File permissions and ownership

Hi there. It's great to have you back!
Let's continue to learn more about how to work in Linux as a security analyst.
In this video, we'll explore file and directory permissions.
We'll learn how Linux represents permissions and how you can check for the permissions associated with files and directories.

Permissions are the type of access granted for a file or directory.
Permissions are related to authorization.
Authorization is the concept of granting access to specific resources in a system.
Authorization allows you to limit access to specified files or directories.
A good rule to follow is that data access is on a need-to-know basis.
You can imagine the security risk it would impose if anyone could access or modify anything they wanted to on a system.

There are three types of permissions in Linux that an authorized user can have.
The first type of permission is read.
On a file, read permissions means contents on the file can be read.
On a directory, this permission means you can read all files in that directory.
Next are write permissions.
Write permissions on a file allow modifications of contents of the file.
On a directory, write permissions indicate that new files can be created in that directory.
Finally, there are also execute permissions.
Execute permissions on files mean that the file can be executed if it's an executable file.
Execute permissions on directories allow users to enter into a directory and access its files.

Permissions are granted for three different types of owners.
The first type is the user.
The user is the owner of the file.
When you create a file, you become the owner of the file, but the ownership can be changed.
Group is the next type.
Every user is a part of a certain group.
A group consists of several users, and this is one way to manage a multi-user environment.
Finally, there is other.
Other can be considered all other users on the system.
Basically, anyone else with access to the system belongs to this group.
In Linux, file permissions are represented with a 10-character string.
For a directory with full permissions for the user group, this string would be: drwxrwxrwx.

Let's examine what this means more closely.
The first character indicates the file type.
As shown in this example, d is used to indicate it is a directory.
If this character contains a hyphen instead, it would be a regular file.

The second, third, and fourth characters indicate the permissions for the user.

- In this example,
- r indicates the user has read permissions,
- w indicates the user has write permissions,
- x indicates the user has execute permissions.

If one of these permissions was missing, there would be a hyphen instead of the letter.

In the same way, the fifth, sixth, and seventh characters indicate permissions for the next owner type group.
As it shows here, the type group also has read, write, and execute permissions.
There are no hyphens to indicate that any of these permissions haven't been granted.
Finally, the eighth through tenth characters indicate permissions for the last owner type: other.
They also have read, write, and execute permissions in this example.

Ensuring files and directories are set with their appropriate access permissions is critical to protecting sensitive files and maintaining the overall security of a system.
For example, payroll departments handle sensitive information.
If someone outside of the payroll group could read this file, this would be a privacy concern.
Another example is when the user, the group, and other can all write to a file.
This type of file is considered a world-writable file.
World-writable files can pose significant security risks.

So how do we check permissions?
First, we need to understand what options are.
Options modify the behavior of the command.
The options for a command can be a single letter or a full word.
Checking permissions involves adding options to the ls command.
First, ls -l displays permissions to files and directories.
You might also want to display hidden files and identify their permissions.
Hidden files, which begin with a period before their name, don't
normally appear when you use ls to display file contents.
Entering ls -a displays hidden files.
Then you can combine these two options to do both.
Entering ls -la displays permissions to files and directories, including hidden files.

Let's get into Bash and try out these options.
Right now, we're in the project subdirectory.
First, let's use the ls command to display its contents.
The output displays the files in this directory, but we don't know anything about their permissions.
By using ls -l instead, we get expanded information on these files. Let's do this.
The file names are now on the right side of each row.
The first piece of information in each row shows the permissions in the format that we discussed earlier.
Since these are all files and not directories, notice how the first character is a hyphen.
Let's focus on one specific file: project1.txt.

The second through fourth characters of its permissions show us the user has both read and write permissions but lacks execute permissions.
In both the fifth through seventh characters and eighth through tenth characters, the sequence is r--.
This means group and other have only read privileges.
After the permissions, ls -l first displays the username.
Here, that's us, analyst.
Next comes the group name; in our case, the security group.
Now let's use ls -a
The output includes two more files—hidden files with the names: .hidden1.txt and .hidden2.txt
Finally, we can also use ls -la to show the permissions for all files, including these hidden files.

I thought that was pretty interesting. Did you?
You now know a little more about file permissions and ownership.
This will be helpful when working in security because monitoring and setting correct permissions is essential for protecting information.
Take a small break and meet me in the next video.

Revision #1
Created 2023-07-06 13:21:23 UTC by naruzkurai
Updated 2023-07-06 17:30:26 UTC by naruzkurai