Session 3 - Linux Security and Files

Users, groups, and passwords

System accounts are stored in `/etc/passwd`; system groups are stored in `/etc/group`; account passwords are stored in `/etc/shadow`. Originally the passwords were stored in the second field in `/etc/passwd` but as hackers began brute force cracking the encrypted passwords in `/etc/passwd` it became evident that the passwords needed to be moved to a restricted file. The file `/etc/shadow` now contains the passwords and is restricted to root only.

Run the following commands to see the permissions on those files:

```
ls -l /etc/passwd
ls -l /etc/shadow
ls -l /etc/group
```

Types of users on Linux Systems

There are two types of users on Linux systems:

- system users (UID from 0 to 999). These are accounts for administrative tasks and also accounts which are associated with system services. Some system user accounts have higher privileges than other users. For example, the root account has full system privileges
- regular users (UID from 1000 to 60000). These accounts are for regular users of your system. These accounts have limited privileges.

Groups

User accounts are also assigned to *groups*. Groups are a way of organizing associated users into logical divisions. For example, a server at a large company may have an accounting group, a human resources group, an engineering group, and a management group. Your group membership on the server will depend on which department you work in.

User accounts on the system are listed in `/etc/passwd` and groups on the system are listed in `/etc/group`. Passwords for accounts are stored in `/etc/shadow`. Every user on the system is assigned to at least one group and some users are assigned to multiple groups.

Run the following commands:
`cat /etc/passwd`
Verify that system users have a UserID between 0 and 999
Verify that regular users have a UserID between 1000 and 60,000

`cat /etc/group`
Verify that system users are assigned to groups

You can view your own UID (userid) and GID (groupid) by typing in a terminal window
`id -a`

Permission Bits

After you have been introduced to Linux and the command line you will surely notice that files and directories have permission settings and ownership settings. This section will explain the Linux ownership and permissions model.

One of the most important features built into a multi-user system like Linux is the protection of data from unauthorized users. Linux has very robust data protection features built in and these are directly related to any discussion on the **CIA** aspects of cyber security.

File Permissions and Ownership

When listing file attributes with the `ls -l` command you will see 10 characters at the beginning of each output line with the format similar to `-rwxr-x-r-x`, `drw-rw-r--`, or `lrwxrwxrwx`. The first character will be

- A dash `-` to signify a file
- A `d` to signify a directory
- An `l` (lowercase L) to signify a symbolic link
- A `b` to signify a block device
- A `c` to signify a character device
- An `s` to signify a socket
- A `p` to signify a named pipe

The remaining nine characters are broken up into groups of three characters (`rwx`) called octets

- Each group of three characters signify the read, write, and execute permission for the file owner, group assigned to the file, and all others (respectively)
- If a dash appears inside one of these groups instead of the letter, it means the permission for that associated letter is turned off

Other important details about File Permissions

- You may see an `s` where you would normally see an `x` in the user or group bits like so, `-rwsr-xr-x`, or `-rwxr-sr-x`, or both `-rwsrwsr-x`. When the `s` appears it means any user can launch the command but the ownership of the running command is assigned to the application user/group instead of the user launching the command. This is referred to as SetUID (when the `s` appears in the user octet) and SetGID (when the `s` appears in the group octet)
- If a `t` appears at the end of a directory listing it means the sticky bit is set on that directory. For example `drwxrwxr-t`. The owner of a directory can set the sticky bit on a directory where other users can create files in order to prevent them from deleting files which do not belong to them. Some Linux distributions have the sticky bit set on the `/tmp` directory because all users can write files there but they are not allowed to delete files which they do not own.
- If you see a plus sign (+) or a period (.) at the end of the permission bits it means that the file has extended SELinux (Security Enhanced Linux) settings or ACL (access control list) settings.

Examples:

```
ls -l /usr/bin/date
-rwxr-xr-x. 1 root root 62200 Nov  5  2016 /usr/bin/date
```

This example shows that owner root has read-write-execute permissions on the date command, the root group has read-execute permissions on the file, and all other users have read-execute permissions on the file.

```
cd ~
ls -l /home/mary/
-rw-rw-r--.  1 mary mary   65 Jul  1 20:43 poem.txt
```

This example shows that owner mary has read-write permissions on the poem file, the mary group has read-write permissions, and all other users have read permission.

```
ls -l my_first_poem
lrwxrwxrwx. 1 mary mary 10 Jul  1 20:50 my_first_poem -> ./poem.txt
```

This example shows that mary has a symbolic link pointing to her poem file. Note: permissions on symbolic links are not taken into account by the file system. Modifying the permissions on a symbolic link will change the permissions on the file pointed to by the symbolic link. Although it appears that any user can modify the link (because of w in other grouping) only the owner of the link and root user may modify the symbolic link.

Because files and directories are different from each other, the read-write-execute permissions for them mean different things. The table below highlights each meaning

Permission	File	Directory
Read	View the contents of the file	View the files and subdirectories it contains
Write	Change the file contents, rename the file, or delete the file	Add or remove files and subdirectories to the directory
Execute	Run the file as a program	Change to the directory with the `cd` command, execute a program in the directory, view file details of files in the directory

Changing Permissions with `chmod` command

The command `chmod` (which means “change mode”) is used to change the way a file or directory can be accessed. There are two ways to use this command, using number or letters, and we will review both methods.
If you own a file or are the root user you can change the permissions on a file or directory any way you please. Recall that the nine permission characters are broken into groups of three. Each character has a numeric value as follows, `r = 4, w = 2, x = 1, -(dash) = 0`. Each octet of 3 characters (`rwx`) can have numeric values from 0 to 7 (a total of 8 possible values). This is why the group of 3 characters is called an *octet*. So the octet `rwx` would have the numeric value of 7 (4 + 2 + 1). This octet `rw-` would have the numeric value of 6 (4 + 2 + 0) and so on.
Here are some examples of using the `chmod` command with numeric values:

Original file

```
ls -l my_script
-rw-rw-r--.  1 mary mary   65 Jul  1 20:43 my_script
```

`chmod 777 my_script`, results in `-rwxrwxrwx`
`chmod 755 my_script`, results in `-rwxr-xr-x`
`chmod 644 my_script`, results in `-rw-r--r--`
`chmod 000 my_script`, results in `----------`
`chmod -R 755 my_apps_dir`, results in all files and directories below, and including the `my_apps_dir` being recursively changed to `rwxr-xr-x`.

The other method of using the chmod command involves using letters and plus (`+`) and minus (`-`) signs. You can change the read, write, and execute bits for the (`u`) user or owner of the file, (`g`) the group assigned to the file, (`o`) other users, and (`a`) all users.
Here are some examples of using the `chmod` command with letters:

Original file

```
ls -l my_script
-rw-rw-r--.  1 mary mary   65 Jul  1 20:43 my_script
```

Command	Results in
`chmod a-w my_script`	`-r--r--r--`
`chmod a+x my_script`	`-rwxrwxr-x`
`chmod u+x my_script`	`-rwxrw-r--`
`chmod o+w my_script`	`-rw-rw-rw-`
`chmod ug+x my_script`	`-rwxrwxr--`

Setting the SetUID, SetGID, and sticky bit

A numeric mode consists of one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Omitted digits are assumed to be leading zeros. The first digit selects the SetUID (4) and SetGID (2) and sticky bit (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

For example to set the SetUID and SetGID permissions on a file
`chmod 6775 my_script` would yield `-rwsrwsr-x`

To set the sticky bit on a directory
`chmod 1777 /home/mary/my_temp_dir` would yield `drwxrwxrwt`

When a regular user creates a file it is given a default permission of `-rw-rw-r--` and a newly created directory is given a default permission of `rwxrwxr-x`. Linux systems have a setting called umask which you can see by typing `umask` in a terminal window. It is typically set to `0002`. A non-executable regular file with wide-open permissions would be `666` or `rw-rw-rw-` and a directory with wide-open permissions would be `777` or `rwxrwxrwx`. If you ignore the first zero of the umask value, you would have `002`. Subtracting the umask value from a wide-open setting for a file would be

```
 666
-002
------
 664  ⇒ rw-rw-r--
```

Subtracting the umask value from a wide-open setting for a directory would be

```
 777
-002
------
 775 ⇒ rwxrwxr-x
```

This example shows the results of creating new files or directories with a umask value of `022`

```
 666
-022
------
 644  ⇒ rw-r--r--

 777
-022
------
 755 ⇒ rwxr-xr-x
```

Changing file ownership with the `chown` command

The root user can change the ownership of any files on the system. Regular users cannot change the file ownership of the files they have created. The following examples (which are run as root user) show how the root user can use this command.

Original file

```
ls -l my_script
-rw-rw-r--.  1 mary mary   65 Jul  1 20:43 my_script
mv /home/mary/my_script ~/roots_script
chown root:root ~/roots_script
```

Original directory

```
ls -ld /media/myusb
drwxr-xr-x. 2 root root 6 Jul  2 15:57 /media/myusb
chown -R mary:mary /media/myusb
```

Session 3 Homework

Try variations on all of the commands located under Input/Output Redirectors and Process Handling in the cheat sheet. Remember to put a command in front of the redirection symbol. For example: `ls -la > files_in_my_dir`

Save the outputs to a file called homework3.txt to turn in to show what you tried. Attach this file in an email to your instructor with "Homework 3" in the subject line.

Also make note of which programs are not installed.

REMEMBER: if you are having difficulty getting a command to work read the man page.