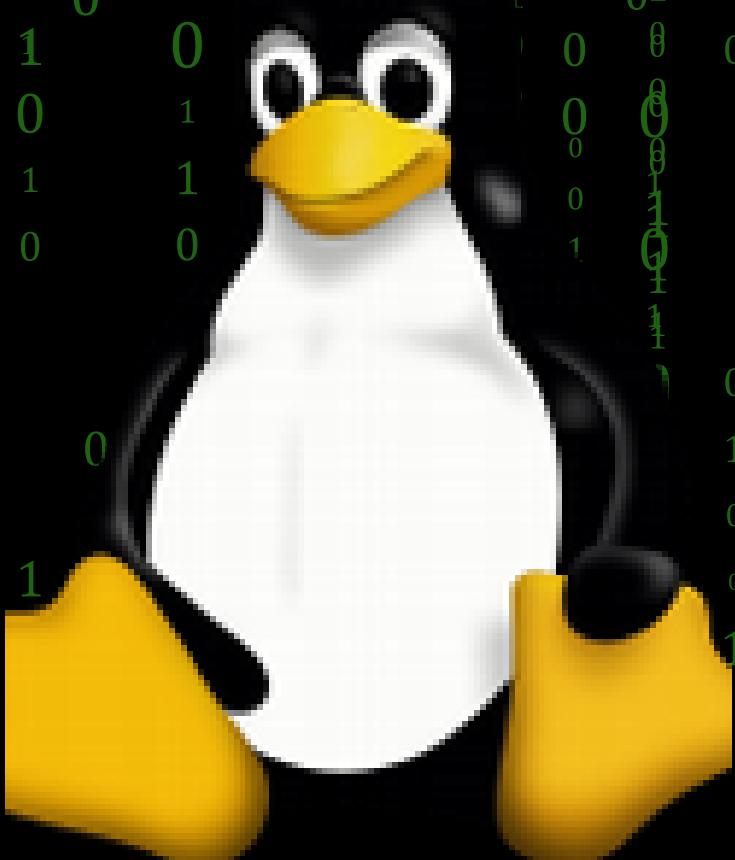


ESCALADA DE PRIVILEGIOS EN LINUX



By: Elisa Elias

¿QUE ES LA ESCALADA DE PRIVILEGIOS?

La escalada de privilegios implica el cambio desde una cuenta con autorizaciones menores a una cuenta con autorizaciones superiores. Desde un punto de vista más técnico, implica aprovechar una vulnerabilidad, un error de diseño o una configuración descuidada en un sistema operativo o una aplicación para obtener acceso a recursos que típicamente están limitados para los usuarios.

¿POR QUE ES IMPORTANTE PODER REALIZAR UNA ESCALADA DE PRIVILEGIOS?

- Restablecimiento de contraseñas
- Eludir los controles de acceso para poner en peligro los datos protegidos
- Edición de configuraciones de software
- Habilitación de la persistencia
- Cambiar los privilegios de los usuarios existentes (o nuevos)
- Ejecutar cualquier comando administrativo

PRIMERO, TENEMOS QUE ENTENDER

¿QUE SON LOS PRIVILEGIOS DE USUARIO?

- En sistemas Unix/Linux, los usuarios tienen diferentes niveles de privilegios.
- El usuario normal tiene acceso limitado, mientras que el superusuario (root) tiene acceso total al sistema.



Una vez que tenemos acceso inicial a una computadora con sistema operativo Unix, existen una serie de cosas que podemos averiguar de la maquina que nos van a ayudar a trazar una ruta para realizar nuestra escalada de privilegios.

- Información del Sistema
- Drives
- Software instalado
- Procesos
- Trabajos Programados/Cron Jobs
- Servicios
- Sockets
- D-Bus
- Red
- Usuarios
- Path
- Comandos SUDO y SUID
- Capacidades
- ACLs
- Sesiones de Shell abiertas
- Archivos intereantes
- Archivos grabables
- NFS

ENUMERACION: INFORMACIÓN DEL SISTEMA OPERATIVO

```
1 (cat /proc/version || uname -a ) 2>/dev/null  
2
```

Este comando intenta mostrar información sobre la versión del kernel del sistema operativo. El comando **cat /proc/version** muestra la versión del kernel que está siendo ejecutada, mientras que **uname -a** muestra información detallada sobre el sistema operativo.

```
1 lsb_release -a 2>/dev/null
```

Este comando intenta mostrar información detallada sobre la distribución del sistema utilizando **lsb_release**, que es un estándar de Linux para proporcionar información sobre la distribución del sistema.

```
1 cat /etc/os-release 2>/dev/null
```

Este comando intenta mostrar información sobre la distribución del sistema leyendo el archivo **/etc/os-release**, que es común en muchas distribuciones modernas de Linux.

/dev/null, Significa que cualquier error generado por los comandos anteriores se descarta y no se muestra al usuario.

```
1 cat /proc/version
```

Este comando mostrará información detallada sobre la versión del kernel, incluyendo el número de versión, la fecha de compilación y otra información relacionada con el kernel del sistema operativo. El contenido de este archivo nos ayudara a buscar exploits para el Kernel.

```
1 cat /etc/issue
```

Este archivo suele contener información sobre la distribución del sistema operativo y la versión específica.

```
1 echo $PATH  
2 #La salida puede verse asi  
3 /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin  
4
```

Esto significa que el sistema buscará comandos ejecutables en los directorios /usr/local/sbin, /usr/local/bin, /usr/sbin, /usr/bin, /sbin, y /bin, en ese orden.

```
1 (env || set) 2>/dev/null
```

Este comando muestra las variables de entorno y las funciones definidas en el shell actual.

(Podemos encontrar contraseñas o claves de API)

```
1 date 2>/dev/null #Fecha y hora actuales del SO
2 (df -h || lsblk) #Estatus del SO
3 lscpu #Informacion de CPU
4 lpstat -a 2>/dev/null #Impresoras disponibles
```

Este conjunto de comandos proporciona información útil sobre la fecha y la hora, el espacio en disco, la CPU y las impresoras disponibles en el sistema

```
1 if [ `which aa-status 2>/dev/null` ]; then
2     aa-status
3     elif [ `which apparmor_status 2>/dev/null` ]; then
4         apparmor_status
5     elif [ `ls -d /etc/apparmor* 2>/dev/null` ]; then
6         ls -d /etc/apparmor*
7     else
8         echo "Not found AppArmor"
9 fi
```

Este bloque de código verifica la presencia de AppArmor en el sistema y muestra su estado o ubicación en función de los comandos y directorios disponibles.

AppArmor es un sistema de seguridad basado en perfiles que funciona en sistemas operativos tipo Unix, como Linux. Su objetivo principal es reforzar el control de acceso a los recursos del sistema mediante la definición de políticas de seguridad específicas para las aplicaciones.

```
1 ls /dev 2>/dev/null | grep -i "sd"
2 cat /etc/fstab 2>/dev/null | grep -v "^#" | grep -Pv "\W*\#\" 2>/dev/null
3 grep -E "(user|username|login|pass|password|pw|credentials)[=:]" /etc/fstab /etc/mtab 2>/dev/null
```

Este comando realiza la búsqueda de dispositivos de almacenamiento, la revisión de configuraciones de montaje de archivos y la búsqueda de posibles credenciales de usuario o contraseñas en archivos del sistema. Las redirecciones 2>/dev/null eliminan los mensajes de error para una salida más limpia.

```
1 dpkg -l #Debian
```

Vamos a verificar si hay algún paquete o servicio que tenga una versión antigua y pueda ser vulnerado mediante un exploit.

```
1 ps #Enumerar procesos
2 ps -A #Ver todos los procesos en ejecución
3 ps auxjf #Ver el árbol de procesos
```

Estos comandos permiten a los usuarios ver los procesos en ejecución en el sistema y obtener información detallada sobre su jerarquía y estado.

SUDO

El comando sudo permite ejecutar programas con privilegios de administrador (root). En ocasiones, los administradores pueden necesitar dar a usuarios regulares ciertas capacidades específicas sin otorgarles acceso completo como administradores. Por ejemplo, un desarrollador puede necesitar instalar paquetes de software adicionales sin tener privilegios de root. En este caso, el administrador puede permitir al usuario ejecutar comandos específicos con privilegios de root mientras mantiene su nivel de acceso regular en el sistema.

Podemos comprobar nuestra situación actual de privilegios usando el siguiente comando.

```
| 1 sudo -l
```

También podemos ejecutar algunos programas con privilegios SUDO, el siguiente link te será útil para eso.

<https://gtfobins.github.io>

SUID

SUID (Set User ID) es un mecanismo de permisos en sistemas operativos Unix y Unix-like que permite que un programa sea ejecutado con los permisos del propietario del archivo en lugar de los del usuario que lo está ejecutando.

```
1 find / -type f -perm -04000 -ls 2>/dev/null
```

El comando busca archivos en todo el sistema de archivos que tienen el bit SUID establecido y lista información detallada sobre estos archivos.

El link anterior tambien te ayudara a que se filtraren los binarios que se sabe que son explotables cuando se establece el bit SUID.

<https://gtfobins.github.io>

CRON JOBS

Los cronjobs se definen mediante el uso del archivo cron, que contiene las programaciones de las tareas y los comandos que se ejecutarán.

```
1 crontab -l
2 ls -al /etc/cron* /etc/at*
3 cat /etc/cron* /etc/at* /etc/anacrontab /var/spool/cron/crontabs/root 2>/dev/null | grep -v "^#"
```

Este conjunto de comandos muestra las tareas programadas en el sistema, incluyendo las tareas cron y at, así como la configuración de anacron y las tareas del usuario root, mostrando solo las configuraciones activas y excluyendo los comentarios.

Los administradores de sistemas usan "cron jobs" para ejecutar scripts automáticamente en momentos específicos. Si eliminan un script pero olvidan borrar el cron job asociado, pueden dejar una brecha de seguridad. Si la ubicación del script no está especificada en el cron job, el sistema buscará el script en las ubicaciones definidas por la variable PATH, lo que podría permitir la ejecución de scripts no deseados.

PATH

La variable \$PATH generalmente incluye directorios como /usr/bin, /bin, /usr/sbin, /sbin, /usr/local/bin, entre otros, donde se encuentran los programas ejecutables comunes del sistema.

```
1 sudo awk 'BEGIN {system("/bin/sh")}'  
2 sudo find /etc -exec sh -i \;  
3 sudo tcpdump -n -i lo -G1 -w /dev/null -z ./runme.sh  
4 sudo tar c a.tar -I ./runme.sh a  
5 ftp>!/bin/sh  
6 less>! <shell_comand>
```

Estos son comandos que podrían utilizarse con el comando "sudo" para ejecutar tareas con privilegios de administrador

Si descubres que puedes escribir dentro de alguna carpeta de la variable \$PATH, es posible que puedas escalar privilegios creando una puerta trasera dentro de la carpeta donde tienes permisos de escritura, con el nombre de algún comando que será ejecutado por otro usuario (idealmente root) y que no proviene de una carpeta ubicada antes que tu carpeta con permisos de escritura en la variable \$PATH.

NFS

NFS opera en un modelo cliente-servidor, donde un servidor NFS comparte uno o más directorios o sistemas de archivos y los clientes NFS pueden montar estos sistemas de archivos remotos en sus propios sistemas para acceder a los archivos compartidos.

Cuando se configura un sistema de archivos compartido a través de NFS (Network File System), hay opciones específicas que pueden tener un gran impacto en la seguridad del sistema.

no_root_squash: Esta opción en el archivo /etc/exports permite al usuario root en un cliente NFS acceder a los archivos en el servidor NFS como si fuera el usuario root local en el servidor. Esto significa que el cliente root tiene el mismo nivel de control sobre los archivos en el servidor NFS que el usuario root en la propia máquina del servidor. Sin embargo, esto también puede ser peligroso, ya que podría permitir a un usuario malintencionado modificar archivos críticos en el servidor NFS desde un cliente remoto.

Ejemplo: Si un directorio está configurado con no_root_squash y un usuario tiene acceso root en un cliente NFS, puede modificar archivos críticos en el servidor NFS, lo que podría comprometer la integridad del sistema.

no_all_squash: Similar a no_root_squash, pero se aplica a los usuarios que no son root. Por ejemplo, si un usuario tiene una cuenta como "nobody" en el cliente NFS y el servidor NFS tiene la opción no_all_squash configurada, el usuario "nobody" en el cliente podría acceder y modificar archivos en el servidor NFS con los mismos permisos que tendría en el cliente.

Ejemplo: Si un usuario "nobody" tiene acceso a un cliente NFS y el servidor NFS está configurado con no_all_squash, el usuario "nobody" podría modificar archivos en el servidor NFS como lo haría en el cliente, lo que podría conducir a problemas de seguridad.

COMANDOS UTILES

- **sudo:**

- Permite a los usuarios ejecutar comandos con los privilegios de otro usuario, generalmente el superusuario (root).
- Sintaxis: sudo [comando]

- **su:**

- Permite cambiar al usuario superusuario (root) o a otro usuario.
- Sintaxis: su [usuario]

- **chmod:**

- Cambia los permisos de archivos y directorios.
- Sintaxis: chmod [permisos] [archivo/directorio]

- **chown:**

- Cambia el propietario y/o el grupo de un archivo o directorio.
- Sintaxis: chown [usuario:grupo] [archivo/directorio]

- **setuid/setgid:**

- Algunos programas tienen el bit setuid o setgid activado, lo que les permite ejecutarse con los privilegios del propietario o del grupo propietario, respectivamente.
- Para ver qué programas tienen estos bits activados, puedes utilizar el comando find junto con ls o stat.

- **/etc/sudoers:**

- Archivo de configuración que define qué usuarios pueden ejecutar qué comandos con sudo.
- Debes tener cuidado al modificar este archivo, ya que un error puede comprometer la seguridad del sistema.

- **find:**

- El comando find se utiliza para buscar archivos y directorios en el sistema de archivos.
- Es útil durante la escalada de privilegios para buscar archivos que puedan tener configuraciones inseguras o permisos incorrectos.
- Sintaxis básica: find [ruta] [criterios de búsqueda]

- **grep:**

- grep es una herramienta de línea de comandos que se utiliza para buscar patrones en archivos de texto.
- Puede ayudar a buscar información específica dentro de archivos que podrían revelar detalles importantes durante una escalada de privilegios.
- Sintaxis básica: grep [patrón] [archivo(s)]

- **ps:**

- El comando ps muestra información sobre los procesos en ejecución en el sistema.
- Puede ser útil durante la escalada de privilegios para identificar procesos sospechosos o aquellos que podrían ser explotados.
- Sintaxis básica: ps aux

- **netstat:**

- netstat muestra información sobre las conexiones de red, tablas de enrutamiento, estadísticas de interfaces, entre otros.
- Puede ayudar a identificar conexiones de red inusuales que podrían indicar actividad maliciosa.
- Sintaxis básica: netstat -tuln (muestra las conexiones TCP y UDP que están escuchando)

- **Isof:**
 - Isof muestra una lista de todos los archivos abiertos por procesos en el sistema.
 - Puede ser útil para identificar qué archivos están siendo utilizados por procesos específicos, lo que puede revelar información valiosa durante una escalada de privilegios.
 - Sintaxis básica: Isof -i (muestra conexiones de red abiertas)
- **sudoedit:**
 - sudoedit permite a los usuarios editar archivos como superusuario utilizando sudo.
 - A diferencia de sudo, sudoedit abre un editor de texto y realiza una copia temporal del archivo antes de editarlo, lo que puede ayudar a prevenir cambios accidentales o maliciosos.
 - Sintaxis básica: sudoedit [archivo]
- **strace:**
 - strace se utiliza para rastrear las llamadas al sistema y las señales que realiza un proceso.
 - Puede ser útil para entender el comportamiento de un programa y encontrar posibles vulnerabilidades o puntos de intervención durante la escalada de privilegios.
 - Sintaxis básica: strace [comando]

HERRAMIENTAS DE ENUMERACION

- **LinPeas:**

<https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS>

- **LinEnum:**

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

- LES (Linux Exploit Suggester):

<https://github.com/mzet-/linux-exploit-suggester>

- **Enumeración inteligente de Linux:**

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

- **Comprobador de privilegios de Linux:**

<https://github.com/linted/linuxprivchecker>