Permission Management

Under Linux, permissions are assigned to users and groups. Each user can be a member of different groups, and membership in these groups gives the user specific, additional permissions. Each file and directory belongs to a specific user and a specific group. So the permissions for users and groups that defined a file are also defined for the respective owners. When we create new files or directories, they belong to the group we belong to and us.

When a user wants to access the contents of a Linux directory, they must first traverse the directory, which means navigating to that directory, requiring the user to have `execute` permissions on the directory. Without this permission, the user cannot access the directory's contents and will instead be presented with a “`Permission Denied`" error message.

```
cry0l1t3@htb[/htb]$ ls -l

drw-rw-r-- 3 cry0l1t3 cry0l1t3   4096 Jan 12 12:30 scripts


cry0l1t3@htb[/htb]$ ls -al mydirectory/

ls: cannot access 'mydirectory/script.sh': Permission denied
ls: cannot access 'mydirectory/..': Permission denied
ls: cannot access 'mydirectory/subdirectory': Permission denied
ls: cannot access 'mydirectory/.': Permission denied
total 0
d????????? ? ? ? ?            ? .
d????????? ? ? ? ?            ? ..
-????????? ? ? ? ?            ? script.sh
d????????? ? ? ? ?            ? subdirectory
```

It is important to note that `execute` permissions are necessary to traverse a directory, no matter the user's level of access. Also, `execute` permissions on a directory do not allow a user to execute or modify any files or contents within the directory, only to traverse and access the content of the directory.

To execute files within the directory, a user needs to `execute` permissions on the corresponding file. To modify the contents of a directory (create, delete, or rename files and subdirectories), the user needs `write` permissions on the directory.

The whole permission system on Linux systems is based on the octal number system, and basically, there are three different types of permissions a file or directory can be assigned:

- (`r`) - Read
- (`w`) - Write
- (`x`) - Execute

The permissions can be set for the `owner`, `group`, and `others` like presented in the next example with their corresponding permissions.

```
cry0l1t3@htb[/htb]$ ls -l /etc/passwd

- rwx rw- r--   1 root root 1641 May  4 23:42 /etc/passwd
- --- --- ---   |  |    |    |   |__________|
|  |   |   |    |  |    |    |        |_ Date
|  |   |   |    |  |    |    |__________ File Size
|  |   |   |    |  |    |_______________ Group
|  |   |   |    |  |____________________ User
|  |   |   |    |_______________________ Number of hard links
|  |   |   |_ Permission of others (read)
|  |   |_____ Permissions of the group (read, write)
|  |_________ Permissions of the owner (read, write, execute)
|____________ File type (- = File, d = Directory, l = Link, ... )
```

Change Permissions

We can modify permissions using the chmod command, permission group references (u - owner, g - Group, o - others, a - All users), and either a [+] or a [-] to add remove the designated permissions. In the following example, a user creates a new shell script owned by that user, not executable, and set with read/write permissions for all users.

```
cry0l1t3@htb[/htb]$ ls -l shell

-rwxr-x--x   1 cry0l1t3 htbteam 0 May  4 22:12 shell
```

We can then apply read permissions for all users and see the result.

```
cry0l1t3@htb[/htb]$ chmod a+r shell && ls -l shell

-rwxr-xr-x   1 cry0l1t3 htbteam 0 May  4 22:12 shell
```

We can also set the permissions for all other users to read only using the octal value assignment.

```
cry0l1t3@htb[/htb]$ chmod 754 shell && ls -l shell

-rwxr-xr--   1 cry0l1t3 htbteam 0 May  4 22:12 shell
```

Let us look at all the representations associated with it to understand better how the permission assignment is calculated.

```
Binary Notation:                4 2 1  |  4 2 1  |  4 2 1
----------------------------------------------------------
Binary Representation:          1 1 1  |  1 0 1  |  1 0 0
----------------------------------------------------------
Octal Value:                      7    |    5    |    4
----------------------------------------------------------
Permission Representation:      r w x  |  r - x  |  r - -
```

If we sum the set bits from the `Binary Representation` assigned to the values from `Binary Notation` together, we get the `Octal Value`. The `Permission Representation` represents the bits set in the `Binary Representation` by using the three characters, which only recognizes the set permissions easier.

Change Owner

To change the owner and/or the group assignments of a file or directory, we can use the `chown` command. The syntax is like following:

Syntax - chown

Syntax - chown

```
cry0l1t3@htb[/htb]$ chown <user>:<group> <file/directory>
```

In this example, "shell" can be replaced with any arbitrary file or folder.

Syntax - chown

```
cry0l1t3@htb[/htb]$ chown root:root shell && ls -l shell

-rwxr-xr--   1 root root 0 May  4 22:12 shell
```

SUID & SGID

Besides assigning direct user and group permissions, we can also configure special permissions for files by setting the `Set User ID` (`SUID`) and `Set Group ID` (`SGID`) bits. These `SUID`/`SGID` bits allow, for example, users to run programs with the rights of another user. Administrators often use this to give their users special rights for certain applications or files. The letter "`s`" is used instead of an "`x`". When executing such a program, the SUID/SGID of the file owner is used.

It is often the case that administrators are not familiar with the applications but still assign the SUID/SGID bits, which leads to a high-security risk. Such programs may contain functions that allow the execution of a shell from the pager, such as the application "`journalctl`."

If the administrator sets the SUID bit to "`journalctl`," any user with access to this application could execute a shell as `root`. More information about this and other such applications can be found at GTFObins.

Sticky Bit

Sticky bits are a type of file permission in Linux that can be set on directories. This type of permission provides an extra layer of security when controlling the deletion and renaming of files within a directory. It is typically used on directories that are shared by multiple users to prevent one user from accidentally deleting or renaming files that are important to others.

For example, in a shared home directory, where multiple users have access to the same directory, a system administrator can set the sticky bit on the directory to ensure that only the owner of the file, the owner of the directory, or the root user can delete or rename files within the directory. This means that other users cannot delete or rename files within the directory as they do not have the required permissions. This provides an added layer of security to protect important files, as only those with the necessary access can delete or rename files. Setting the sticky bit on a directory ensures that only the owner, the directory owner, or the root user can change the files within the directory.

When a sticky bit is set on a directory, it is represented by the letter “`t`" in the execute permission of the directory's permissions. For example, if a directory has permissions “`rwxrwxrwt`", it means that the sticky bit is set, giving the extra level of security so that no one other than the owner or root user can delete or rename the files or folders in the directory.

Syntax - chown

```
cry0l1t3@htb[/htb]$ ls -l

drw-rw-r-t 3 cry0l1t3 cry0l1t3   4096 Jan 12 12:30 scripts
drw-rw-r-T 3 cry0l1t3 cry0l1t3   4096 Jan 12 12:32 reports
```

In this example, we see that both directories have the sticky bit set. However, the `reports` folder has an uppercase `T`, and the `scripts` folder has a lowercase `t`.

If the sticky bit is capitalized (`T`), then this means that all other users do not have `execute` (`x`) permissions and, therefore, cannot see the contents of the folder nor run any programs from it. The lowercase sticky bit (`t`) is the sticky bit where the `execute` (`x`) permissions have been set.

Start Instance

0 / 1 spawns left

Waiting to start...

Previous Next Mark Complete & Next

Cheat Sheet

Table of Contents