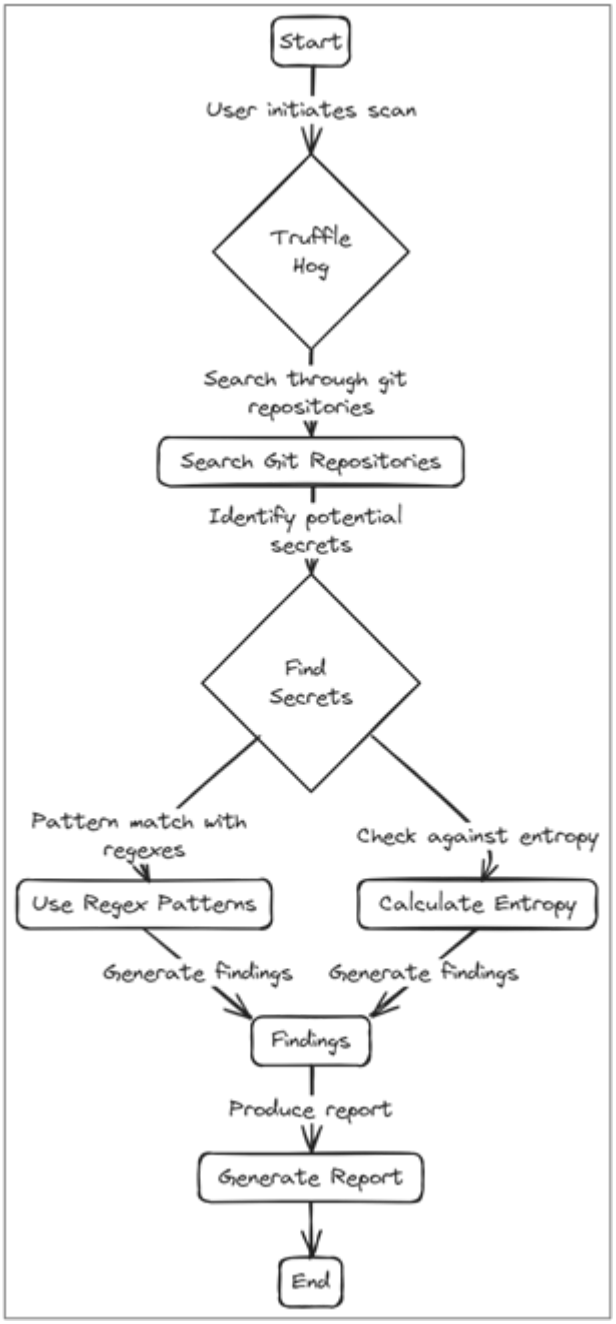
The impact of this lapse in security could manifest as financial losses, reputational damage, and legal consequences.

So how do you know if you have buried secrets hiding in the vast digital landscape of your organization? Easy. You employ a truffle hog.

TruffleHog

TruffleHog[1] is a *free* security tool designed to root around for sensitive information exposure within version control systems, CI, cloud assets, and file systems. Specifically, it helps identify and mitigate security risks related to the inadvertent storage of credentials, secrets, and other sensitive data.

For example, TruffleHog could scan a Git code repository for patterns that resemble known sensitive information, helping the organization and developers proactively identify and remove such data before it becomes a security vulnerability.



Identifying and cleaning up leaked secrets *before* an attacker can find them is a crucial component to security.

Installation

Installing TruffleHog is easy using APT by executing the command below.

```
sudo apt install trufflehog
```

APT not your thing? Don't worry. The tool supports several other methods for installation:

- Using `brew` on MacOS
- Docker

- Binary releases via `https://github.com/trufflesecurity/trufflehog/releases`
- Git clone and compile from source
- Using the `install.sh` script on GitHub (also supports specific version installation

Exact steps for these alternative installation methods can be found at https://github.com/trufflesecurity/trufflehog#floppy_disk-installation.

Sub-Commands

Once installed, it's time to familiarize yourself with the nine available "sub-commands" that TruffleHog uses to root around for secrets. These can be listed by using the `--help` flag from the command line as shown below.

```
trufflehog --help
```

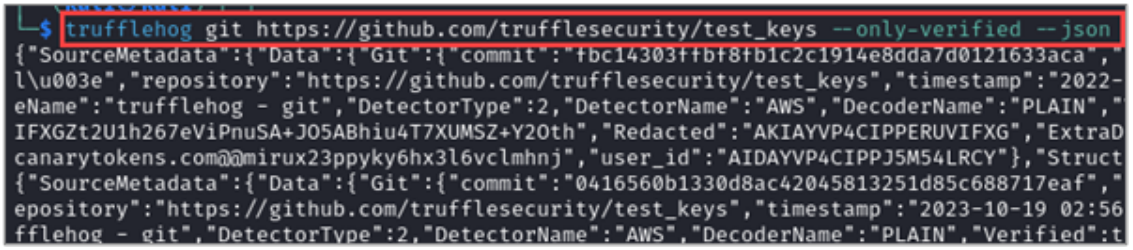
```
git [<flags>] <uri>
    Find credentials in git repositories.

github [<flags>]
    Find credentials in GitHub repositories.

gitlab --token=TOKEN [<flags>]
    Find credentials in GitLab repositories.

filesystem [<flags>] [<path> ...]
    Find credentials in a filesystem.

s3 [<flags>]
    Find credentials in S3 buckets.

gcs [<flags>]
    Find credentials in GCS buckets.

syslog [<flags>]
    Scan syslog

circleci --token=TOKEN
    Scan CircleCI

docker --image=IMAGE
    Scan Docker Image
```

TruffleHog Sub-Commands

Each of the commands above has specific subsequent "flags" that can be set when executing TruffleHog. These additional flags help to both extend functionality and narrow the tool's scope. These flags can be listed by including the `--help` flag after any of the above sub-commands as shown below.

Optional Flags (Snippet)

There are some flags that are available across every sub-command. The `--json` flag, for example, outputs the tool's results in JSON format.



Sample JSON Output

This could then be consumed and parsed by a custom script to convert any findings into even more actionable intelligence. Given the sample TruffleHog JSON output above, let's say you want to extract information about each detected issue, specifically the commit, file, email, repository, and the detected AWS keys. You can use `jq` to accomplish this!

The `jq` command-line tool is a powerful and lightweight way to process and manipulate JSON data. It provides a convenient and efficient way to extract, transform, and filter JSON content, making it a valuable tool for working with JSON-based APIs, configuration files, and data processing.

Some of `jq`'s useful features are:

- Querying and Selecting Data
- Filtering and Transformation
- Prettifying Output
- Conditional Processing
- Combining with Other Unix Tools (i.e., `cat`, `grep`, `sed`)
- Scripting Support

The command below takes our TruffleHog JSON output and extracts commit, file, email, repository, and the detected AWS keys to display in a shortened JSON format.

```
cat trufflehog_output.json | jq -c '.SourceMetadata.Data.Git as $git | {commit: $git.commit, file: $git.file, 
```

```
└─$ cat trufflehog_output.json | jq -c '.SourceMetadata.Data.Git as $git | {
commit: $git.commit, file: $git.file, email: $git.email, repository: $git.re
pository, awsKey: .Raw}'

{"commit":"0416560b1330d8ac42045813251d85c688717eaf","file":"new_key","email
":"counter <hello@trufflesec.com>","repository":"https://github.com/truffles
ecurity/test_keys","awsKey":"AKIAQYLPMN5HHHFPZAM2"}
{"commit":"fbc14303ffbf8fb1c2c1914e8dda7d0121633aca","file":"keys","email":"
counter <counter@counters-MacBook-Air.local>","repository":"https://github.c
om/trufflesecurity/test_keys","awsKey":"AKIAYVP4CIPPERUVIFXG"}
{"commit":"77b2a3e56973785a52ba4ae4b8dac61d4bac016f","file":"keys","email":"
counter <counter@counters-MacBook-Air.local>","repository":"https://github.c
om/trufflesecurity/test_keys","awsKey":"https://admin:admin@the-internet.her
okuapp.com"}
```

jq Parsing

Another shared sub-command flag is `--only-verified`, which directs TruffleHog to check every potential credential that is detected against the API that it thinks it belongs to. This additional step can help eliminate false positives. For example, the AWS credential detector performs a `GetCallerIdentity` API call against the AWS API to verify if an AWS credential is active.

```
└─$ trufflehog github --org=trufflesecurity --only-verified
🐷🔑🐷  TruffleHog. Unearth your secrets. 🐷🔑🐷

2024-01-04T09:33:13-05:00       info-0  trufflehog      Completed enumeration
Found verified result 🐷🔑
Detector Type: AWS
Decoder Type: PLAIN
Raw result: A[redacted]G
Account: 5[redacted]3
User_id: A[redacted]Y
Arn: arn:aws:iam::595918472158:user/[redacted]
Commit: fbc14303ffbf8fb1c2c1914e8dda7d0121633aca
Email: counter <counter@counters-MacBook-Air.local>
File: keys
Line: 4
Link: https://github.com/trufflesecurity/test_keys/blob/fbc14303ffbf8fb1c2c1914e
Repository: https://github.com/trufflesecurity/test_keys.git
Timestamp: 2022-06-16 17:17:40 +0000
```

Sample Verified Key

Other flags are sub-command specific like `--since-commit` and `--max-depth`, which are available in the `git` command and control how far back and to what depth into commits the scan focuses on. These are useful to narrow the scope of the scan and incorporate it into your CI process to identify problems *before* they reach a distributed repository.

Now that we know what TruffleHog is, why it's important, and understand the basics of how it works, let's look at some real examples.

Web Application… err… Application

TruffleHog has been so successful in reviewing repositories, filesystems, cloud assets, and CI implementations that it has also been adapted (by third parties) into browser extensions (Chrome[2] and Firefox[3]) to scan web application code returned by a server for secrets too!

For example, the image below shows a React application that graciously returned numerous secrets for a company's CI/CD pipeline within the `main.js` file. Including GitHub, Bamboo, Polaris, AWS, and SonarQube secrets.

API keys for CI/CD pipeline

This issue is made worse by the file not requiring authentication to get — meaning anyone online could retrieve these keys. With a little extra legwork, and the help of GitHub's API, an attacker would discover the GitHub token allowed for full read-write to the organization's private GitHub. This could also permit user information for who issued the token, the organization's larger list of users, and repository enumeration.

GitHub Token Authorization Sample

The leaked AWS keys were also valid and could be abused using AWS's own cli tool[4].

My Code Has Secrets. What Now?

BHIS recommends taking the following steps when you encounter secrets in your (or your customer's) code:

- Remove all secrets.
- Remove the previous commit(s) in the repository's history that contained the secret.
- Periodically run open-source token scanning software such as TruffleHog.
- Review the CI/CD configurations.

What Next?

If you're interested in learning more advanced usage of TruffleHog, you can start by checking out their guide on GitHub (https://github.com/trufflesecurity/trufflehog#advanced-usage).

References

1. https://github.com/trufflesecurity/trufflehog ↩
2. https://chromewebstore.google.com/detail/trufflehog/bafhdnhjnlcdbjcdcnafhdcphhnfnhjc ↩
3. https://addons.mozilla.org/en-US/firefox/addon/trufflehog/ ↩

4. https://aws.amazon.com/cli/ ↩

Enjoyed this blog? Want to learn more?

Chris will be presenting live, online training during **The Most Offensive Con That Ever Offensived**!

Find more details here:

Advanced Offensive Tooling

Source: https://www.blackhillsinfosec.com/rooting-for-secrets-with-trufflehog/