Práctica de laboratorio: Navegar por el sistema de archivos y la configuración de permisos de Linux

Objetivos

En esta práctica de laboratorio se familiarizarán con el sistema de archivos de Linux.

Parte 1: Explorar sistemas de archivo en Linux

Parte 2: Permisos de archivo

Parte 3: Enlaces simbolicos y otros tipos de archivos especiales

Recursos necesarios

- VM CyberOps Workstation

Instrucciones

Parte 1: Explorar sistemas de archivos en Linux

El sistema de archivos de Linux es una de sus características más populares. Si bien Linux admite muchos tipos diferentes de sistemas de archivos, aquí nos enfocaremos en la familia **ext**, uno de los sistemas de archivos más comunes en Linux.

Paso 1: Accedan a la línea de comandos.

Abran la VM CyberOps Workstation y, luego, una ventana del terminal.

Paso 2: Muestren los sistemas de archivos montados en este momento.

Los sistemas de archivos se deben *montar* primero para poder acceder a ellos y utilizarlo. En informática, *montar un sistema de archivos* significa tomar las medidas necesarias para que el sistema operativo pueda acceder a él. El montaje de un sistema de archivos es el proceso de vincular la partición física en el dispositivo de bloques (disco duro, unidad SSD, pen drive, etc.) a un directorio, a través del cual se puede acceder a todo el sistema de archivos. Como el directorio antes mencionado pasa a ser la raíz del sistema de archivos recién montado, también se lo conoce como *punto de montaje*.

a. Utilicen el comando **lsblk** para mostrar todos los dispositivos de bloques:

```
[analyst@secOps ~]$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda 8:0 0 10G 0 disk
└─sda1 8:1 0 10G 0 part /
sdb 8:16 0 1G 0 disk
└─sdb1 8:17 0 1023M 0 part
sr0 11:0 1 1024M 0 rom
```

En la salida anterior se muestra que la VM CyberOps Workstation tiene tres dispositivos de bloques instalados: sr0, sda y sdb. La salida en forma de árbol también muestra las particiones de sda y sdb. Convencionalmente, Linux utiliza /dev/sdX para representar discos duros, y el número del final representa el número de partición dentro de ese dispositivo. En las computadoras con varios discos duros probablemente se verán más dispositivos /dev/sdX. Si Linux se estuviera ejecutando en una computadora con cuatro disco por ejemplo, los mostraría como /dev/sda, /dev/sdb, /dev/sdc y /dev/sdd de manera predeterminada. La salida implica que sda y sdb son discos duros, y que cada uno contiene

una sola partición. En la salida también se muestra que sda es un disco de 10GB mientras que sdb tiene 1GB.

Nota: A menudo, Linux también muestra las unidades Flash USB como /dev/sdX, dependiendo de su tipo de firmware.

b. Utilicen el comando **mount** para mostrar información más detallada sobre los sistemas de archivos montados en este momento en la VM CyberOps Workstation.

```
[analyst@secOps ~]$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs (rw,nosuid,relatime,size=1030408k,nr_inodes=218258,mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
<output omitted>
```

Muchos de los sistemas de archivos anteriores están fuera del alcance de este curso y son irrelevantes para la práctica de laboratorio. Nos enfocaremos en el *sistema de archivos raíz*, el que está almacenado en **/dev/sda1**. El sistema de archivos raíz es el sitio en el que se almacena el sistema operativo Linux en sí; todos los programas, las herramientas y los archivos de configuración se almacenan en el sistema de archivos raíz de manera predeterminada.

c. Volver a ejecutar el comando **mount**, pero esta vez, agregar el pipe | para enviar el resultado de mount a **grep** para filtrar el resultado y solo mostrar el archivo de sistema root:

```
[analyst@secOps ~]$ mount | grep sda1
/dev/sda1 on / type ext4 (rw,relatime)
```

En la salida filtrada anterior, el montaje nos indica que el sistema de archivos raíz está ubicado en la primera partición del dispositivo de bloques sda (/dev/sda1). Sabemos que es el sistema de archivos raíz gracias al punto de montaje utilizado: "/" (el símbolo de la barra diagonal). La salida también nos informa el tipo de formato que se utilizó en la partición: ext4, en este caso. La información entre paréntesis está relacionada con las opciones de montaje de particiones.

d. Emitan los siguientes dos comandos en la **VM CyberOps Workstation**:

```
[analyst@secOps ~]$ cd /
[analyst@secOps /]$ ls -l
```

¿Cuál es el significado de la salida? ¿Dónde se almacenan físicamente los archivos de la lista?

¿Por qué no se muestra **/dev/sdb1** en la salida de arriba?

Paso 3: Montaje y desmontaje manual de sistemas de archivos

El comando **mount** también se puede utilizar para montar y desmontar sistemas de archivos. Tal como se ve en el Paso 1. La VM CyberOps Workstation tiene dos discos duros instalados. El kernel reconoció al primero como /dev/sda y al segundo como /dev/sdb. Para poder montar un dispositivo de bloques, este debe tener un punto de montaje.

a. Utilicen el comando **ls -l** para verificar que el directorio **second_drive** sea el directorio de inicio del analista.

```
[analyst@secOps /]$ cd ~
[analyst@secOps ~]$ ls –l
total 28
drwxr-xr-x 2 analyst analyst 4096 Sep 26 2014 Desktop
drwx------ 3 analyst analyst 4096 Jul 14 11:28 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 25 16:27 lab.support.files
drwxr-xr-x 2 analyst analyst 4096 Mar 3 15:56 second_drive
-rw-r--r-- 1 analyst analyst 142 Aug 16 15:11 some_text_file.txt
-rw-r--r-- 1 analyst analyst 254 Aug 16 13:38 space.txt
```

Nota: Si el directorio **second_drive** no existe, utilice el comando **mkdir second_drive** para crearlo.

```
[analyst@secOps ~]$ mkdir second_drive
```

Nota: Dependiendo del estado de su VM, su listado probablemente tenga otros archivos y directorios.

b. Vuelvan a utilizar **ls -l** para generar una lista con el contenido del directorio second_drive recién creado.

```
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

Observen que el directorio está vacío.

c. Utilicen el comando **mount** para montar **/dev/sdb1** en el directorio **second_drive** recién creado. La sintaxis del montaje es la siguiente: **mount [opciones] <dispositivo que se debe montar> <punto de montaje>**.

```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/
[sudo] contraseña para analyst:
```

No se genera ninguna salida; eso quiere decir que el proceso de montaje tuvo éxito.

d. Ahora que se ha montado **/dev/sdb1** en **/home/analyst/second_drive**, utilicen **ls -l** para volver a generar una lista con el contenido del directorio.

```
[analyst@secOps ~]$ ls -l second_drive/
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-r--r-- 1 root root 183 Mar 3 15:42 myFile.txt
```

¿Por qué ya no está vacío el directorio? ¿Dónde se almacenan físicamente los archivos de la lista?

e. Emitan el comando **mount** sin opciones nuevamente para mostrar información detallada sobre la partición **/dev/sdb1**. Como antes, utilicen el comando **grep** para mostrar solamente los sistemas de archivos de **/dev/sdX**:

```
[analyst@secOps ~]$ mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime)
/dev/sdb1 on /home/analyst/second_drive type ext4 (rw,relatime)
```

f. Desmontar sistemas de archivos es igual de simple. Recordar cambiar de directorio a una ubicación fuera del punto de montaje y utilicen el comando **umount** tal como se indica a continuación:

```
[analyst@secOps ~]$ sudo umount /dev/sdb1
[sudo] password for analyst:
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l second_drive/
total 0
```

Parte 2: Permisos de archivo

Los sistemas de archivos de Linux tienen características integradas para controlar la capacidad que tienen los usuarios de ver, cambiar, navegar y ejecutar el contenido del sistema de archivos. Esencialmente, cada archivo de los sistemas de archivos lleva consigo su propio conjunto de permisos, siempre con un conjunto de definiciones sobre lo que los usuarios y grupos pueden hacer con el archivo.

Paso 1: Visualizar y cambiar los permisos de archivo

a. Diríjanse a /home/analyst/lab.support.files/scripts/.

```
[analyst@secOps ~]$ cd lab.support.files/scripts/
```

b. Utilicen el comando **ls -l** para mostrar los permisos de archivo.

```
[analyst@secOps scripts]$ ls -l
total 60
-rwxr-xr-x 1 analyst analyst 190 Jun 13 09:45 configure_as_dhcp.sh
-rwxr-xr-x 1 analyst analyst 192 Jun 13 09:45 configure_as_static.sh
-rwxr-xr-x 1 analyst analyst 3459 Jul 18 10:09 cyberops_extended_topo_no_fw.py
-rwxr-xr-x 1 analyst analyst 4062 Jul 18 10:09 cyberops_extended_topo.py
-rwxr-xr-x 1 analyst analyst 3669 Jul 18 10:10 cyberops_topo.py
-rw-r--r-- 1 analyst analyst 2871 Apr 28 11:27 cyops.mn
-rwxr-xr-x 1 analyst analyst 458 May 1 13:50 fw_rules
-rwxr-xr-x 1 analyst analyst 70 Apr 28 11:27 mal_server_start.sh
drwxr-xr-x 2 analyst analyst 4096 Jun 13 09:55 net_configuration_files
-rwxr-xr-x 1 analyst analyst 65 Apr 28 11:27 reg_server_start.sh
-rwxr-xr-x 1 analyst analyst 189 Dec 15 2016 start_ELK.sh
-rwxr-xr-x 1 analyst analyst 85 Dec 22 2016 start_miniedit.sh
-rwxr-xr-x 1 analyst analyst 76 Jun 22 11:38 start_pox.sh
-rwxr-xr-x 1 analyst analyst 106 Jun 27 09:47 start_snort.sh
-rwxr-xr-x 1 analyst analyst 61 May 4 11:45 start_tftpd.sh
```

Consideren el archivo **cyops.mn** a modo de ejemplo. ¿Quién es el propietario del archivo? ¿Y del grupo?

Los permisos para **cyops.mn** son **–rw-r--r--**. ¿Qué significa esto?

c. El comando **touch** es muy simple y útil. Permite crear rápidamente un archivo de texto vacío. Utilicen el siguiente comando para crear un archivo vacío en el directorio **/mnt**:

```
[analyst@secOps scripts]$ touch /mnt/myNewFile.txt
touch: cannot touch '/mnt/myNewFile.txt': Permission denied
```

¿Por qué no se creó el archivo? Genere una lista con los permisos, la propiedad y el contenido del directorio **/mnt** y explique qué sucedió. Cuando se agrega la opción **-d**, muestra el permiso del directorio principal. Registren las respuestas en las siguientes líneas.

```
[analyst@secOps ~]$ ls -ld /mnt
drwxr-xr-x 2 root root 4096 Jan 5 2018 /mnt
```

¿Qué se puede hacer para que el comando **touch** que se muestra arriba tenga éxito?

d. El comando **chmod** se utiliza para cambiar los permisos de un archivo o directorio. Como antes, monten la partición **/dev/sdb1** en el directorio **/home/analyst/second_drive** que ya se creó en esta práctica de laboratorio:

```
[analyst@secOps ~]$ sudo mount /dev/sdb1 ~/second_drive/
```

e. Pasen al directorio **second_drive** y generen una lista con su contenido:

```
[analyst@secOps ~]$ cd ~/second_drive
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-r--r-- 1 root root 183 Mar 3 15:42 myFile.txt
```

¿Cuáles son los permisos del archivo **myFile.txt**?aquí.

f. Utilicen el comando **chmod** para cambiar los permisos de **myFile.txt**.

```
[analyst@secOps second_drive]$ sudo chmod 665 myFile.txt
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-rw-r-x 1 root root 183 Mar 3 15:42 myFile.txt
```

¿Cambiaron los permisos? ¿Cuáles son los permisos de **myFile.txt**?

El comando **chmod** toma permisos en formato octal. De esa manera, el desglose del 665 es el siguiente:

6 en octal es 110 en binario. Suponiendo que cada posición de los permisos de un archivo puede ser 1 o 0, 110 significa rw- (leer=1, escribir=1 y ejecutar=0).

Por lo tanto, el comando **chmod 665 myFile.txt** cambia los permisos a:

Owner: rw- (6 in octal or 110 in binary)

Group: rw- (6 in octal or 110 in binary)

Other: r-x (5 in octal or 101 in binary)

¿Qué comando cambiaría los permisos de myFile.txt a rwxrwxrwx, con lo que se otorgaría acceso total al archivo a cualquier usuario del sistema?

g. El comando **chown** se utiliza para cambiar la titularidad de un archivo o directorio. Emitir el comando de abajo para hacer a root propietario de **myFile.txt**:

```
[analyst@secOps second_drive]$ sudo chown analyst myFile.txt
[sudo] contraseña para analyst:
[analyst@secOps second_drive]$ ls -l
total 20
drwx------ 2 root root 16384 Mar 3 10:59 lost+found
-rw-rw-r-x 1 analyst root 183 Mar 3 15:42 myFile.txt
[analyst@secOps second_drive]$
```

Nota: Para cambiar ambos, propietario y el grupo a **analyst** al mismo tiempo, utilice el **sudo chown analyst:analyst myFile.txt** format.

h. Ahora que el usuario analyst es el propietario del archivo, anexen la palabra 'test' (prueba) al final de **myFile.txt**.

```
[analyst@secOps second_drive]$ echo test >> myFile.txt
[sudo] contraseña para analyst:
[analyst@secOps second_drive]$ cat myFile.txt
```

¿Fue exitosa la operación? Explique.

Paso 2: Directorios y permisos

En forma similar a lo que sucede con archivos comunes, los directorios también tienen permisos , tanto los archivos como los directorios tienen 9 bits para los permisos del propietario/user, group y otros. También hay tres bits más para permisos especiales: setuid, setgid y sticky que está más allá del alcance de este laboratorio.

a. Regresen al directorio /home/analyst/lab.support.files y emitan el comando **ls -l** para generar una lista de todos los archivos con detalles:

```
[analyst@secOps second_drive]$ cd ~/lab.support.files/
[analyst@secOps lab.support.files]$ ls -l
total 580
-rw-r--r-- 1 analyst analyst 649 Jun 28 18:34 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst 126 Jun 28 11:13 applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst 4096 Aug 7 15:29 attack_scripts
-rw-r--r-- 1 analyst analyst 102 Jul 20 09:37 confidential.txt
-rw-r--r-- 1 analyst analyst 2871 Dec 15 2016 cyops.mn
-rw-r--r-- 1 analyst analyst 75 May 24 11:07 elk_services
-rw-r--r-- 1 analyst analyst 373 Feb 16 16:04 h2_dropbear.banner
-rw-r--r-- 1 analyst analyst 147 Mar 21 15:30 index.html
-rw-r--r-- 1 analyst analyst 255 May 2 13:11 letter_to_grandma.txt
-rw-r--r-- 1 analyst analyst 24464 Feb 7 2017 logstash-tutorial.log
drwxr-xr-x 2 analyst analyst 4096 May 25 13:01 malware
-rwxr-xr-x 1 analyst analyst 172 Jul 25 16:27 mininet_services
drwxr-xr-x 2 analyst analyst 4096 Feb 14 2017 openssl_lab
```

```
drwxr-xr-x 2 analyst analyst 4096 Aug 7 15:25 pcaps
drwxr-xr-x 7 analyst analyst 4096 Sep 20 2016 pox
-rw-r--r-- 1 analyst analyst 473363 Feb 16 15:32 sample.img
-rw-r--r-- 1 analyst analyst 65 Feb 16 15:45 sample.img_SHA256.sig
drwxr-xr-x 3 analyst analyst 4096 Jul 18 10:10 scripts
-rw-r--r-- 1 analyst analyst 25553 Feb 13 2017 SQL_Lab.pcap
```

Comparen los permisos del directorio **malware** con el archivo **mininet_services**. ¿Cuál es la diferencia entre la parte inicial de la línea de malware y la línea mininet_services?

La letra 'd' al principio de la línea indica que el tipo de archivo es un directorio y no un archivo. Otra diferencia entre los permisos de un archivo y los de un directorio es el bit de ejecución. Si un archivo tiene activado su bit de ejecución, quiere decir que el sistema puede ejecutarlo. Los directorios son diferentes de los archivos con el bit de ejecución definido (un archivo con el bit de ejecución definido es un script o programa ejecutable). Un directorio con el bit de ejecución definido especifica si un usuario puede entrar a ese directorio.

Los comandos **chmod** y **chown** funcionan con los directorios del mismo modo que con los archivos.

Parte 3: Enlaces simbólicos y otros tipos de archivo especiales

Acaban de ver algunos de los diferentes tipos de archivo de Linux. El primer carácter de cada listado de archivos con el comando **ls -l** muestra el tipo de archivo. Los tres tipos diferentes de archivos en Linux, incluidos sus subtipos y caracteres, son los siguientes:

- **Archivos comunes (-),** entre los que se incluyen los siguientes:
  - Archivos que pueden leerse: archivos de texto
  - Archivos binarios: programas
  - Archivos de imagen
  - Archivos comprimidos
- **Archivos de directorio (d)**
  - Carpetas
- **Archivos especiales,** entre los que se incluyen los siguientes:
  - **Archivos de bloques (b)**: archivos que se utilizan para acceder a hardware físico como puntos de montaje para acceder a discos duros.
  - **Archivos de dispositivos de caracteres (c)**: archivos que proporcionan un flujo en serie de entrada y salida. Los terminales tty son ejemplos de este tipo de archivo.
  - **Archivos Pipe (p)** - Archivo usado para pasar información donde los primeros bytes entrantes son los primeros bytes en salir. También se conoce como FIFO (First In First Out, Primero en entrar primero en salir).
  - **Archivos de enlace simbólico (l)**: archivos que se utilizan para enlazar a otros archivos o directorios. Hay dos tipos: enlaces simbólicos y enlaces rígidos.
  - **Archivos de socket (s)**: se los utiliza para transmitir información de una aplicación a otra con el fin de comunicarse por una red.

Paso 1: Examinar tipos de archivo

a. Utilizar el comando **ls -l** para mostrar los archivos en la carpeta /home/analyst. Observar que los primeros caracteres de cada línea son un "–" indicando que es un archivo o una "d" indicando que es un directorio.

```
[analyst@secOps ~]$ ls -l
total 28
drwxr-xr-x 3 analyst analyst 4096 Aug 16 15:15 cyops_folder2
drwxr-xr-x 2 analyst analyst 4096 Sep 26 2014 Desktop
drwx------ 3 analyst analyst 4096 Jul 14 11:28 Downloads
drwxr-xr-x 9 analyst analyst 4096 Jul 25 16:27 lab.support.files
drwxr-xr-x 3 analyst analyst 4096 Mar 3 18:23 second_drive
-rw-r--r-- 1 analyst analyst 142 Aug 16 15:11 some_text_file.txt
-rw-r--r-- 1 analyst analyst 254 Aug 16 13:38 space.txt
```

b. Generen un listado del directorio **/dev**. Desplácense hasta la mitad de la salida y observen que los archivos de bloques comienzan con una “b”, los de dispositivos de caracteres con una “c” y los de enlaces simbólicos con una “l”:

```
[analyst@secOps ~]$ ls -l /dev/
<output omitted>
crw-rw-rw- 1 root tty 5, 2 May 29 18:32 ptmx
drwxr-xr-x 2 root root 0 May 23 06:40 pts
crw-rw-rw- 1 root root 1, 8 May 23 06:41 random
crw-rw-r-- 1 root root 10, 56 May 23 06:41 rfkill
lrwxrwxrwx 1 root root 4 May 23 06:41 rtc -> rtc0
crw-rw---- 1 root audio 253, 0 May 23 06:41 rtc0
brw-rw---- 1 root disk 8, 0 May 23 06:41 sda
brw-rw---- 1 root disk 8, 1 May 23 06:41 sda1
brw-rw---- 1 root disk 8, 16 May 23 06:41 sdb
brw-rw---- 1 root disk 8, 17 May 23 06:41 sdb1
drwxrwxrwt 2 root root 40 May 28 13:47 shm
crw------- 1 root root 10, 231 May 23 06:41 snapshot
drwxr-xr-x 2 root root 80 May 23 06:41 snd
brw-rw----+ 1 root optical 11, 0 May 23 06:41 sr0
lrwxrwxrwx 1 root root 15 May 23 06:40 stderr -> /proc/self/fd/2
lrwxrwxrwx 1 root root 15 May 23 06:40 stdin -> /proc/self/fd/0
lrwxrwxrwx 1 root root 15 May 23 06:40 stdout -> /proc/self/fd/1
crw-rw-rw- 1 root tty 5, 0 May 29 17:36 tty
crw--w---- 1 root tty 4, 0 May 23 06:41 tty0
<output omitted>
```

c. Los enlaces simbólicos de Linux son como los accesos directos de Windows. Hay dos tipos de enlaces en Linux: simbólicos y rígidos. La diferencia entre los enlaces simbólicos y los rígidos es que un archivo de enlace simbólico apunta al nombre de otro archivo y uno de enlace rígido apunta al contenido de otro archivo. Creen dos archivos con la función echo:

```
[analyst@secOps ~]$ echo "symbolic" > file1.txt
[analyst@secOps ~]$ cat file1.txt
symbolic
[analyst@secOps ~]$ echo "hard" > file2.txt
[analyst@secOps ~]$ cat file2.txt
hard
```

d. Utilicen **ln –s** para crear un enlace simbólico a file1.txt, y **ln** para crear un enlace rígido a file2.txt:

```
[analyst@secOps ~]$ ln -s file1.txt file1symbolic
[analyst@secOps ~]$ ln file2.txt file2hard
```

e. Utilicen el comando **ls –l** y examinen el listado del directorio:

```
[analyst@secOps ~]$ ls -l
total 40
drwxr-xr-x 3 analyst analyst 4096 Aug 16 15:15 cyops_folder2
drwxr-xr-x 2 analyst analyst 4096 Sep 26 2014 Desktop
drwx------ 3 analyst analyst 4096 Jul 14 11:28 Downloads
lrwxrwxrwx 1 analyst analyst 9 Aug 17 16:43 file1symbolic -> file1.txt
-rw-r--r-- 1 analyst analyst 9 Aug 17 16:41 file1.txt
-rw-r--r-- 2 analyst analyst 5 Aug 17 16:42 file2hard
-rw-r--r-- 2 analyst analyst 5 Aug 17 16:42 file2.txt
drwxr-xr-x 9 analyst analyst 4096 Jul 25 16:27 lab.support.files
drwxr-xr-x 3 analyst analyst 4096 Mar 3 18:23 second_drive
-rw-r--r-- 1 analyst analyst 142 Aug 16 15:11 some_text_file.txt
-rw-r--r-- 1 analyst analyst 254 Aug 16 13:38 space.txt
```

Observen que el archivo **file1symbolic** es un enlace simbólico y que tiene una **l** al comienzo de la línea y un puntero -> a **file1.txt**. **file2hard** parece ser un archivo común porque, de hecho, es un archivo común que apunta al mismo inodo de la unidad de disco duro que **file2.txt**. En otras palabras, **file2hard** apunta a los mismos atributos y ubicación de bloques del disco que **file2.txt**. El número 2 en la quinta columna de la lista para **file2hard** y **file2.txt** indica que hay 2 archivos vinculados al mismo inodo. Para un directorio que enumera la quinta columna indica el número de directorios dentro del directorio, incluidas las carpetas ocultas.

f. Cambien los nombres de los archivos originales: file1.txt y file2.txt, y observen el efecto en los archivos vinculados.

```
[analyst@secOps ~]$ mv file1.txt file1new.txt
[analyst@secOps ~]$ mv file2.txt file2new.txt

[analyst@secOps ~]$ cat file1symbolic
cat: file1symbolic: no such file or directory

[analyst@secOps ~]$ cat file2hard
hard
```

Observe que **file1symbolic** ahora es un enlace simbólico roto porque ha cambiado el nombre del archivo que apuntaba a to **file1.txt**, pero el archivo de enlace rígido **file2hard** todavía funciona correctamente porque apunta al inodo de **file2.txt** y no a su nombre, que ahora es **file2new.txt**.

¿Qué creen que sucedería con **file2hard** si abrieran un editor de texto y modificaran el texto de **file2new.txt**?

Reflexión

Los permisos y la titularidad de los archivos son dos de los aspectos más importantes de Linux. También son una causa común de problemas. Un archivo con los permisos o la titularidad definidos incorrectamente no estará disponible para los programas que necesitan acceder a él. En esta situación hipotética, el programa generalmente se interrumpirá y se encontrarán errores.