

Activity - Navigating the Linux Files Permission Settings

File Permissions

Linux filesystems have built-in features to control the ability of the users to view, change, navigate, and execute the contents of the filesystem. Essentially, each file in filesystems carries its own set of permissions, always carrying a set of definitions about what users and groups can do with the file.

Step 1: Visualize and Change the File Permissions.

- a. Navigate to /home/analyst/lab.support.files/scripts/.

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

- b. Use the **ls -l** command to display file permissions.

```
[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
```

Consider the **cyops.mn** file as an example. Who is the owner of the file? How about the group?

The permissions for **cyops.mn** are **-rw-r--r--**. What does that mean?

- c. The **touch** command is very simple and useful. It allows for the quick creation of an empty text file. Use the command below to create an empty file in the **/mnt** directory:

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

Why was the file not created? List the permissions, ownership and content of the **/mnt** directory and explain what happened. With the addition of **-d** option, it lists the permission of the parent directory. Record the answer in the lines below.

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

What can be done for the **touch** command shown above to be successful?

- d. The **chmod** command is used to change the permissions of a file or directory. As before, mount the **/dev/sdb1** partition on the **/home/analyst/second_drive** directory created earlier in this lab:

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

- e. Change to the **second_drive** directory and list the contents of it:

```
[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
```

What are the permissions of the **myFile.txt** file?

- f. Use the **chmod** command to change the permissions of **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
```

Did the permissions change? What are the permissions of **myFile.txt**?

The **chmod** command takes permissions in the octal format. In that way, a breakdown of the 665 is as follows:

6 in octal is 110 in binary. Assuming each position of the permissions of a file can be 1 or 0, 110 means rw- (read=1, write=1 and execute=0).

Therefore, the **chmod 665 myFile.txt** command changes the permissions to:

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)

What command would change the permissions of **myFile.txt** to **rw-rw-rwx**, granting any user in the system full access to the file?

- g. The **chown** command is used to change ownership of a file or directory. Issue the command below to make root the owner of the **myFile.txt**:

Activity - Navigating the Linux Files Permission Settings

```
[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]$
```

Note: To change both the owner and the group to **analyst** at the same time, use the **sudo chown analyst:analyst myFile.txt** format.

- h. Now that analyst is the file owner, try appending the word 'test' to the end of **myFile.txt**.

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

Was the operation successful? Explain.

Step 2: Directory and Permissions

Similar to regular files, directories also carry permissions. Both files and directories have 9 bits for the permissions of the owner/user, the group, and others. There are also three more bits for special permissions: setuid, setgid, and sticky which is beyond the scope of this lab.

- a. Change back to the **/home/analyst/lab.support.files** directory and issue the **ls -l** command to list all the files with details:

```
[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
```

Activity - Navigating the Linux Files Permission Settings

Compare the permissions of the **malware** directory with the **mininet_services** file. What is the difference between beginning part of the malware line and the mininet_services line?

The letter 'd' at the beginning of the line indicates that the file type is a directory and not a file. Another difference between file and directory permissions is the execution bit. If a file has its execution bit turned on, it means it can be executed by the system. Directories are different than files with the execution bit set (a file with the execution bit set is an executable script or program). A directory with the execution bit set specifies whether a user can enter that directory.

The **chmod** and **chown** commands work for directories in the same way they work for files.