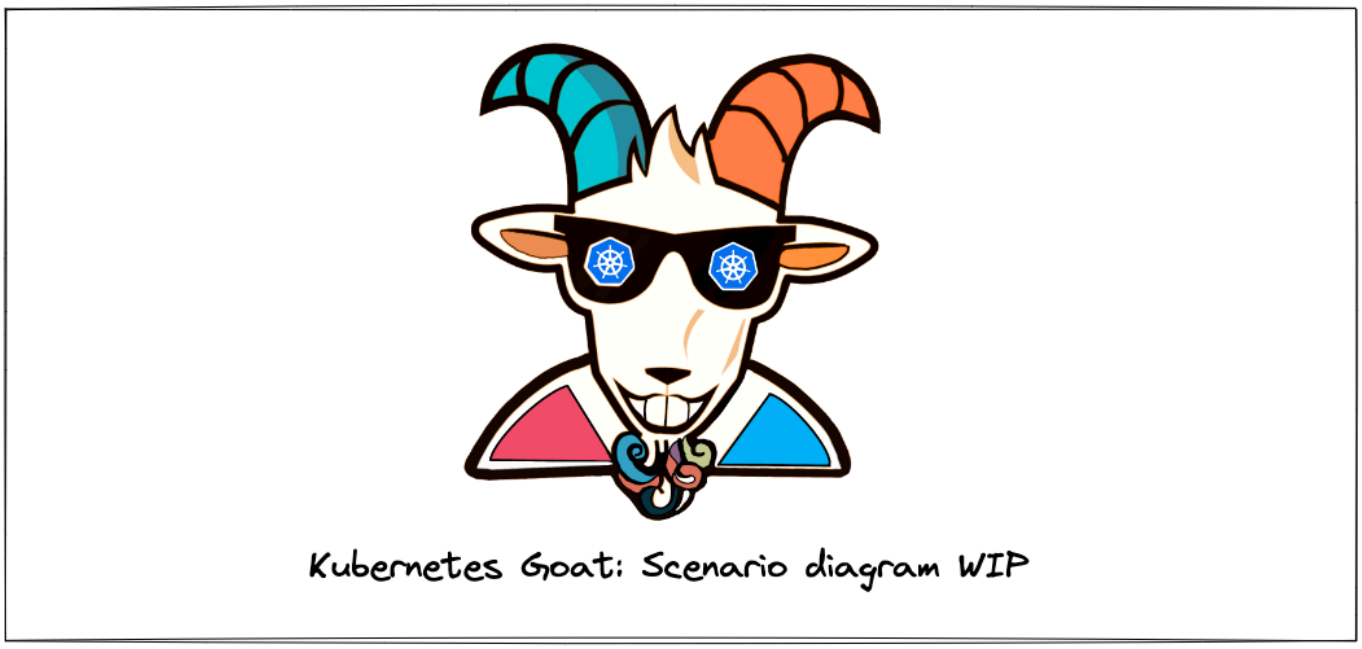
⎈ Hidden in layers

🙌 Overview

Most of the container images download and used on the internet are created by someone else. If we don't know how they get created (which means if we don't have `Dockerfile`) then I think we might be in trouble sometimes. In this scenario, we can see how to analyze the docker container image layers by using the built-in utilities and also some popular open-source utilities like dive to understand the container image layers.



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

1. How to explore, introspect and analyze docker container images
2. Using open source tools like dive to perform container image analysis
3. Able to work with standard command-line utilities

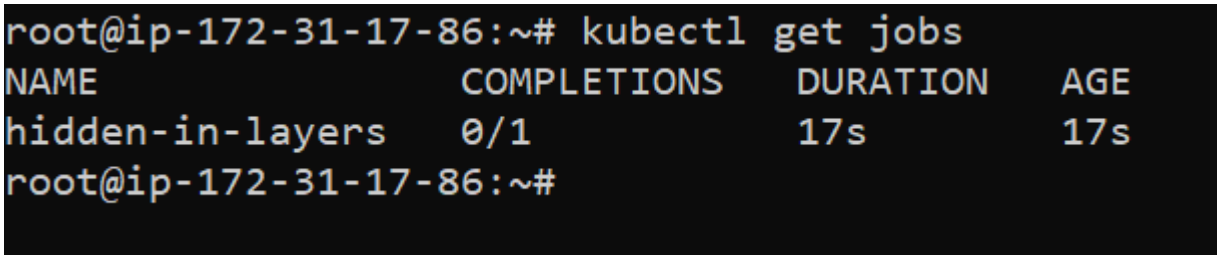
⚡ The story

Sensitive information disclosure is one of the most common vulnerabilities existing in the wild. Mishandling of passwords, private keys, tokens, etc in the containerization world is easy. Here in this scenario, we will analyze and identify one of such mishandled bad practices that leads to sensitive information disclosure.

INFO

- To get started with the scenario, run the following command and explore the `hidden-layers` job

```
kubectl get jobs
```



🎯 Goal

TIP

Find the `k8s_goat_flag` flag value in one of the hidden layers of the container, then you have completed this scenario.

☐ Hints & Spoilers

▸ ✨ Still trying to understand layers?

▸ ✨ Want to get the flag?

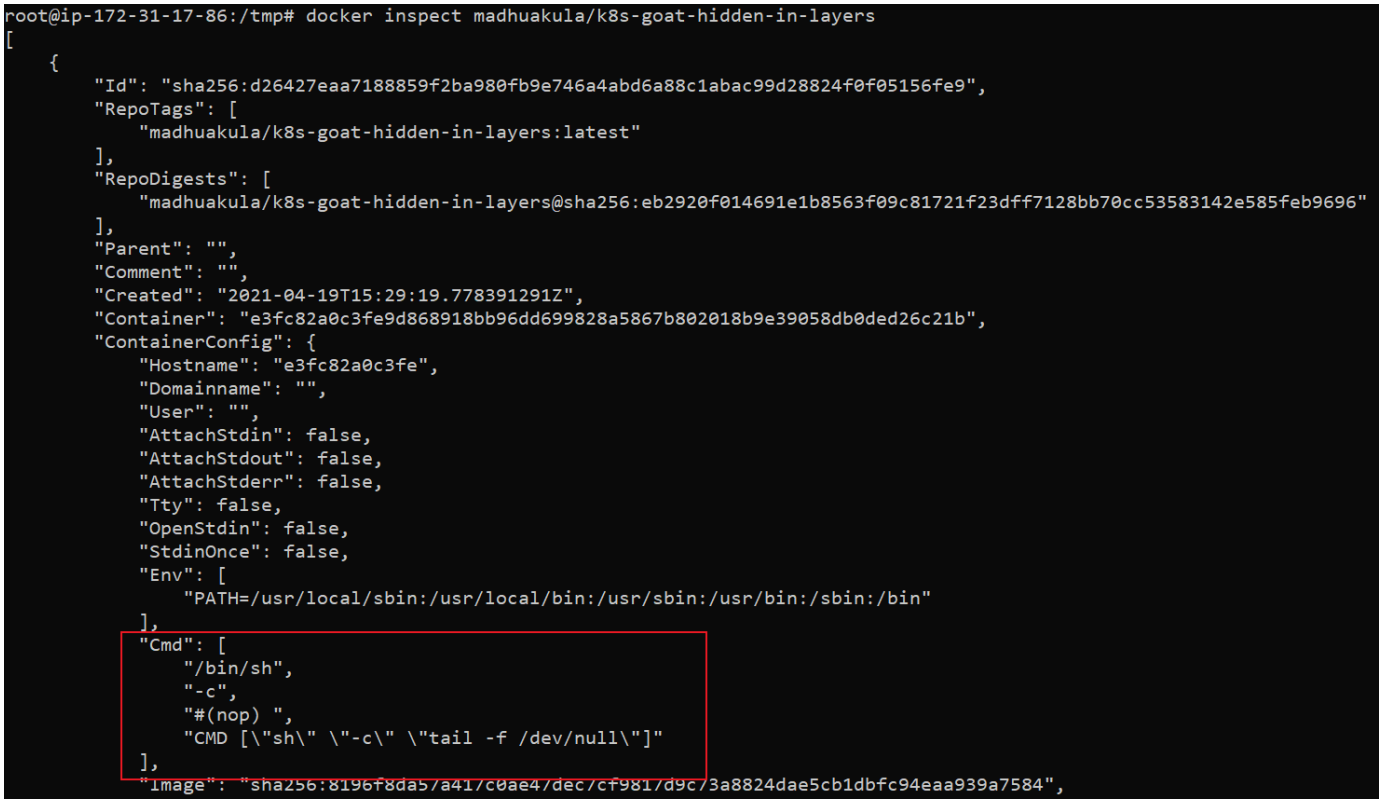
🎉 Solution & Walkthrough

INFO

Try exploring all files, environment variables, etc in the running container. Next, try to analyze the image used above with different tools to find exposed sensitive information.

- Docker CLI is an amazing tool with lots of features, let's start with inspecting the image

```
docker inspect madhuakula/k8s-goat-hidden-in-layers
```



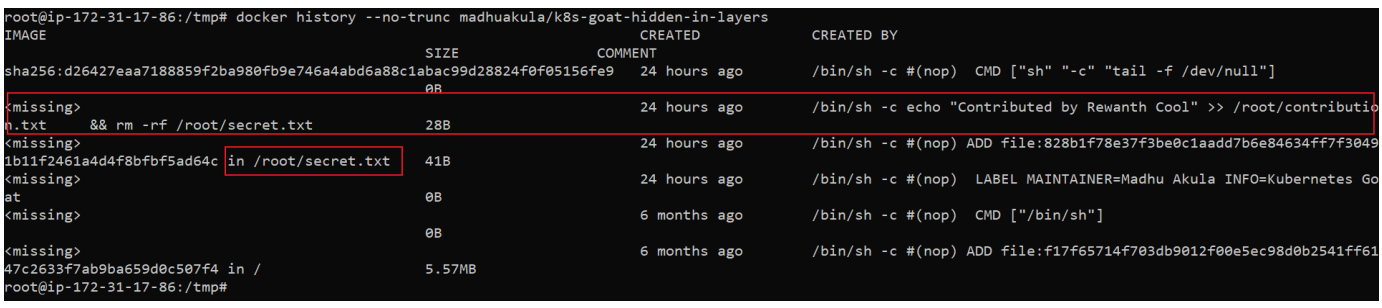
- We can observe the `cmd` section in the above output. It shows the default command executed by this image on startup. Though this shows some interesting information, it's not good enough for us

- Maybe it would be more helpful for us if we get to know how this image is built from scratch. For that, we need to analyze the `Dockerfile` of the image. If you have `Dockerfile`, it's good. If not, there are a few ways to analyze it

Method 1

- We can explore each layer by using the default `docker history` command

```
docker history --no-trunc madhuakula/k8s-goat-hidden-in-layers
```

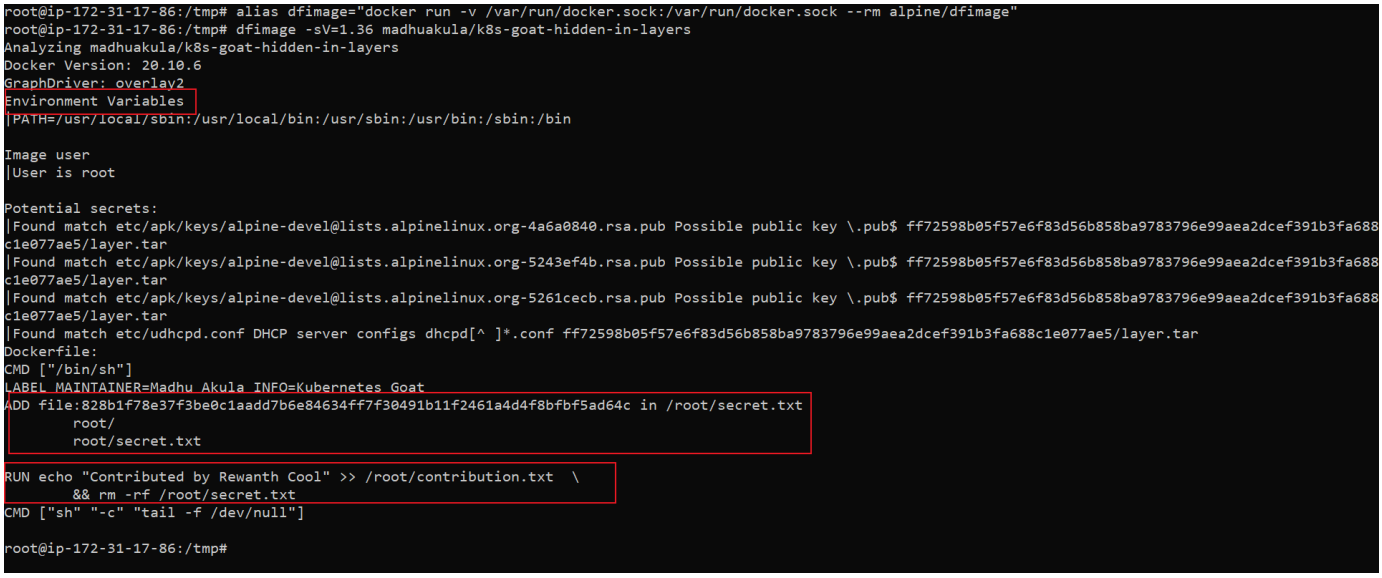


Method 2

- We can use the `dfimage` to generate a Dockerfile of any given image. First, we can set up that and perform it by running the following commands

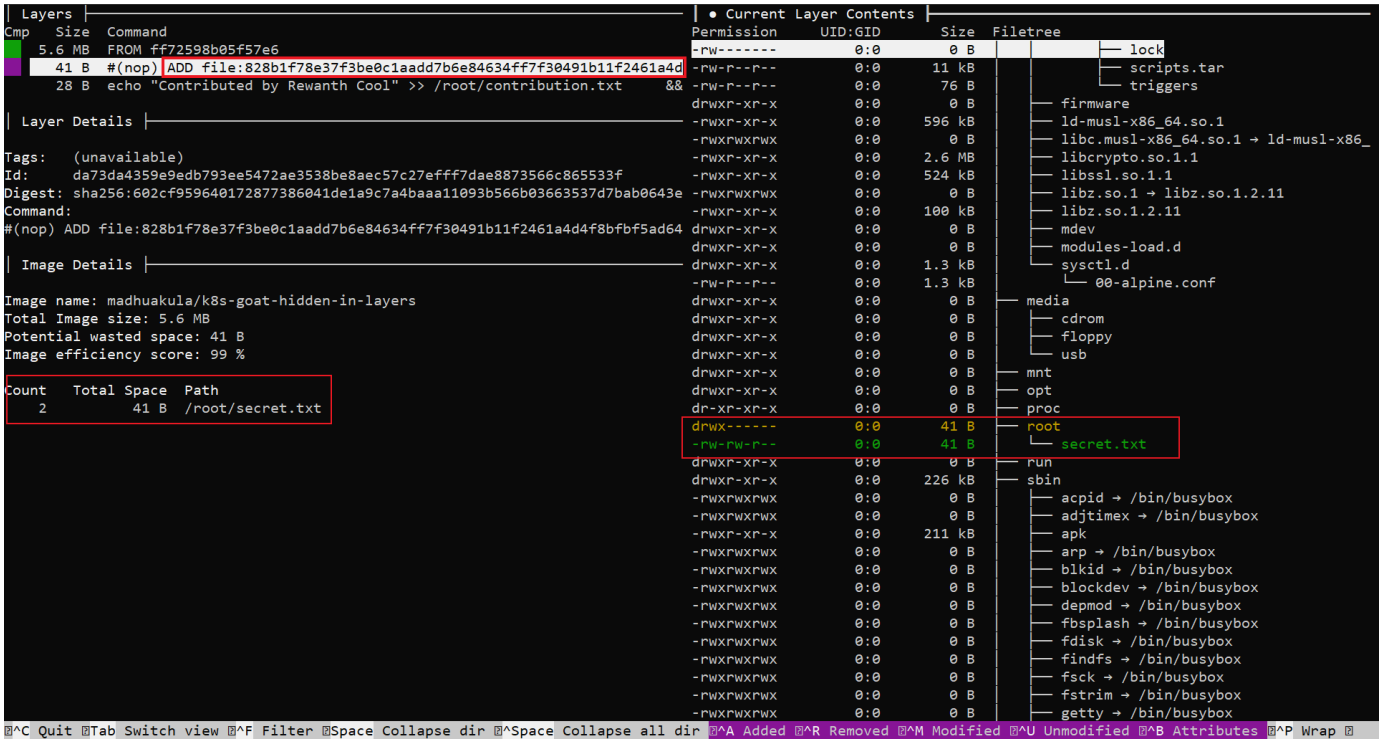
```
alias dfimage="docker run -v /var/run/docker.sock:/var/run/docker.sock --rm alpine/dfimage"
```

```
dfimage -sV=1.36 madhuakula/k8s-goat-hidden-in-layers
```



Method 3

- **`dive`** is an amazing tool that helps with analyzing each layer of an image



> **NOTE**
>
> From all the above analyses, we can see some significant changes in these two files, `/root/contributions.txt`, `/root/secret.txt`. The above methods cannot help us to read the contents of these files. Let's see if we can find these files in the running container.

```
root@ip-172-31-17-86:~# kubectl run hello --rm --restart=Never -it --image=madhuakula/k8s-goat-hidden-in-layers -- sh
If you don't see a command prompt, try pressing enter.
/ # ls /root -la
total 16
drwx------    1 root     root          4096 Apr 20 16:06 .
drwxr-xr-x    1 root     root          4096 Apr 20 16:06 ..
-rw-------    1 root     root            13 Apr 20 16:06 .ash_history
-rw-r--r--    1 root     root            28 Apr 19 15:29 contribution.txt
/ # cat /root/contribution.txt
Contributed by Rewanth Cool
/ # cat /root/secret.txt
cat: can't open '/root/secret.txt': No such file or directory
/ #
```

- We can't see `/root/secret.txt` as it is deleted from the next layers. We can recover the `/root/secret.txt` by leveraging the docker built-in command to export the docker image as a tar file

```
docker save madhuakula/k8s-goat-hidden-in-layers -o hidden-in-layers.tar
```

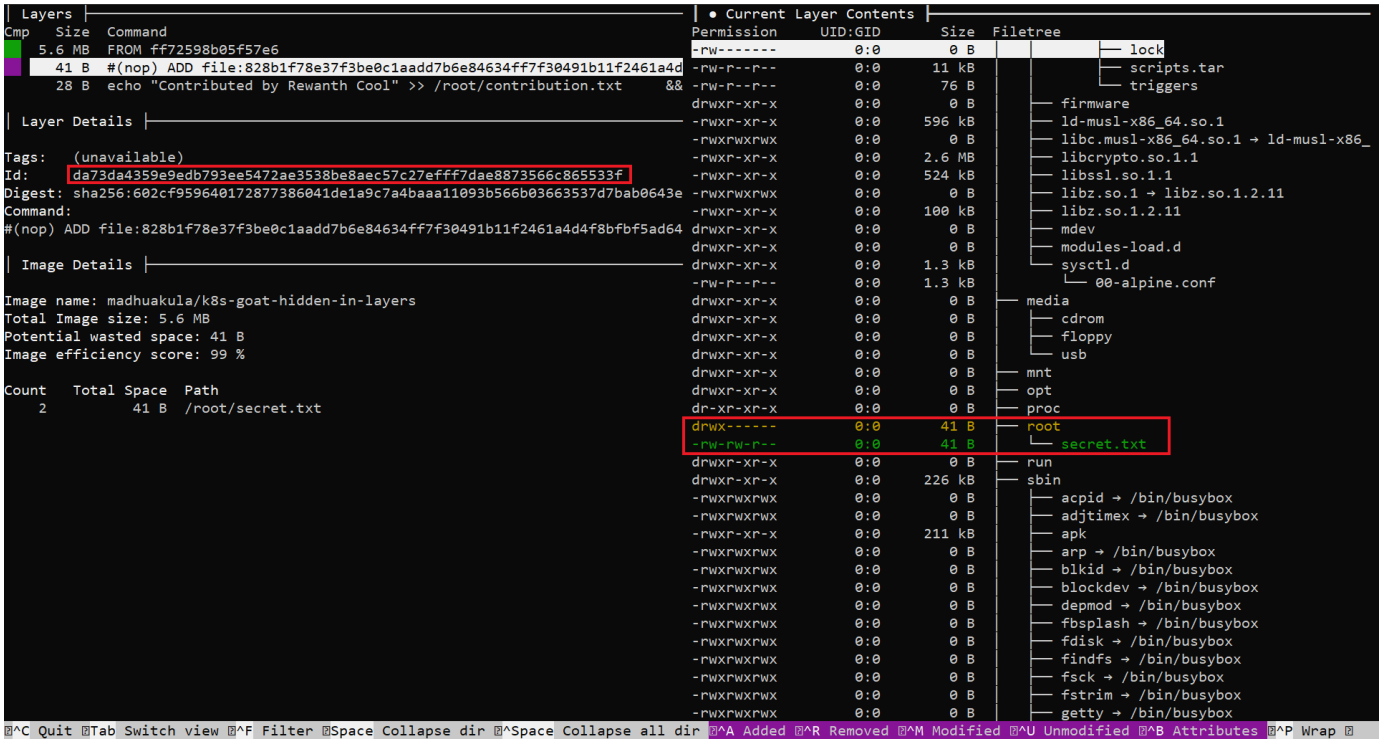
- Now we have the artifact and we can extract the tar file to explore the layers

```
tar -xvf hidden-in-layers.tar
```

```
root@ip-172-31-17-86:/tmp/hidden-in-layers# ls
hidden-in-layers.tar
root@ip-172-31-17-86:/tmp/hidden-in-layers# tar xvf hidden-in-layers.tar
d26427eaa7188859f2ba980fb9e746a4abd6a88c1abac99d28824f0f05156fe9.json
d81b961e28e91cdb9ea742b87bdc69685a6c4c76368a0a3de0934584a992e051/
d81b961e28e91cdb9ea742b87bdc69685a6c4c76368a0a3de0934584a992e051/VERSION
d81b961e28e91cdb9ea742b87bdc69685a6c4c76368a0a3de0934584a992e051/json
d81b961e28e91cdb9ea742b87bdc69685a6c4c76368a0a3de0934584a992e051/layer.tar
da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f/
da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f/VERSION
da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f/json
da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f/layer.tar
ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688c1e077ae5/
ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688c1e077ae5/VERSION
ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688c1e077ae5/json
ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688c1e077ae5/layer.tar
manifest.json
repositories
root@ip-172-31-17-86:/tmp/hidden-in-layers#
```

> TIP
>
> We can see each layer getting exported as a single tar file. We have 3 layers in this image, so we have 3 tar files. Since we have only 3 layers, it's easy to extract all of them and check the contents but that's not the conventional approach. What if we have hundreds of layers?

- Let's review the dive output again. In the below image, we saw a new file, `/root/secret.txt` is being created



- Observe the `Id` of that layer, `da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f`. Since, we have `/root/secret.txt` created in this layer, let's extract the tar file of this layer first

```
cd da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f
```

```
tar -xvf layer.tar
```

```
cat root/secret.txt
```

```
root@ip-172-31-17-86:/tmp/hidden-in-layers# ls
d26427eaa7188859f2ba980fb9e746a4abd6a88c1abac99d28824f0f05156fe9.json  ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688c1e077ae5  repositories
d81b961e28e91cdb9ea742b87bdc69685a6c4c76368a0a3de0934584a992e051       hidden-in-layers.tar
da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f       manifest.json
root@ip-172-31-17-86:/tmp/hidden-in-layers# cd da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f
root@ip-172-31-17-86:/tmp/hidden-in-layers/da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f# ls
VERSION  json  layer.tar
root@ip-172-31-17-86:/tmp/hidden-in-layers/da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f# tar xvf layer.tar
root/
root/secret.txt
root@ip-172-31-17-86:/tmp/hidden-in-layers/da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f# cat root/secret.txt
k8s-goat-3b7a7dc7f51f4014ddf3446c25f8b772root@ip-172-31-17-86:/tmp/hidden-in-layers/da73da4359e9edb793ee5472ae3538be8aec57c27efff7dae8873566c865533f#
```

- Hooray 😴, now we found Kubernetes Goat flag

References

- Digging into docker layers
- dive

Edit this page