ATTACK
DEFENSE

by PentesterAcademy

| **Name** | User Namespace Remapping |
|---|---|
| **URL** | https://attackdefense.com/challengedetails?cid=2309 |
| **Type** | Container Security : Security Private Docker Registry |

**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.

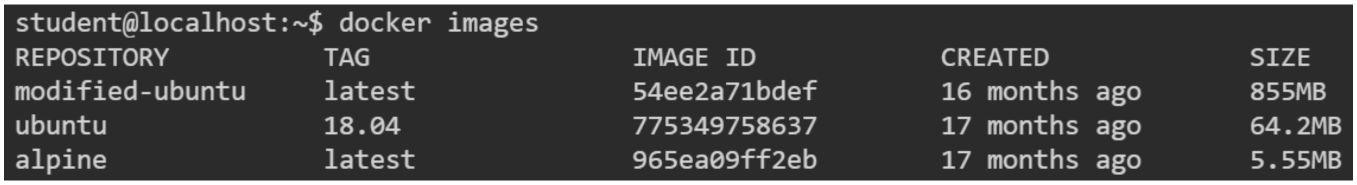
User namespace mapping is a protective measure to protect Docker host machines in case of container breakout or local privilege escalation. Read more here: https://docs.docker.com/engine/security/userns-remap/

**Objective:** Learn how to enable user namespace remapping and how it protects the Docker host in the event of container breakout or local privilege escalation!

**Solution:**

**Step 1:** The "student" user can run docker operations.

**Command:** docker images

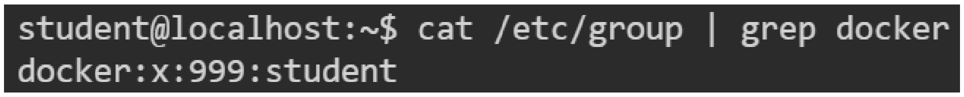


```
student@localhost:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
modified-ubuntu     latest              54ee2a71bdef        16 months ago       855MB
ubuntu              18.04               775349758637        17 months ago       64.2MB
alpine              latest              965ea09ff2eb        17 months ago       5.55MB
```

This means that the "student" user is a member of Docker group.

The same can be verified by checking the /etc/group file.

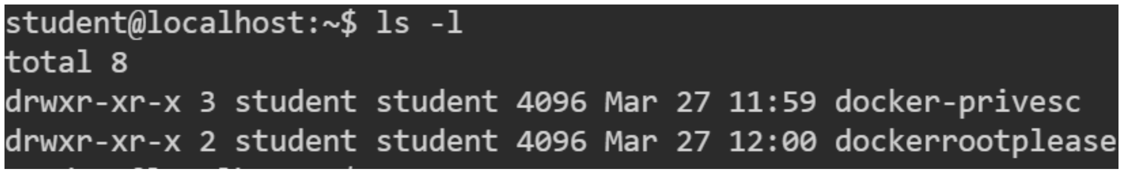
**Command:** cat /etc/group | grep docker



```
student@localhost:~$ cat /etc/group | grep docker
docker:x:999:student
```

**Step 2:** There are two projects present in the home directory of the student user. These can be used to perform the privilege escalation by abusing the Docker group membership.

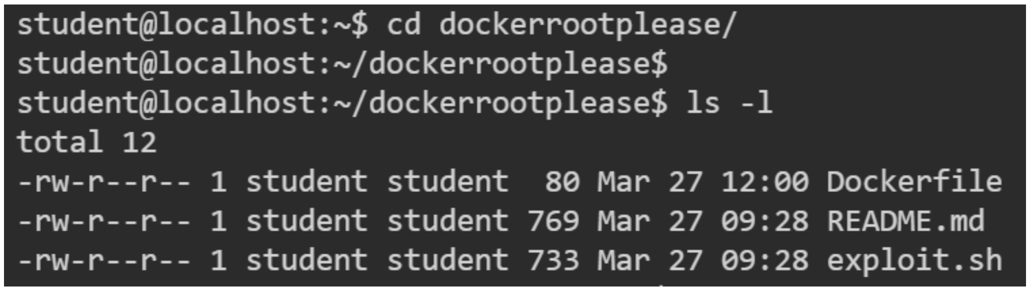
**Command:** ls -l



```
student@localhost:~$ ls -l
total 8
drwxr-xr-x 3 student student 4096 Mar 27 11:59 docker-privesc
drwxr-xr-x 2 student student 4096 Mar 27 12:00 dockerrootplease
```

**Step 3:** Switch to dockerrootplease directory and check the contents

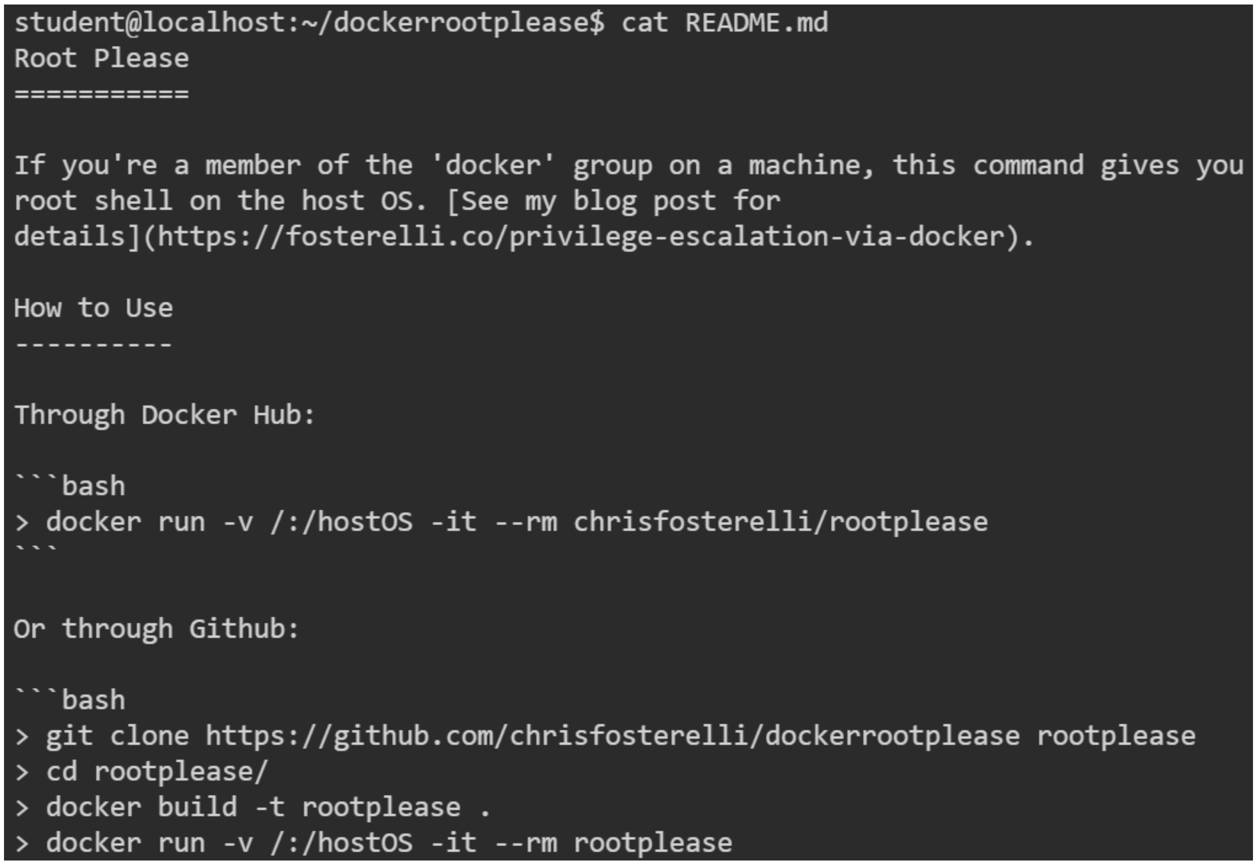
**Commands:**
cd dockerrootplease
ls -l



```
student@localhost:~$ cd dockerrootplease/
student@localhost:~/dockerrootplease$
student@localhost:~/dockerrootplease$ ls -l
total 12
-rw-r--r-- 1 student student  80 Mar 27 12:00 Dockerfile
-rw-r--r-- 1 student student 769 Mar 27 09:28 README.md
-rw-r--r-- 1 student student 733 Mar 27 09:28 exploit.sh
```

**Step 4:** Check the README.md file of this project.

**Command:** cat README.md



~~~
student@localhost:~/dockerrootplease$ cat README.md
Root Please
===========

If you're a member of the 'docker' group on a machine, this command gives you
root shell on the host OS. [See my blog post for
details](https://fosterelli.co/privilege-escalation-via-docker).

How to Use
----------

Through Docker Hub:

```bash
> docker run -v /:/hostOS -it --rm chrisfosterelli/rootplease
~~~

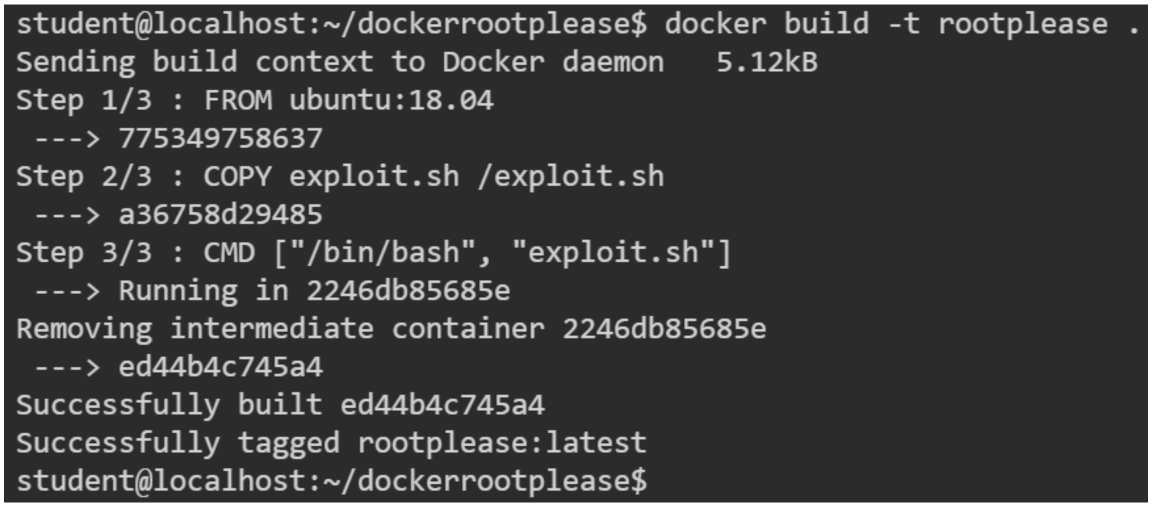
Or through Github:

```bash
> git clone https://github.com/chrisfosterelli/dockerrootplease rootplease
> cd rootplease/
> docker build -t rootplease .
> docker run -v /:/hostOS -it --rm rootplease
```

As per the instructions provided in the README file, build the image and run the image.

**Step 5:** Build the Docker image using the Dockerfile of the project and instruction provided in README.md

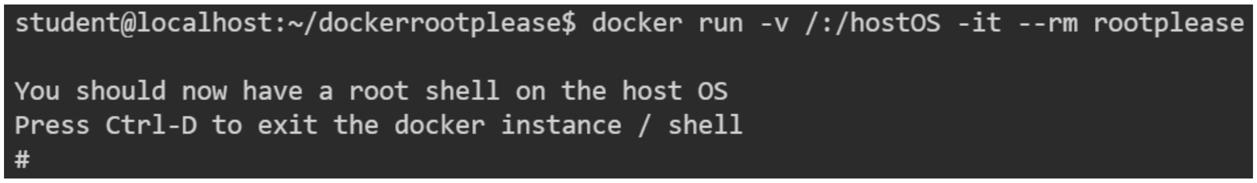
**Command:** docker build -t rootplease .



```
student@localhost:~/dockerrootplease$ docker build -t rootplease .
Sending build context to Docker daemon   5.12kB
Step 1/3 : FROM ubuntu:18.04
 ---> 775349758637
Step 2/3 : COPY exploit.sh /exploit.sh
 ---> a36758d29485
Step 3/3 : CMD ["/bin/bash", "exploit.sh"]
 ---> Running in 2246db85685e
Removing intermediate container 2246db85685e
 ---> ed44b4c745a4
Successfully built ed44b4c745a4
Successfully tagged rootplease:latest
student@localhost:~/dockerrootplease$
```

**Step 6:** Run the image while mounting the host filesystem on to the container.

**Command:** docker run -v /:/hostOS -it --rm rootplease



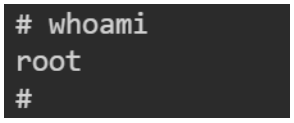
```
student@localhost:~/dockerrootplease$ docker run -v /:/hostOS -it --rm rootplease

You should now have a root shell on the host OS
Press Ctrl-D to exit the docker instance / shell
#
```

This results in a shell on the Docker host.

**Step 7:** Check the current user.

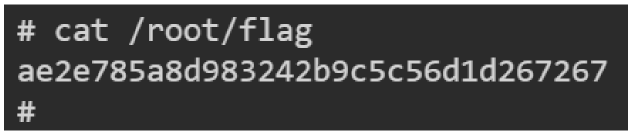
**Command:** whoami



```
# whoami
root
#
```

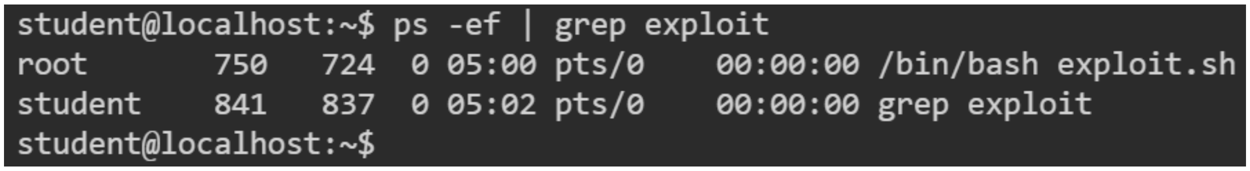
**Step 8:** Retrieve the flag kept in the /root directory of the Docker host machine.

**Command:** cat /root/flag


```
# cat /root/flag
ae2e785a8d983242b9c5c56d1d267267
#
```

Open another terminal. It is known that the rootplease image runs exploit.sh script inside it to do the breakout, switch to this new terminal and check the user running this script.

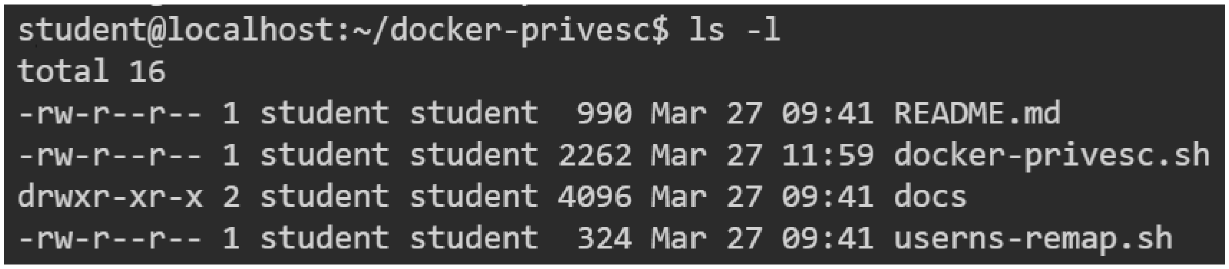
**Command:** ps -ef | grep exploit


```
student@localhost:~$ ps -ef | grep exploit
root       750   724  0 05:00 pts/0    00:00:00 /bin/bash exploit.sh
student    841   837  0 05:02 pts/0    00:00:00 grep exploit
student@localhost:~$
```

The exploit.sh is running as root.

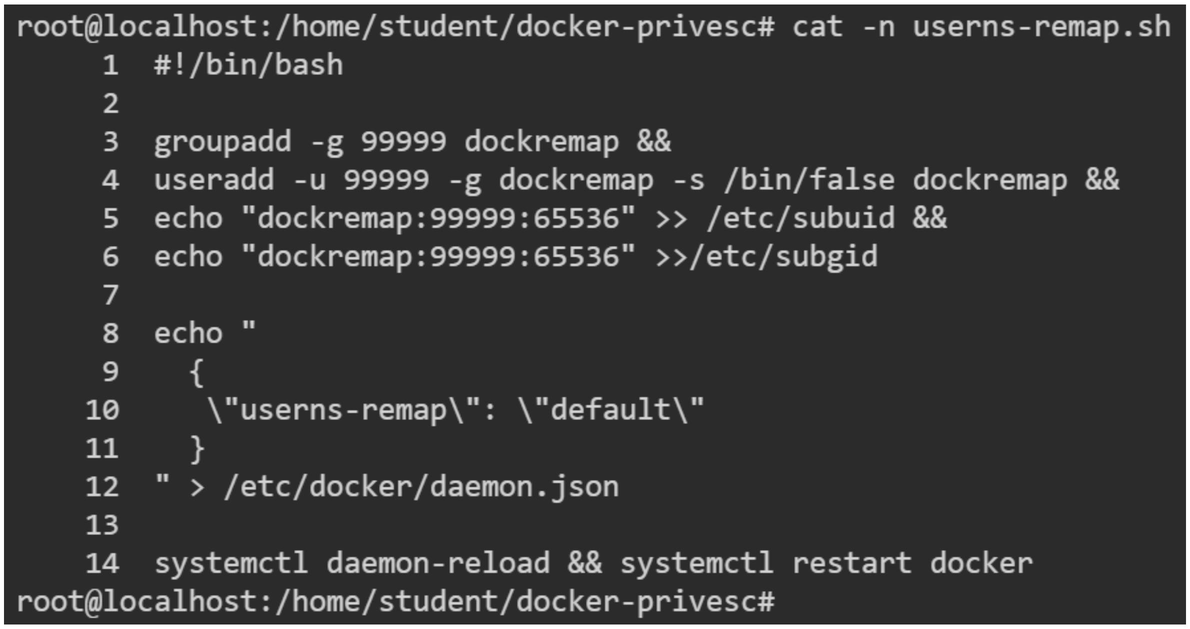
**Step 9:** Switch to docker-privesc directory and check the contents

**Commands:**
cd docker-privesc
ls -l


```
student@localhost:~/docker-privesc$ ls -l
total 16
-rw-r--r-- 1 student student  990 Mar 27 09:41 README.md
-rw-r--r-- 1 student student 2262 Mar 27 11:59 docker-privesc.sh
drwxr-xr-x 2 student student 4096 Mar 27 09:41 docs
-rw-r--r-- 1 student student  324 Mar 27 09:41 userns-remap.sh
```

**Step 10:** Check the userns-remap.sh script.

**Command:** cat -n userns-remap.sh



```
root@localhost:/home/student/docker-privesc# cat -n userns-remap.sh
     1  #!/bin/bash
     2
     3  groupadd -g 99999 dockremap &&
     4  useradd -u 99999 -g dockremap -s /bin/false dockremap &&
     5  echo "dockremap:99999:65536" >> /etc/subuid &&
     6  echo "dockremap:99999:65536" >>/etc/subgid
     7
     8  echo "
     9    {
    10     \"userns-remap\": \"default\"
    11    }
    12  " > /etc/docker/daemon.json
    13
    14  systemctl daemon-reload && systemctl restart docker
root@localhost:/home/student/docker-privesc#
```

**Explanation**

**Line 1:** Bash shebang

**Line 3:** Add a group “dockremap” with guid 99999

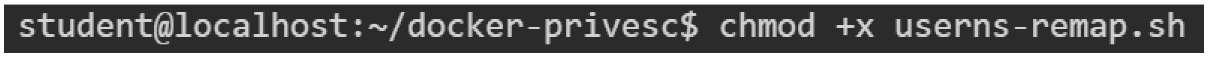
**Line 4:** Add a used “dockremap” with group “dockremap” and uid 99999

**Line 8 - 12:** Adding config to enable user namespace remapping to /etc/docker/daemon.json file

**Line 14:** Reloading configuration and restarting Docker daemon

Make the userns-remap.sh script executable.

**Command:** chmod +x  userns-remap.sh

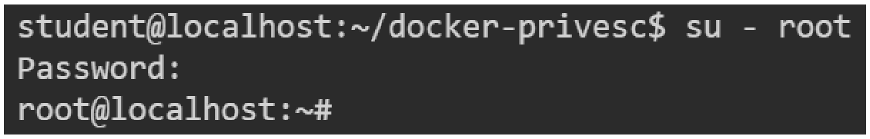


```
student@localhost:~/docker-privesc$ chmod +x userns-remap.sh
```

**Step 11:** This script is going to change Docker configuration file, so it needs to be executed with root privileges. Switch to the root account.

**Command:** su - root

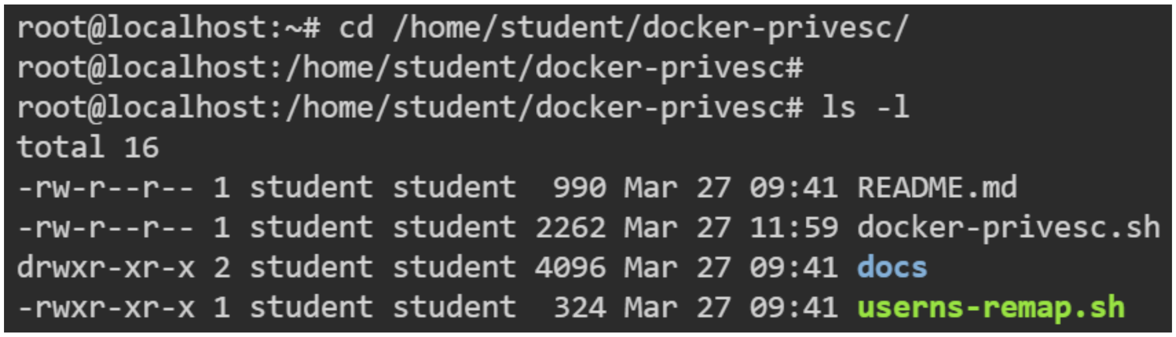
**Password of root user:** dockerpassword123



```
student@localhost:~/docker-privesc$ su - root
Password:
root@localhost:~#
```

Switch to docker-privesc directory.

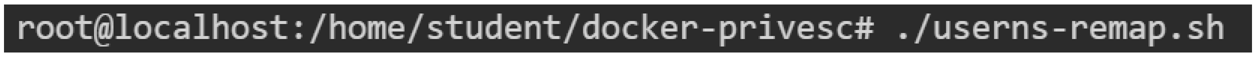
**Commands:**
cd /home/student/docker-privesc/
ls -l



```
root@localhost:~# cd /home/student/docker-privesc/
root@localhost:/home/student/docker-privesc#
root@localhost:/home/student/docker-privesc# ls -l
total 16
-rw-r--r-- 1 student student  990 Mar 27 09:41 README.md
-rw-r--r-- 1 student student 2262 Mar 27 11:59 docker-privesc.sh
drwxr-xr-x 2 student student 4096 Mar 27 09:41 docs
-rwxr-xr-x 1 student student  324 Mar 27 09:41 userns-remap.sh
```

And, execute the script

**Command:** ./userns-remap.sh

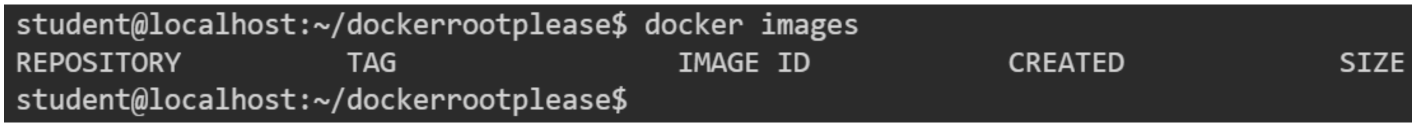


```
root@localhost:/home/student/docker-privesc# ./userns-remap.sh
```

This script will enable user namespace remapping and restart Docker daemon.

**Step 12:** List the Docker images present on the docker host machine.

**Command:** docker images



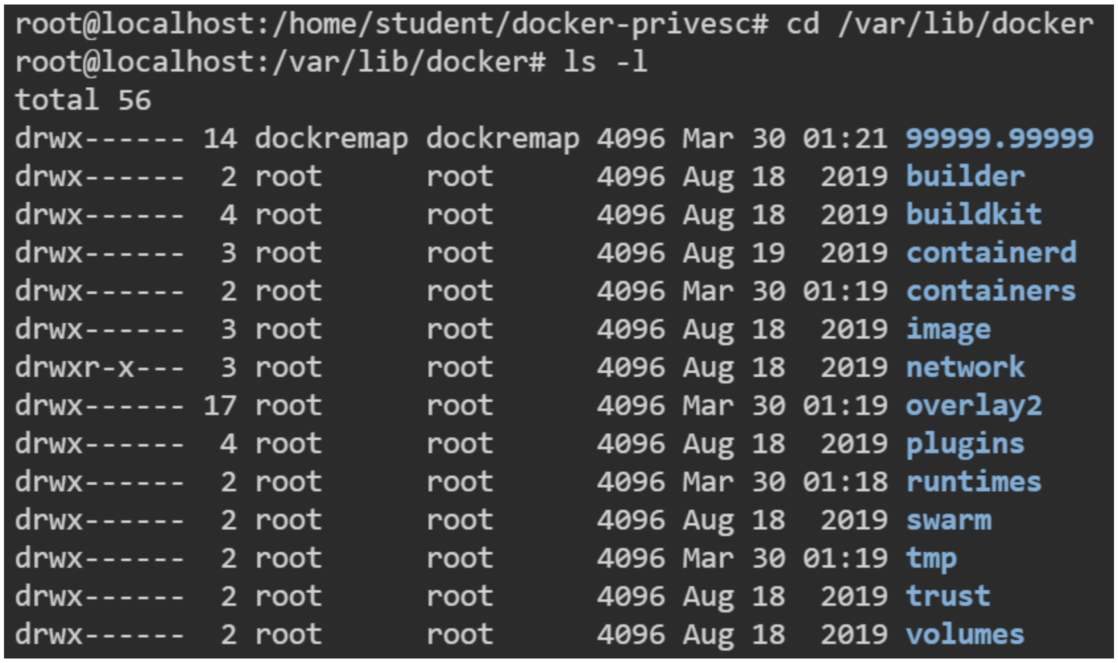
```
student@localhost:~/dockerrootplease$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
student@localhost:~/dockerrootplease$
```

The images are not visible.

This is because Docker (while running without user namespace remapping uses **/var/lib/docker/** directory whereas the after user namespace remapping the **/var/lib/docker/99999.99999/** is used.

**Step 13:** Change to /var/lib/docker/ and check the contents.

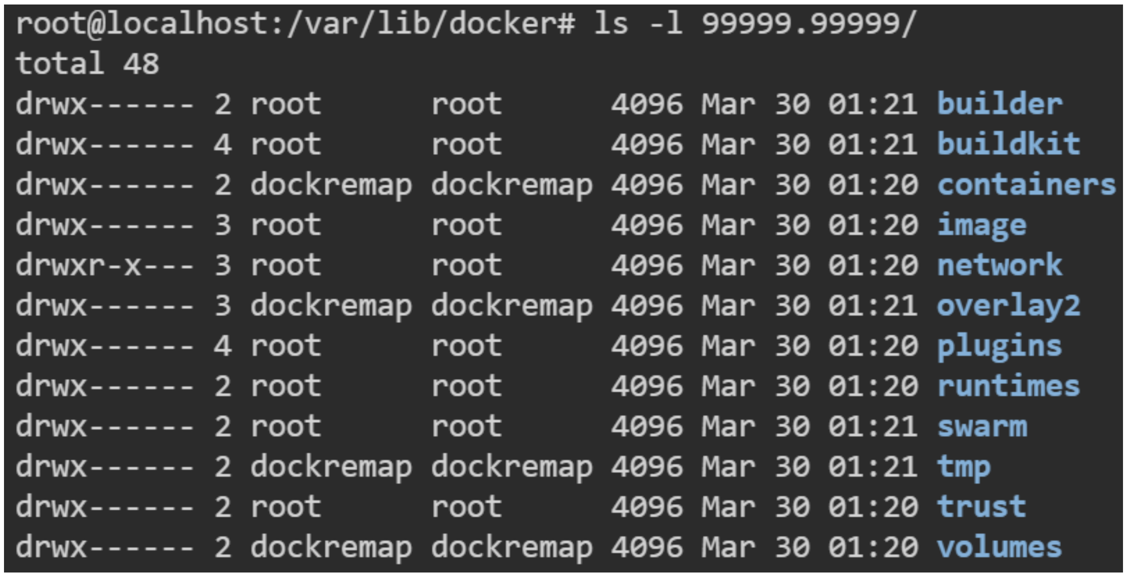
**Commands:**
cd /var/lib/docker/
ls -l



```
root@localhost:/home/student/docker-privesc# cd /var/lib/docker
root@localhost:/var/lib/docker# ls -l
total 56
drwx------ 14 dockremap dockremap 4096 Mar 30 01:21 99999.99999
drwx------  2 root      root      4096 Aug 18  2019 builder
drwx------  4 root      root      4096 Aug 18  2019 buildkit
drwx------  3 root      root      4096 Aug 19  2019 containerd
drwx------  2 root      root      4096 Mar 30 01:19 containers
drwx------  3 root      root      4096 Aug 18  2019 image
drwxr-x---  3 root      root      4096 Aug 18  2019 network
drwx------ 17 root      root      4096 Mar 30 01:19 overlay2
drwx------  4 root      root      4096 Aug 18  2019 plugins
drwx------  2 root      root      4096 Mar 30 01:18 runtimes
drwx------  2 root      root      4096 Aug 18  2019 swarm
drwx------  2 root      root      4096 Mar 30 01:19 tmp
drwx------  2 root      root      4096 Aug 18  2019 trust
drwx------  2 root      root      4096 Aug 18  2019 volumes
```

Also list the contents of /var/lib/docker/99999.99999/

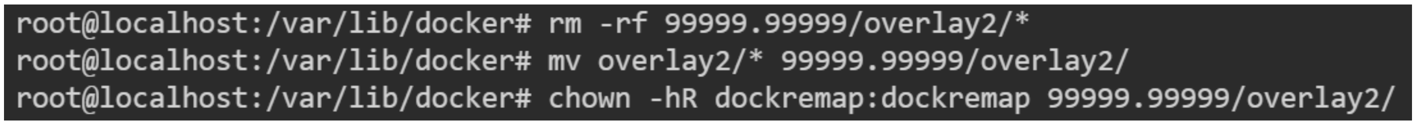
**Command:** ls -l 99999.99999


```
root@localhost:/var/lib/docker# ls -l 99999.99999/
total 48
drwx------ 2 root      root      4096 Mar 30 01:21 builder
drwx------ 4 root      root      4096 Mar 30 01:21 buildkit
drwx------ 2 dockremap dockremap 4096 Mar 30 01:20 containers
drwx------ 3 root      root      4096 Mar 30 01:20 image
drwxr-x--- 3 root      root      4096 Mar 30 01:20 network
drwx------ 3 dockremap dockremap 4096 Mar 30 01:21 overlay2
drwx------ 4 root      root      4096 Mar 30 01:20 plugins
drwx------ 2 root      root      4096 Mar 30 01:20 runtimes
drwx------ 2 root      root      4096 Mar 30 01:21 swarm
drwx------ 2 dockremap dockremap 4096 Mar 30 01:21 tmp
drwx------ 2 root      root      4096 Mar 30 01:20 trust
drwx------ 2 dockremap dockremap 4096 Mar 30 01:20 volumes
```

**Step 14:** Move relevant files from overlay2 and image directory to corresponding directories in 99999.99999. Also, make sure to change the ownership of these files from root to dockremap.

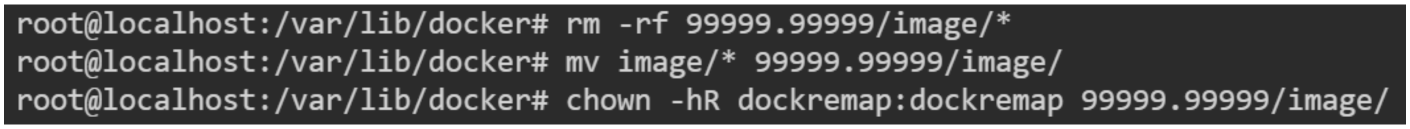
**For overlay2 directory**

**Commands:**
rm -rf 99999.99999/overlay2/*
mv overlay2/* 99999.99999/overlay2/
chown -hR dockremap:dockremap 99999.99999/overlay2/


```
root@localhost:/var/lib/docker# rm -rf 99999.99999/overlay2/*
root@localhost:/var/lib/docker# mv overlay2/* 99999.99999/overlay2/
root@localhost:/var/lib/docker# chown -hR dockremap:dockremap 99999.99999/overlay2/
```

**For image directory**

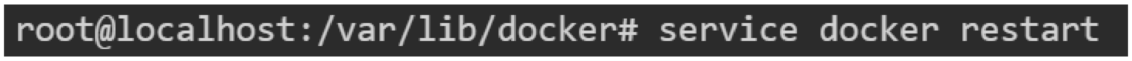
**Commands:**
rm -rf 99999.99999/image/*
mv image/* 99999.99999/image/
chown -hR dockremap:dockremap 99999.99999/image/



```
root@localhost:/var/lib/docker# rm -rf 99999.99999/image/*
root@localhost:/var/lib/docker# mv image/* 99999.99999/image/
root@localhost:/var/lib/docker# chown -hR dockremap:dockremap 99999.99999/image/
```

And, restart the Docker daemon.

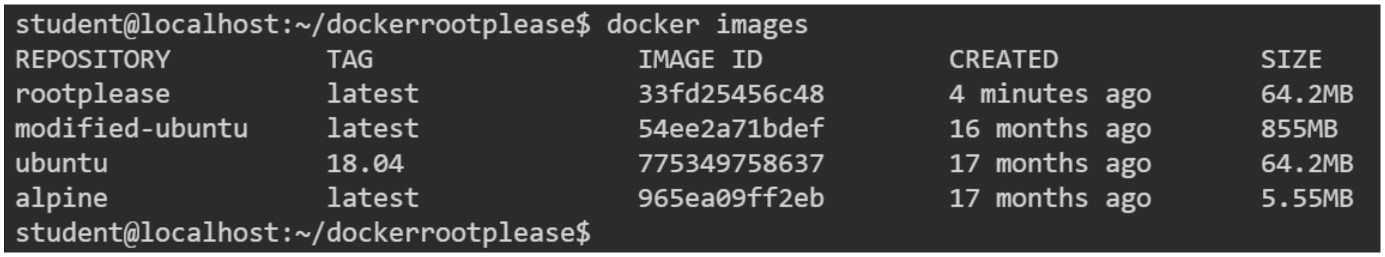
**Command:** service docker restart



```
root@localhost:/var/lib/docker# service docker restart
```

**Step 15:** Switch to the first terminal, and list the Docker images

**Command:** docker images

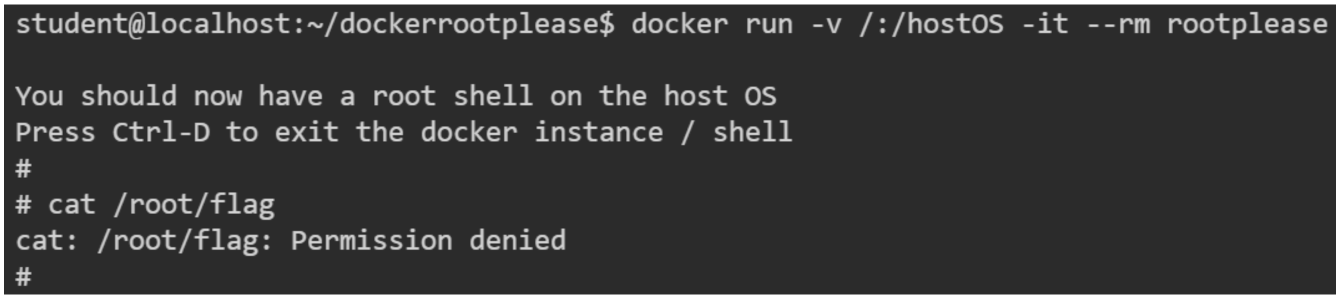


```
student@localhost:~/dockerrootplease$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
rootplease          latest              33fd25456c48        4 minutes ago       64.2MB
modified-ubuntu     latest              54ee2a71bdef        16 months ago       855MB
ubuntu              18.04               775349758637        17 months ago       64.2MB
alpine              latest              965ea09ff2eb        17 months ago       5.55MB
student@localhost:~/dockerrootplease$
```

Images are available now.

**Step 16:** Run the rootplease image and try to read the flag kept in the home directory of the root user (/root/flag) again.

**Commands:**
docker run -v /:/hostOS -it --rm rootplease
cat /root/flag



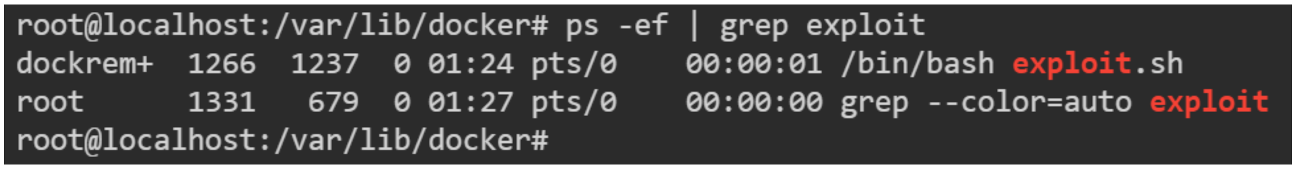
```
student@localhost:~/dockerrootplease$ docker run -v /:/hostOS -it --rm rootplease

You should now have a root shell on the host OS
Press Ctrl-D to exit the docker instance / shell
#
# cat /root/flag
cat: /root/flag: Permission denied
#
```

This time the root user (of the container) was not able to read the /root/flag.

**Step 17:** It is known that the rootplease image runs exploit.sh script inside it to do the breakout, switch to the second terminal and check the user running this script.

**Command:** ps -ef | grep exploit



```
root@localhost:/var/lib/docker# ps -ef | grep exploit
dockrem+  1266  1237  0 01:24 pts/0    00:00:01 /bin/bash exploit.sh
root      1331   679  0 01:27 pts/0    00:00:00 grep --color=auto exploit
root@localhost:/var/lib/docker#
```

The process is running as the user “dockremap” instead of user “root”.

In this manner, the user namespace remapping protects the Docker host machine in event of container breakout.

**References:**

1. Dockerrootplease (https://github.com/flast101/docker-privesc)
2. Docker-privesc (https://github.com/flast101/docker-privesc)
3. User namespace remapping (https://docs.docker.com/engine/security/userns-remap/)