8.11.2 Assigning Special File Permissions

Click one of the buttons to take you to that part of the video.

Assign Special File Permissions 0:00-1:28

In this demonstration, we're going to discuss assigning special file permissions. Let's begin by looking at assigning the SUID and SGID special permissions to a file in the file system.

I am currently logged in to the system as my rtracy user, and I'm in my /home directory. If we run the 'ls -l' command, we can see that there is a file here named runme. You can see that currently the file's owner, which is me, is assigned read/write permissions to the file. The owning group, which is users, is assigned read/write permissions to the file. And other authenticated users are assigned just read access to the file.

This file is actually a script that can be executed. If I run 'cat runme', we can see that it is a very simple script that simply displays a single line of text on the screen that says, "Hello world." However, currently, if I try to run this file, it gives me an error saying "Permission denied."

The script is there. The code is there that can run, but the execute permission has not been assigned to that file--either to me as a user or through group as a member of the users group, which I currently am.

What we want to do here is make this file readable, writable, and executable by owner and group. We're not going to worry about others; we'll just leave them with their default read permission for now.

Set the SUID and SGID Special Permissions 1:29-4:10

In addition, we also want to add the SUID and the SGID special permissions. This will cause any user who runs this executable file to temporarily become that file's owner. It will cause any user who runs this executable file to temporarily become a member of this file's owning group, which is users.

This is done by adding an extra digit to the beginning, not the end, but the beginning of this file's mode. It works the same way if we were doing it with a directory.

These two special permissions are added by adding an extra digit to the beginning of the file's mode. SUID has a value of 4. SGID has a value of 2. Sticky bit has a value of 1. We'll work with that later.

For now, we're working with SUID and SGID, so we add those two values together to get 6. First, we need to switch to my root user account, and we need to switch to the '/home/rtracy' directory. Do an 'ls -l' once again to make sure that we're in the right place. Yep.

We run 'chmod', and then we specify an extra '6' at the beginning, indicating SUID (4) and SGID (2) and then the rest of the mode that we want to assign to the file. Let's assign '7' for user, the owner, which will grant read (4), write (2), and execute (1), for a total of '7'.

We'll assign the same permissions to group, and we'll leave others with just read access to the file. Then we need to specify the name of the file, of course, 'runme'. Hit Enter.

Let's run 'ls -l' again and see what changed. Notice over here in the mode, we have rws permissions assigned to the file's owner. Notice that the s replaced the x that would normally be there if we made this file executable. This indicates that the SUID special permission has been assigned. The file is still executable. It will run, but it will run using the SUID special permissions.

Likewise, an S has replaced the X in the group portion of the mode--again indicating that the SGID special permission has been assigned. At this point, I should be able to just do the 'runme' command here, and the command actually executed.

When it did, the SUID and SGID special permissions were applied. It didn't actually make any difference, because I am currently the file owner, and I am already a member of users.

But if I had logged in as a different user account and ran them, that user would have temporarily become this file's owner while it was running, and that user would have temporarily become a member of the users group because of the SUID and SGID permissions that were enabled on this file.

Set the Sticky Bit Permissions 4:11-10:34

We're going to shift gears a little bit and look at setting the sticky bit permissions. Specifically, we're going to set the sticky bit permission on a directory.

This has a very important use, because when you enable the sticky bit permission on a directory, then users will be allowed to delete files within that directory only if they are that file's owner. If they are not the file's owner, they won't be allowed to delete that file. As you can imagine, that's probably a pretty good idea.

Currently, we have a shared folder already defined on this system, /mnt/shared. If we do an 'ls' command, we see that within the /shared directory, we have a directory named /RandD. Let's actually look at its mode.

Right now, the directory's owner has read, write, and execute permissions to the RandD directory, allowing me to list file contents; add or delete files to the folder, which allows me to list all the files in the folder; it allows me to add files to the folder; delete files from the folder; and also enter the folder. That's the execute permission.

Group has the same permissions. Anyone who is a member of the RandD group can list the files in the folder, add or delete files in the folder, and enter the folder because of the mode that's been assigned, while other authenticated users have no access to the folder at all.

If we run the 'tail' command to view the end of the 'etc/group' file, we see that my user account--rtracy--and the ksanders user account are both members of the RandD group, which is the owning group for the RandD folder.

Because of the mode that's been assigned to the RandD folder, either user, rtracy or ksanders, is allowed to delete any file they want within the RandD directory. This has some negative ramifications.

I'm going to go ahead and switch into the RandD folder. Actually, I want to 'exit'. I want to be out of my root user account. What I want to do is switch to '/mnt/shared/RandD' as my rtracy user because I want to create a user in this directory as rtracy.

Let's just do a 'touch' command 'MyTestFile'. This creates an empty, blank file in the file system. If we do an 'ls -l' command, we should see that the owner of the file is rtracy and my default group--my primary group, which is rtracy as well--is the owning group of MyTestFile.

I have read/write access, group has read/write access, and other authenticated users should have only read access to the file. However, watch what happens. I'm going to 'su' to the 'ksanders -' user, not to root, but to ksanders. Provide her password. Oops. Try it again. I think I typed the password wrong. Try that. That's better.

If I print the working directory, we can see that I'm still in the same directory. I'm in the RandD directory in /mnt/shared. Do an 'ls' command. There's the MyTestFile. Watch what happens if I run the 'rm' command to remove 'MyTestFile'. It says "Do you want to remove this regular empty file?" Sure, go ahead. It did it. It's gone.

The reason for that is because ksanders is a member of the RandD group, and the RandD group has read, write, and execute access to the RandD folder. Even though ksanders was not the file's owner and was not a member of the owning group of the file, ksanders was still able to delete that file.

You can imagine in a shared folder situation, that could be a mess if you had somebody who was a clean freak and going through and thinking they needed to clean up everybody's files for them. You could lose a lot of data that would be otherwise very important to save.

What we want to do is fix this by enabling the sticky bit on the parent directory of these files, which is the /RandD directory. When we do this, then only the user who owns a file will be allowed to delete a file. Let's go up a level. We do need to 'exit' out. We need to switch to our root user account this time. Oops. One more time. I typed the password wrong. There we go.

Now I want to 'cd /mnt/shared'. Run the 'ls -l' command to view the current mode of the RandD folder, and let's change this by running 'chmod'. Just as before, we add a special permission to the mode of the RandD directory by adding an extra digit at the beginning of the mode. In this case, we want to add just a '1'.

Remember, a value of 1 for a special permission enables the sticky bit in the mode. Then we want to leave the rest of the mode the way it is. We want to use a '7' for the directory owner. That gives us read, write, and execute like we had before, and we also want to use '7' for the owning group. That provides read, write, and execute for the RandD group.

We'll block access to this directory by any other authenticated user on the system by adding a '0' at the end. Enter. Oops. I forgot the name of the directory. That would be good to do. 'RandD/'. Let's run 'ls -l' again.

Notice that there is a T over here. This T indicates that the sticky bit has been enabled. Let's verify that it's actually doing what it's supposed to do. I'm going to enter 'exit' to get out of the root account. We're now back to rtracy. Let's switch into the 'RandD/' directory. Let's create a new file again. Let's do 'touch anothernewfile'.

We'll run the 'ls -l' command again, and it has exactly the same mode as the first file that we created in this directory. But now let's switch to the ksanders user again and try to delete that file.

I'm in the RandD directory. There's the file. Let's try to delete that this time. Prompted, "Do you want to delete this file?" Yep, I sure do. Ah, I cannot. "Operation not permitted." That's because with the sticky bit set, only that file's owner is allowed to delete it.

Summary 10:35-10:49

That's it for this demonstration. In this demo, we talked about configuring special permissions. We looked at setting the SUID and SGID special permissions on a file in the file system, and then we ended this demonstration by configuring the sticky bit permission on a directory in the file system.
