Laboratório - Navegando no Sistema de Arquivos Linux e Configurações de Permissão

# Objetivos

Neste laboratório, você usará familiarizar-se com sistemas de arquivos Linux.

Parte 1: Explorando sistemas de arquivos no Linux

Parte 2: Permissões de arquivo

Parte 3: Links simbólicos e outros tipos de arquivos especiais

# Recursos necessários

=   Security Workstation VM

# Instruções

## Parte 1: Explorando sistemas de arquivos no Linux

O sistema de arquivos Linux é um de seus recursos mais populares. Embora o Linux suporte muitos tipos diferentes de sistemas de arquivos, este laboratório se concentra na família **ext**, um dos sistemas de arquivos mais comuns encontrados no Linux.

### Etapa 1: Acesse a linha de comando

Inicie a VM Worstation CyberOps e abra uma janela de terminal.

### Etapa 2:Exibe os sistemas de arquivos atualmente montados.

Os sistemas de arquivos devem ser *montados* antes que possam ser acessados e usados. Na computação, *montar um sistema de arquivos* significa torná-lo acessível ao sistema operacional. A montagem de um sistema de arquivos é o processo de vincular a partição física no dispositivo de bloco (disco rígido, unidade SSD, pen drive, etc.) a um diretório, através do qual todo o sistema de arquivos pode ser acessado. Como o diretório acima mencionado se torna a raiz do sistema de arquivos recém-montado, ele também é conhecido como *ponto de montagem.*

a.     Use o comando **lsblk** para exibir todos os dispositivos de bloco:

[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

A saída acima mostra que a VM CyberOps Workstation tem três dispositivos de bloco instalados: sr0, sda e sdb. A saída em forma de árvore também mostra partições em sda e sdb. Convencionalmente, /dev/sdx é usado pelo Linux para representar discos rígidos, com o número à direita representando o número da partição dentro desse dispositivo. Computadores com vários discos rígidos provavelmente exibirão mais dispositivos /dev/SDX. Se o Linux estivesse sendo executado em um computador com quatro discos rígidos, por exemplo, ele os mostraria como /dev/sda, /dev/sdb, /dev/sdc e /dev/sdd, por padrão. A saída implica que sda e sdb são discos rígidos, cada um contendo uma única partição. A saída também mostra que sda é um disco de 10 GB enquanto sdb tem 1 GB.

**Nota**: O Linux geralmente exibe unidades flash USB como /dev/SDX também, dependendo do tipo de firmware.

b.     Use o comando **mount** para exibir informações mais detalhadas sobre os sistemas de arquivos atualmente montados na 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>

Muitos dos sistemas de arquivos acima estão fora do escopo deste curso e irrelevantes para o laboratório. Vamos nos concentrar no *sistema de arquivos raiz*, o sistema de arquivos armazenado em **/dev/sda1**. O sistema de arquivos raiz é onde o próprio sistema operacional Linux é armazenado; todos os programas, ferramentas, arquivos de configuração são armazenados no sistema de arquivos raiz por padrão.

c.     Execute o comando **mount** novamente, mas desta vez, use o pipe | para enviar a saída de mount para **grep** para filtrar a saída e exibir apenas o sistema de arquivos raiz:

[analyst@secOps ~]$ **mount | grep sda1**

/dev/sda1 on / type ext4 (rw,relatime)

Na saída filtrada acima, mount nos mostra que o sistema de arquivos raiz está localizado na primeira partição do dispositivo de bloco sda (/dev/sda1). Sabemos que este é o sistema de arquivos raiz devido ao ponto de montagem usado: “/” (o símbolo da barra). A saída também nos diz o tipo de formatação usado na partição, ext4 neste caso. As informações entre parênteses estão relacionadas com as opções de montagem de partição.

d.     Execute os dois comandos a seguir na **VM CyberOps Workstation**:

[analyst@secOps ~]$ **cd /**

[analyst@secOps ~]$ **ls –l**

#### Perguntas:

Qual é o significado da saída? Onde os arquivos listados são armazenados fisicamente?

Digite suas respostas aqui.

Por que **/dev/sdb1** não é mostrado na saída acima?

Digite suas respostas aqui.

### Montagem e desmontagem manual de sistemas de arquivos

O comando **mount** também pode ser usado para montar e desmontar sistemas de arquivos. Como visto na Etapa 1, a VM Security Workstation tem dois discos rígidos instalados. O primeiro foi reconhecido pelo kernel como /dev/sda enquanto o segundo foi reconhecido como /dev/sdb. Antes que um dispositivo de bloco possa ser montado, ele deve ter um ponto de montagem.

a.     Use o comando **ls -l** para verificar se o diretório **second\_drive** está no diretório home do 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

**Observação**: Se o diretório **second\_drive** não existir, use o comando **mkdir second\_drive** para criá-lo.

[analyst@secOps ~]$ **mkdir second\_drive**

**Observação**: Dependendo do estado de sua VM, sua listagem provavelmente terá arquivos e diretórios diferentes.

b.     Use **ls -l** novamente para listar o conteúdo do diretório second\_drive recém-criado.

[analyst@secOps ~]$ **ls -l second\_drive/**

total 0

Observe que o diretório está vazio.

c.     Use o comando **mount** para montar **/dev/sdb1** no diretório **second\_drive** recém-criado. A sintaxe de mount é: **mount [options] <device to be mounted> <mounting point>**.

[analyst@secOps ~]$ **sudo mount /dev/sdb1 ~/second\_drive/**

[sudo] password for analyst:

Nenhuma saída é fornecida, o que significa que o processo de montagem foi bem-sucedido.

d.     Agora que o **/dev/sdb1** foi montado em **/home/analyst/second\_drive**, use **ls -l** para listar o conteúdo do diretório novamente.

[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

#### Pergunta:

Por que o diretório não está mais vazio? Onde os arquivos listados são armazenados fisicamente?

Digite suas respostas aqui.

e.     Execute o comando **mount** sem opções novamente para exibir informações detalhadas sobre a partição **/dev/sdb1**. Como antes, use o comando **grep** para exibir somente os sistemas de arquivos **/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 arquivos é tão simples quanto possível. Certifique-se de alterar o diretório para algo fora do ponto de montagem e usar o comando **umount** , como mostrado abaixo:

[analyst@secOps ~]$ **sudo umount /dev/sdb1**

[sudo] password for analyst:

[analyst @secOps ~] $

[analyst@secOps ~]$ **ls -l second\_drive/**

total 0

## Parte 2: Permissões de arquivo

Os sistemas de arquivos Linux têm recursos internos para controlar a capacidade dos usuários de visualizar, alterar, navegar e executar o conteúdo do sistema de arquivos. Essencialmente, cada arquivo em sistemas de arquivos carrega seu próprio conjunto de permissões, sempre carregando um conjunto de definições sobre o que usuários e grupos podem fazer com o arquivo.

### Etapa 1: Visualize e altere as permissões de arquivo.

a.     Navegue até /home/analyst/lab.support.files/scripts/.

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

b.     Use o comando **ls -l** para exibir permissões de arquivo.

[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

#### Perguntas:

Considere o arquivo **cyops.mn** como um exemplo. Quem é o proprietário do arquivo? Que tal o grupo?

Digite suas respostas aqui.

As permissões para **cyops.mn** são **—rw-r—r—**. O que isso quer dizer?

Digite suas respostas aqui.

c.     O comando **touch** é muito simples e útil. Permite a criação rápida de um arquivo de texto vazio. Use o comando abaixo para criar um arquivo vazio no diretório **/mnt** :

[analyst@secOps scripts]$ **touch /mnt/myNewFile.txt**

touch: cannot touch '/mnt/myNewFile.txt': Permission denied

#### Perguntas:

Por que o arquivo não foi criado? Liste as permissões, a propriedade e o conteúdo do diretório **/mnt** e explique o que aconteceu. Com a adição da opção  **-d**, ele lista a permissão do diretório pai. Registre a resposta nas linhas abaixo.

[analyst@secOps ~]$ **ls -ld /mnt**

drwxr-xr-x 2 root root 4096 Jan  5  2018 /mnt

Digite suas respostas aqui.

O que pode ser feito para que o comando de **touch** mostrado acima seja bem-sucedido?

Digite suas respostas aqui.

d.     O comando **chmod** é usado para alterar as permissões de um arquivo ou diretório. Como antes, monte a partição **/dev/sdb1** no diretório **/home/analyst/second\_drive** criado anteriormente neste laboratório:

[analyst@secOps ~]$ **sudo mount /dev/sdb1 ~/second\_drive/**

e.     Mude para o diretório **second\_drive** e liste o conteúdo dele:

[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

#### Pergunta:

Quais são as permissões do arquivo **myFile.txt** ?

Digite suas respostas aqui.

f.      Use o comando **chmod** para alterar as permissões 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

#### Pergunta:

As permissões mudaram? Quais são as permissões do **myFile.txt**?

Digite suas respostas aqui.

O comando **chmod** tem permissões no formato octal. Deste modo, a repartição dos 665 é a seguinte:

6 em octal é 110 em binário. Assumindo que cada posição das permissões de um arquivo pode ser 1 ou 0, 110 significa rw- (read=1, write=1 e execute=0).

Portanto, o comando **chmod 665 myFile.txt** altera as permissões para:

**Proprietário**: rw- (6 em octal ou 110 em binário)

**Grupo**: rw- (6 em octal ou 110 em binário)

**Outros**: r-x (5 em octal ou 101 em binário)

#### Pergunta:

Qual comando alteraria as permissões do myFile.txt para rwxrwxrwx, concedendo a qualquer usuário no sistema acesso total ao arquivo?

Digite suas respostas aqui.

g.     O comando **chown** é usado para alterar a propriedade de um arquivo ou diretório. Execute o comando abaixo para tornar root o proprietário do **myFile.txt**:

[analyst@secOps second\_drive]$ **sudo chown analyst myFile.txt**

[sudo] password for 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]$

**Observação**: Para alterar o proprietário e o grupo para **analyst** ao mesmo tempo, use o formato **sudo chown analyst:analyst myFile.txt**.

h.     Agora que o analista é o proprietário do arquivo, tente anexar a palavra 'test' ao final de **myFile.txt**.

[analyst@secOps second\_drive]$ **echo test >> myFile.txt**

[sudo] password for analyst:

[analyst@secOps second\_drive]$ **cat myFile.txt**

#### Pergunta:

A operação foi bem sucedida? Explique.

Digite suas respostas aqui.

### Etapa 2: Diretório e permissões

Semelhante aos arquivos normais, os diretórios também carregam permissões. Ambos os arquivos e diretórios têm 9 bits para as permissões do proprietário/usuário, do grupo e outros. Há também mais três bits para permissões especiais: setuid, setgid e sticky que está além do escopo deste laboratório.

a.     Volte para o diretório /home/analyst/lab.support.files e execute o comando **ls -l** para listar todos os arquivos com detalhes:

[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

#### Pergunta:

Compare as permissões do diretório de **malware** com o arquivo **mininet\_services**. Qual é a diferença entre a parte inicial da linha de malware e a linha mininet\_services?

Digite suas respostas aqui.

A letra 'd' no início da linha indica que o tipo de arquivo é um diretório e não um arquivo. Outra diferença entre permissões de arquivo e diretório é o bit de execução. Se um arquivo tiver seu bit de execução ativado, isso significa que ele pode ser executado pelo sistema. Os diretórios são diferentes dos arquivos com o bit de execução definido (um arquivo com o bit de execução definido é um script ou programa executável).

Os comandos **chmod** e **chown** funcionam para diretórios da mesma forma que trabalham para arquivos.

## Parte 3: Links simbólicos e outros tipos de arquivos especiais

Você já viu alguns dos diferentes tipos de arquivos no Linux. O primeiro caractere em cada arquivo listado com o comando **ls —l** mostra o tipo de arquivo. Os três tipos diferentes de arquivos no Linux incluindo seus subtipos e caracteres são:

=   **Arquivos regulares (-)**, incluindo:

o      Arquivos legíveis — arquivos de texto

o      Arquivos binários - programas

o      Arquivos de imagem

o      Arquivos compactados

=   **Arquivos de diretório (d)**

o      Pastas

=   **Arquivos especiais**, incluindo:

o      **Arquivos de bloco (b)** — Arquivos usados para acessar hardware físico como pontos de montagem para acessar discos rígidos.

o      **Arquivos de dispositivo de caracteres (c)** — Arquivos que fornecem um fluxo serial de entrada e saída. terminais tty são exemplos desse tipo de arquivo.

o      **Arquivos de pipe (p)** — Um arquivo usado para passar informações em que os primeiros bytes são os primeiros bytes de saída. Isso também é conhecido como FIFO (primeiro a entrar primeiro a sair).

o      **Arquivos de link simbólico (l)** — Arquivos usados para vincular a outros arquivos ou diretórios. Existem dois tipos: links simbólicos e links rígidos.

o      **Arquivos de soquete** — Eles são usados para passar informações de aplicativo para aplicativo, a fim de se comunicar através de uma rede.

### Etapa 1: Examine os tipos de arquivo.

a.     Use o comando **ls -l** para exibir os arquivos na pasta /home/analyst. Observe que os primeiros caracteres de cada linha são um “— “indicando um arquivo ou um “d” indicando um diretório.

[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.     Produza uma listagem do diretório **/dev**. Role até o meio da saída e observe como os arquivos de bloco começam com um “b”, os arquivos de dispositivo de caractere começam com um “c” e os arquivos de link simbólico começam com um “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.     Links simbólicos no Linux são como atalhos no Windows. Existem dois tipos de links no Linux: links simbólicos e links rígidos. A diferença entre links simbólicos e links rígidos é que um arquivo de link simbólico aponta para o nome de arquivo de outro arquivo e um arquivo de link rígido aponta para o conteúdo de outro arquivo. Crie dois arquivos usando 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.     Use **ln —s** para criar um link simbólico para file1.txt e **ln** para criar um link rígido para file2.txt:

[analyst@secOps ~]$ **ln –s file1.txt file1symbolic**

[analyst@secOps ~]$ **ln file2.txt file2hard**

e.     Use o comando ls —l e examine a lista de diretórios:

[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

Observe como o arquivo **file1symbolic** é um link simbólico com um **l** no início da linha e um ponteiro -> para **file1.txt**. O **file2hard** parece ser um arquivo regular, porque na verdade é um arquivo regular que aponta para o mesmo inode na unidade de disco rígido como **file2.txt**. Em outras palavras, **file2hard** aponta para os mesmos atributos e localização do bloco de disco como **file2.txt**. O número 2 na quinta coluna da listagem para **file2hard** e **file2.txt** indica que existem 2 arquivos vinculados ao mesmo inode. Para um diretório listando a quinta coluna indica o número de diretórios dentro do diretório, incluindo pastas ocultas.

f.      Altere os nomes dos arquivos originais: file1.txt e file2.txt, e observe como isso afeta os arquivos 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 como **file1symbolic** é agora um link simbólico quebrado porque o nome do arquivo que apontou para **file1.txt** foi alterado, mas o arquivo de link rígido **file2hard** ainda funciona corretamente porque aponta para o inode de **file2.txt** e não seu nome, que agora é **file2new.txt**.

#### Pergunta:

O que você acha que aconteceria com **file2hard** se você abrir um editor de texto e alterar o texto em **file2new.txt**?

Digite suas respostas aqui.

# Reflexão

Permissões de arquivo e propriedade são dois dos aspectos mais importantes do Linux. Eles também são uma causa comum de problemas. Um arquivo que tenha permissões incorretas ou propriedade definida não estará disponível para os programas que precisam acessá-lo. Neste cenário, o programa normalmente irá quebrar e erros serão encontrados.

Fim do documento