File System Security: Permissions

OPS102 Week 4 Class 1

Tiayyba Riaz/John Sellens

September 23, 2024

Seneca Polytechnic

Outline

Introduction to Permissions

File System Security: Setting Permissions

- In multi-user operating systems it is very important to be able to control who has access to files and folders.
- This is achieved by setting permissions of files and folders
- All modern operating systems provide ways to set such permissions
- Linux provides commands and command options for setting permissions
  - Standard Linux/UNIX permission mechanisms are fairly simple.
  - But usually sufficient.
  - There are also more advanced ACLs (Access Control Lists).
  - Which we will not be covering.

File Permissions in Linux

- Every thing in the file system has its own set of permissions (or ”mode”).
- Permissions determine who may access a file, and in what way(s).
- There are three permission indicator, in three sets (user, group, others):

```
[tiayyba.riaz@mtrx-node06pd ~]$ ls -l f1.sh
-rwxr--r-- 1 tiayyba.riaz users 10 Sep 24 10:52 f1.sh
```

r	Read	Read access to the file’s contents.
w	Write	Permission to modify the file’s contents. Usually want read with write.
x (or s)	Execute	Allows the execution of file as a command. Scripts also need read permission to execute. Might be s: Covered later ...
–		Indicates that the related permission is not granted.

Directory Permissions in Linux

- Directory permissions are similar to file permissions.
- But have slightly different meanings.

r	Read	Grant permission to read the contents of directory, list contents using the `"ls"` command.
w	Write	Grant permission to change/edit directory content, create/delete/rename subdirectories and files in that directory. You can delete a file regardless of the file's permissions.
x	Access	"Pass-through" permission – allows access to and through the directory, but does not in itself allow reading or writing. If you know where you're going, this permission allows access through intervening directories.

Viewing Permissions

Viewing Permissions

The `"-l"` (long) option to `"ls"` shows permissions (and other details) about files and directories.

- `"ls -d"` on a directory, shows the directory, not the things it contains.
- Directory permissions allow or prevent removing or renaming files (or other objects) in the directory.

e.g. For the path `"dirA/fileB"`, if you have pass-through or better access to `"dirA"`

- Read permission on `"fileB"` allows reading the file.
- Write permission on `"dirA"` allows removing `"fileB"`.
- If you don't have write permission on `"fileB"` the `"rm"` command will (usually) prompt for confirmation.

Sample `"ls"` Commands for Directories

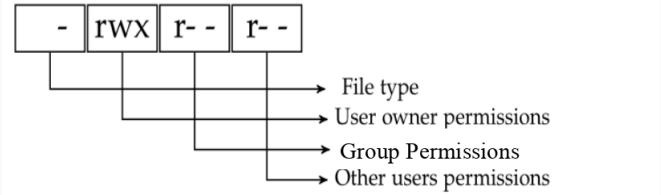
Sample commands to view directory permissions:

- `"ls -ld"` – shows permissions and details of your current diectory.
- `"ls -ld dir1"` – shows permissions and details of `"dir1"`, if you have read access (or better) to the parent of `"dir1"`. (I think.)
- `"ls -l dir1"` – shows permissions and details of the contents of `"dir1"`, if you have read permissions on `"dir1"`.

Decoding Permissions in "ls -l" Output

- Character 1: file type (regular, directory, link, device, etc)
- Character 2-4: Permissions for owner of the object
- Character 5-7: Permissions for members of the group that the object belongs to
- Character 8-10: Permissions for all other users

```
[tiayyba.riaz@mtrx-node06pd ~]$ ls -l f1.sh
-rwxr--r-- 1 tiayyba.riaz users 10 Sep 24 10:52 f1.sh
```



Setting Permissions

Changing Permissions with "chmod"

- The "chmod" ("change mode") command is used to change/grant/revoke permissions to different users/groups
  - "chmod permissions file"
- There are two methods to set permissions with "chmod"
  - Symbolic method: Using alphabetic characters
  - Octal Method: Using Octal numbers

Symbolic Method for "chmod"

- Permissions are set for:
  - user (u), group (g), others (o), or all (a)
- Permissions are set by:
  - adding (+), removing (-) and/or setting(=)
- Permissions are set to:
  - read (r), write (w) and/or execute (x)

Action	Sample Command
Add Permission	`"chmod g+rw file1"` – add group rw permissions
Remove Permission	`"chmod a-w test.txt"` – remove all write permissions
Set Permission	`"chmod go=rx file1"` – set `r-x` for group and other
Combination	`"chmod u+rx,g-x,o= file1"` – for you to decode

Octal Method for `"chmod"`

- Permissions can be set explicitly by `"chmod"` with an octal number.
- Octal numbers represent the permission bits - see `"man 2 stat"`
- A set bit (value 1) grants the permission, unset (value 0) does not grant.
- read = 4, write=2, execute=1
- Combine the octal representation of user, group, and other permissions to form a 3-digit octal number e.g. `rwxr-x---` is octal 750.

OCTAL	READ	WRITE	EXECUTE	PERMISSIONS
0	0	0	0	none
1	0	0	1	execute
2	0	1	0	write
3	0	1	1	write and execute
4	1	0	0	read
5	1	0	1	read and execute
6	1	1	0	read and write
7	1	1	1	read, write, and execute

Octal Method for "chmod" Examples

- You can use the "chmod" command with 3 octal digits to represent the permissions for user, group, and others.
- The command "chmod 754 file1.txt" means the following

	User	Group	Other
Permissions	rwx	r-x	r--
Binary	111	101	100
Octal	7	5	4

- The command "chmod 755 file1.txt" sets the permissions to: rwxr-xr-x
- The command "chmod 531 file1.txt" sets the permissions to : r-x-wx--x

Special Permission Bits

setuid, setgid, and sticky bits

There are 3 additional bits in a file's mode, which come before the permission bits, 12 bits in all.

- setuid: When an executable has the setuid bit enabled, it runs with the permissions of the file owner. This allows it to perform actions that would typically require the owner's privileges.
- setgid: Similarly, when an executable has the setgid bit enabled, it runs with the permissions of the group associated with the file. This enables it to access resources and perform tasks limited to members of that group.
- sticky bit: The sticky bit is typically used on directories. When applied, it ensures that only the owner of a file or the root user can delete or modify it. This helps prevent accidental or unauthorized deletion of files by other users.

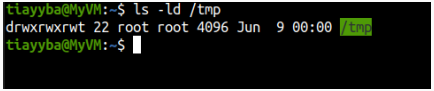
How Does setuid/setgid Work?

- In some cases, you may need to run a program with root priviliges.
- For example, the `"passwd"` command allows users to change their password.
  - This requires changing the `"/etc/shadow"` file, however, only the root user has write access to `"/etc/shadow"`.
- Normal users can execute the passwd command to change their own password without `"sudo(1)"` access for root user permissions.
- This is because the permissions for the `"passwd"` command contains an `s` where you'd expect `x` for the file owner's permissions.
  - This `s` tells us that the setuid bit is set and the command will be executed with the command owner's permissions without requiring `"sudo"` access.

```
tlayyba@MyVM:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 138208 Feb  7  2022 /bin/ls
tlayyba@MyVM:~$ ls -l /bin/passwd
-rwsr-xr-x 1 root root 59976 Nov 24  2022 /bin/passwd
tlayyba@MyVM:~$
```

The Sticky Bit

- The sticky bit in Linux is a special permission that can be set on directories.
- When the sticky bit is enabled on a directory, only the owner of a file within that directory or the root user can delete or rename the file.
- Other users, even if they have write permissions on the directory, cannot remove or modify files owned by other users.
- The main purpose of the sticky bit is to ensure the privacy and security of files in shared directories.
- Since `"/tmp"` is mode `1777`, any user can create files/directories there, but can't delete anything owned by someone else.



Default Permissions and Umask

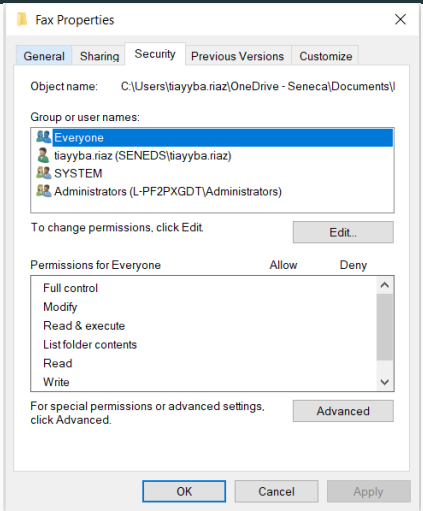
umask

- The umask is an attribute of a process, and is inherited by child processes.
  - Normally set at shell start-up, and applies to all commands run.
- A process's umask value "masks off" (disables) permission bits for any files or directories created by the process. default permissions for newly created files and directories
  - The base permissions for directories are 777 and for files 666
  - i.e. The default is pass-through for directories, but non-executable for files.
- For example: a umask value of 026 means:
  - A newly created directory gets mode 777 – 026 or 751 or `rwxr-x--x`
  - A newly created file gets mode 666 – 026 or 640 or `rw-r-----`

File and Folder Permissions in Windows

File and Folder Permissions in Windows

- In windows OS, you can access the permissions from the properties of any file or folder.
- The security tab in the properties shows the current set of permissions.
- A user with appropriate privileges can modify the permissions.



Summary

- More than one user? Need a permission mechanism.
- Simple permissions are often sufficient.
- Permissions, viewing, changing (symbolic vs octal)
- Special bits – setuid, setuid, stick
- Default permissions and umask
- Windows quick mention