Linux Lab 3 Files and Permissions

Read

Read or listen to CyberAces Module 1 Linux, Session 6, Files and Permissions (https://tutorials.cyberaces.org/tutorials/view/1-1-6.html)
Do the exercises in the CyberAces module!

Read "The Linux Command Line", Chapter 9, pp 77 - 93 in the printed text or pp 90 - 109 in the pdf. Feel free to run any of the examples in a terminal on your Ubuntu VM.

Disk Management

Often, a hard disk is broken into partitions by the operating system. This has advantages:

- if a program uses too much disk space, it will fill its partition but will not affect the other partitions. If it could fill the entire disk the computer could crash.
- It allows different rights to be assigned to different partitions. A partition that contained system binaries that do not change could be marked read-only, so malware couldn't affect it.

Partitions have the disadvantage that they carve disk space into pieces. It may be difficult to resize partitions if you find you need more space in one partition.

You can see the partitions on the disk by using the command, **df**. It shows you how much space is available on the disk file system, and where the partitions are mounted.

The hard disk is called sda. "s" stands for SCSI, "d" for disc, and "a" means it is the first drive. If you had several hard drives, they would be sda, sdb, etc. (For modern Linux, all block devices get the "s" even when they aren't SCSI. Older disks on older Linux may be hda, hdb, etc.) Partitions on the disk are labeled sda1, sda2, etc. The df command shows the logical volumes (LVM) it has created in sda2 as /dev/mapper/… (The other partitions, tmpfs, reside in a portion of RAM.) Why would you want to protect sda1 so that something doesn't use up all its space? What is mounted there?

You should see where the partitions mount on the file system tree. One starts at /dev, and another starts at /run, and another at /. Which one is the very top of the file system tree?

Default partitioning varies dramatically by distribution and by the file system installed (LVM, for example.) Ubuntu desktop partitioning (default) just has the root drive and swap space. Other installations may have more partitions.

Run the command, `fdisk -l` , where lower case "L" means list (you have to run it as root or sudo.) DO NOT use any other options with fdisk, as they allow you to format your disk--not good when your system is already installed. This shows disk information at a more physical level. Here you see partitions that df did not show you, swap space in this case. Swap space is used when the system needs more memory than what is available in RAM. It "swaps" data between RAM and the swap space. Read man fdisk to see what the information means.

Programs that use disk space are assigned often assigned space in either /usr or /var. Often servers put /var in its own disk partition so that programs or logs installed there cannot overflow the disk and crash the machine.

Hand In

Hand in your commands and answers to questions 4, 8, and 11

Exercises

1) If there isn't an extra user on your computer, create one now. I'll call mine user1.

2) As root, create a directory in /var called test.

3) As root, create a group called test.

4) Use ls -l on /var and look at the file rights that are on /var/test. Pay special attention to:

   What is the user (owner) and what permissions does it have?

   What is the file's group, and what permissions does the group have?

   What permissions does the world (other or everyone) have?

5) Use commands to change the file rights on /var/test so that members of the group test have read/write/x access. (hint: you will need to create the test group, use chgrp (or chown with the : option) to make test the group that owns /var/test instead of root, then chmod if the permissions need to change.)

6) Make user1 a member of the group test. (hint: either gpasswd, or usermod)

7) Use the su command to switch user to user1. Create files in the /var/test directory. Check the file rights with ls -l.

8) Exit from su to get back to your regular user. Can your regular user delete files in /var/test? Can it read them? Why? (hint: ls -l, and the command whoami may help. Is your user the user/owner, group, or other?)

9) Change the rights for the /var/test directory, and also the files within it, so that only the user/owner and the group have read and write access. Everyone else/other should have no access at all. (hint: chmod ### is fastest. To change all files within test, you'll need to use the recursive flag for chmod.)

10) Test your changes by trying to read files in /var/test with your regular user.

setuid and setgid

Read

https://linuxconfig.org/how-to-use-special-permissions-the-setuid-setgid-and-sticky-bits
https://docs.oracle.com/cd/E19683-01/816-4883/6mb2joatb/index.html

In addition to the rwxrwxrwx bits, there are three more at the beginning that are not as easy to see. They are called setuid, setgid, and sticky. The first two are very important, as they can be used by an attacker to hide scripts or applications that have root privileges and run them as a normal user. The first article explains the basics of setuid and setgid.

If you use octal mode, the permission r-xr-xr-x would be 555. There is another set of 3 bits ahead of these, setuid, setgid, and sticky. So, the permissions for that file, without any of the 3 bits set would be 0555. If the setuid bit were set, it would be 4555.

In CyberPatriots, you may need to search for evil files hiding under setuid. The second article shows you one way to do this and slide 16 of the CyberAces module shows another. You will get a list of files with setuid set. The way to tell which ones should be there and which ones should not is to run the same command on a clean VM you've created and compare the results to the VM you are concerned about.

11) What files on your VM have the setuid bit set? (Save this so you can compare it with other VMs later.)