Tema 04 - Linux: comandos II

Parte 5

0. Usuario sudoers

sudo permite implementar un control de acceso altamente granulado de qué usuarios ejecutan qué comandos. Si un usuario normal desea ejecutar un comando de root (o de cualquier otro usuario), sudo verifica en su lista de permisos y si está permitido la ejecución de ese comando para ese usuario, entonces sudo se encarga de ejecutarlo. Es decir, sudo es un programa que, basado en una lista de control (/etc/sudoers), permite (o no) la ejecución al usuario que lo invocó sobre un determinado programa propiedad de otro usuario, generalmente del administrador del sistema 'root'.

1. Permisos especiales

Además de los permisos ya vistos, existen tres tipos de permisos considerados especiales y con una funcionalidad concreta para cada uno de ellos.

- **sticky bit:** con valor 1.
- **setgid:** con valor 2.
- **setuid:** con valor 4.

1. Permisos especiales

Permiso sticky bit

Este permiso sólo cobra sentido en directorios. Si intentas activar el permiso sticky bit en un fichero, el sistema lo ignora.

La función de este permiso es limitar quién puede borrar un fichero de un directorio, de tal forma que solo el propietario del fichero pueda, aunque todos los demás usuarios tengan permiso de escritura sobre el fichero.

El directorio /tmp disponible en todos los sistemas Unix, tiene activado este permiso especial.

1. Permisos especiales

Permiso sticky bit

```
miguel@mig-ubuntu-dk:/tmp$ ls -ld
drwxrwxrwt 24 root root 4096 feb 13 08:37 .
miguel@mig-ubuntu-dk:/tmp$
```

ls –ld
Muestra los atributos del directorio indicado, en lugar de los atributos del contenido del directorio.

Como se puede ver en la salida del comando el último bit, w en lugar de una x o – es una t. Esta es la forma que tiene Unix para representar que el permiso sticky bit de un directorio está activado.

El directorio /tmp es un directorio del sistema que se usa de forma temporal, y en el que todos los usuarios del sistema pueden escribir.

Si el permiso sticky bit no estuviera activado, un fichero creado por un usuario, puede ser borrado por otro usuario. Aquí es donde tiene sentido el sticky bit, porque **con este permiso activas que todo el mundo pueda escribir dentro del directorio temporal, pero solo podrá borrar el fichero el propietario del mismo.**

1. Permisos especiales

Permiso sticky bit

Si este permiso no estuviera activado y un usuario intenta borrar un fichero de otro en /tmp, lo que el sistema ve es que se va a intentar borrar un fichero sobre el que no se tiene permiso de escritura, pero como si que se tiene permiso de escritura sobre el directorio, es posible la operación de borrado aunque muestre mensaje de aviso pidiendo confirmación. Si no estuviese activado el sticky bit el fichero se hubiese borrado. En cambio el sticky bit previene que un usuario mal intencionado o despistado borre un fichero que no le pertenezca, así que el sistema cancela la operación de borrado y muestra un mensaje de error.

1. Permisos especiales

Permiso sticky bit

Si un usuario intenta borrar un fichero de otro sin tener activado el permiso sticky bit, pueden pasar dos cosas:

- **El usuario no tiene permisos de escritura:** el sistema comprueba que el usuario no tiene permisos de escritura. El sistema avisa del error.
- **El usuario tiene permisos de escritura:** el sistema comprueba que el usuario tiene permisos de escritura. Si no estuviese activado el sticky bit el fichero se borra.

Por tanto, el sticky bit previene que un usuario mal intencionado o despistado borre un fichero que no le pertenezca, así que el sistema cancela la operación de borrado y muestra un mensaje de error.

1. Permisos especiales

Permiso sticky bit

Cuando se lo asignamos a un directorio, conseguimos que los elementos (ficheros y/o directorios) que haya en ese directorio sólo pueden ser **renombrados o borrados** por:

- El propietario del elemento contenido en el directorio.
- El propietario del directorio.
- El usuario root.

El sticky bit prevalece sobre los permisos normales, lo que quiere decir que tan sólo podrán renombrar o borrar los elementos los usuarios mencionados, aunque el resto de usuarios tenga permisos de escritura.

1. Permisos especiales

Permisos setgid y setuid

Estos dos permisos se aplican sobre ficheros ejecutables.

Vamos a centrarnos en el programa passwd: este programa cambia la contraseña pero, ¿cómo es posible que el comando pueda escribir en el fichero de sistema donde se almacenan las contraseñas?. Hay dos opciones:

- Dando permisos de escritura al fichero de contraseñas para todos los usuarios, algo bastante peligroso.
- Utilizando los permisos setuid y setgid.

1. Permisos especiales

Permiso setuid

El permiso setuid consiste en dar permisos sobre un fichero ejecutable y lo que hace es ejecutar el programa con la identidad del propietario del fichero, en lugar de hacerlo con la identidad del usuario que lo ejecuta.

Estos permisos se utilizan principalmente en /etc/passwd y /usr/bin/passwd.

```
miguel@mig-ubuntu-dk:/tmp$ ls -l /etc/passwd
-rw-r--r-- 1 root root 2890 feb  3 12:04 /etc/passwd
miguel@mig-ubuntu-dk:/tmp$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 59976 nov 24 13:05 /usr/bin/passwd
```

El propietario del archivo tiene permisos de lectura y escritura, mientras que los miembros del grupo de usuarios y otros miembros solo tienen permisos de visualización.

Tanto los usuarios root como los usuarios normales pueden usar el comando / usr / bin / passwd someuser para modificar el archivo / etc / passwd,

1. Permisos especiales

Sabemos que necesitamos modificar la contraseña de un usuario en el sistema.

Es comprensible que el usuario root tenga permiso de escritura en / etc / passwd.

Pero para que los usuarios normales también puedan hacerlo es necesario utilizar el setuid.

Así, los usuarios normales tendrán autoridad de root cuando ejecuten passwd, de modo que puedan modificar el archivo / etc / passwd.

Su marca es: s, que aparecerá en lugar de x, *por ejemplo: -rwsr-xr-x*.

- **setuid:** cambia la identidad del usuario con el que se ejecuta un proceso.
- **setgid:** cambia la identidad del grupo con el que se ejecuta el proceso.

En un fichero pueden estar activos uno de ellos, los dos o ninguno.

1. Permisos especiales

Permiso setuid

Cuando un usuario ejecuta el comando passwd, a todos los efectos, es como si el usuario root con grupo sys hubiese ejecutado el comando. Por eso puede escribir sobre el fichero de contraseñas del sistema, pero de una manera controlada. De esta forma el fichero de contraseñas solo necesita permisos de escritura para el usuario root.

La forma de añadir o quitar este permiso varia:

- **De forma octal:** se añade un nuevo dígito, a la izquierda de los otros tres, pasando a ser un número de 4 cifras. Si se omite se interpreta como un 0.
- **De forma no octal:** se utiliza el carácter s para representar el permiso setuid y setgid.

-rwsr-sr-x
¿cómo se representa este permiso en octal?

-rwsr-sr-x

¿cómo se representa este permiso en octal?

- **usuario**: tiene permisos de lectura, escritura y ejecución.

4 + 2 + 1 = 7

- **grupo:** tiene permiso de lectura y ejecución.

4 + 1 = 5

- resto: tiene permisos de lectura y ejecución.

4 + 1 = 5

- tiene los permisos de setuid y setgid asignados.

4 + 2 = 6

Solución: 6755

7753
¿cómo se representa este permiso en letras?

7753

¿cómo se representa este permiso en letras?

- **usuario**: tiene permisos de lectura, escritura y ejecución.

rwx

- **grupo:** tiene permiso de lectura y ejecución.

r-x

- **resto:** tiene permisos de escritura y ejecución.

-wx

- tiene los permisos sticky bit, setuid y setgid asignados.

Solución: rwsr-s-wt

Tarea 07: permisos especiales

1. Expresa en números, los permisos que tiene un fichero con u=rws,g+rws,o=rw.
2. Expresa con letras el siguiente número 6555.
3. Expresa como activar con letras el siguiente comando chmod 4777 /tmp/file1
4. Expresa como activar con letras el siguiente comando chmod 4444 /tmp/file2
5. Expresa con números los siguientes permisos -rwsrwxrwt
6. Expresa con números los siguientes permisos -r-Sr--r—
7. Expresa como activar con letras el siguiente comando chmod 4536
8. Desde un usuario distinto al tuyo crea dos carpetas nuevas con varios ficheros dentro (pueden ser vacíos). Dale a las dos todos los permisos al propietario, grupo y otros (es decir, de lectura, escritura y ejecución). Después da a una permisos sticky bit y a la otra no. Desde tu usuario intenta borrar algún archivo de cada carpeta. Intenta borrar también la carpeta misma (recuerda utilizar la opción recursiva). Después accede desde root y borra todo lo que no hayas podido borrar.

2. Listas de control de acceso

Las listas de control de acceso (ACL) proporcionan mayor seguridad y amplían los permisos que hasta ahora hemos visto. En cualquier sistema se utilizan para modificar los permisos de acceso a sus elementos. **En un sistema de archivos como puede ser el de Linux, amplía los permisos habituales que se pueden dar con el comando chmod.**

2. Listas de control de acceso

Con las ACL a cada objeto del sistema se le puede asignar un control de acceso personalizado.

Los comandos para gestionar el control de acceso en Linux son los siguientes:

- **getfacl:** obtiene los permisos de una ACL asociada a un fichero o directorio.
- **setfacl:** establece los permisos de una ACL asociada a un fichero o directorio.

2. Listas de control de acceso

getfacl

Podemos comprobar los permisos de la siguiente manera:

```
miguel@mig-ubuntu-dk:~$ getfacl w1
# file: w1
# owner: miguel
# group: miguel
user::rw-
group::rw-
other::r--
```

2. Listas de control de acceso

setfacl

setfacl [opción] [u/g]:[usuario/grupo]:[permisos] [fichero/directorio]

El comando setfacl nos permite colocar los ACL y sus opciones más comunes son:

- **-m (modify):** Modifica los ACL de un archivo o directorio.
- **-x (remove):** Elimina las entradas ACLs.
- **-b (remove-all):** Elimina todas las entradas ACLs.
- **-L (logical):** Enrutamiento de los enlaces simbólicos.
- **-R (recursive):** Aplicación de los ACLs de forma recursiva.

Tarea 08: ACL de piratas

Existe un barco pirata llamado “La perla negra”, del que Jack Sparrow es el capitán, Gibbs, Ragetti y Pintel tienen el cargo de piratas y Will Turner tiene el cargo de piratilla. Independiente de ello, todos pertenecen al grupo “Barco”.

Desde tu usuario has creado dos directorios (Mapa y Caribe) y un fichero (Botín). Solo tú, como propietario, tienes permisos de lectura, escritura y ejecución. El resto de personas no pueden acceder a ellos.

Sin embargo, vas a hacer una excepción:

El Capitán tendrá acceso de lectura, escritura y ejecución a Caribe; al Mapa podrá acceder con permisos de lectura y ejecución y a Botín con permisos de lectura y escritura.

Los que ostentan el cargo de “Pirata” podrán acceder tanto a Mapa como a Caribe con permisos de lectura y ejecución.

El piratilla sólo podrá acceder al Caribe con permisos de lectura.

Con esta información, crea los usuarios y grupos correspondientes y crea las listas de control de acceso necesarias. Muéstralas utilizando el comando getfacl.

Realiza capturas para apoyar tu explicación.

Después, desde el Capitán, accede al fichero Botín y modifícalo.

Como nadie puede enterarse de que el Capitán ha cambiado el Botín, borra esa lista de control de acceso y crea una nueva en la que el Capitán sólo tenga permisos de lectura.

3. Redireccionamiento y tuberías

Redireccionamientos

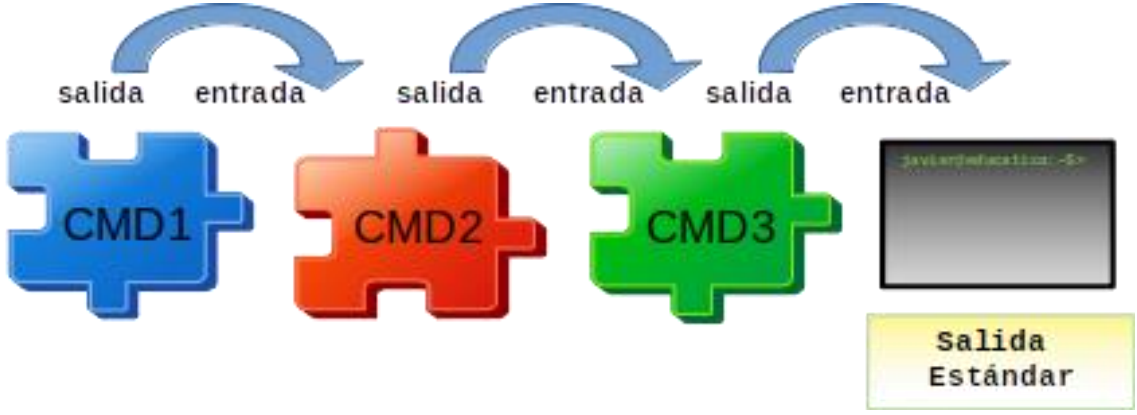
Las redirecciones consisten en trasladar información de un tipo a otro , por ejemplo, de error estándar a la salida estándar o de la salida estándar a la entrada estándar. A través de la terminal, logramos eso haciendo uso del símbolo >.

- `comando < fichero`: Toma la entrada de un fichero
- `comando > fichero`: Envía la salida del comando al fichero sobreescribiendo este
- `comando 2> fichero`: Envía la salida de error del comando al fichero
- `comando >> fichero`: Añade al final del archivo
- `comando 2>&1`: Envía la salida de error a la salida estándar
- `comando &> fichero`: Envía la salida estándar y error al fichero
- `comando 2> errorfile 1>stdout`: Enviar las salidas en archivos separados

3. Redireccionamiento y tuberías

Tuberías

Las tuberías son un tipo especial de redirección que permiten enviar la salida estándar de un comando como entrada estándar de otro. La forma de representarlo es con el símbolo | (pipe). Su principal utilidad es que nos ofrece la posibilidad de concatenar comandos, enriqueciendo la programación.



Tarea 09: Redireccionamientos y tuberías

1. Utilizando tuberías obtener un listado de los ficheros del directorio /etc ordenado por orden alfabético.
2. Utilizando tuberías obtener un listado de los ficheros del directorio /etc que contengan “ar”.
3. Utilizando tuberías obtener un listado de los ficheros del directorio /etc, ordenado por orden alfabético y que contengan “ar”.
4. Con redireccionamientos crea un archivo llamado profes.txt donde añadas el nombre de tres profesores de este año. Lista el contenido de tu carpeta de usuario y añádelo al final del archivo profes.
5. Con redireccionamientos crea un documento alumnos.txt donde guardes al menos 6 nombres de tus compañeros. Vuelca la información ordenada en un nuevo archivo creado por ti.
6. Con redireccionamientos lista el contenido de un directorio que no existe y redirecciona el error a un nuevo archivo creado por ti.
7. Vuelve a listar el contenido de un directorio inexistente, pero ahora añádelo a un archivo ya creado.