Manage directories and files

Previously, you explored how to manage the file system using Linux commands. The following commands were introduced: *mkdir*, *rmdir*, *touch*, *rm*, *mv*, and *cp*. In this reading, you'll review these commands, the nano text editor, and learn another way to write to files.

Creating and modifying directories

mkdir

The *mkdir* command creates a new directory. Like all of the commands presented in this reading, you can either provide the new directory as the absolute file path, which starts from the root, or as a relative file path, which starts from your current directory.

For example, if you want to create a new directory called *network* in your */home/analyst/logs* directory, you can enter *mkdir /home/analyst/logs/network* to create this new directory. If you're already in the */home/analyst/logs* directory, you can also create this new directory by entering *mkdir network*.

Pro Tip: You can use the *ls* command to confirm the new directory was added.

rmdir

The *rmdir* command removes, or deletes, a directory. For example, entering *rmdir /home/analyst/logs/network* would remove this empty directory from the file system.

Note: The *rmdir* command cannot delete directories with files or subdirectories inside. For example, entering *rmdir /home/analyst* returns an error message.

Creating and modifying files

touch and rm

The *touch* command creates a new file. This file won't have any content inside. If your current directory is */home/analyst/reports*, entering *touch permissions.txt* creates a new file in the *reports* subdirectory called *permissions.txt*.

The *rm* command removes, or deletes, a file. This command should be used carefully because it's not easy to recover files deleted with *rm*. To remove the permissions file you just created, enter *rm permissions.txt*.

Pro Tip: You can verify that *permissions.txt* was successfully created or removed by entering *ls*.

mv and cp

You can also use *mv* and *cp* when working with files. The *mv* command moves a file or directory to a new location, and the *cp* command copies a file or directory into a new location. The first argument after *mv* or *cp* is the file or directory you want to move or copy, and the second argument is the location you want to move or copy it to.

To move *permissions.txt* into the *logs* subdirectory, enter *mv permissions.txt /home/analyst/logs*. Moving a file removes the file from its original location. However, copying a file doesn't remove it from its original location. To copy *permissions.txt* into the *logs* subdirectory while also keeping it in its original location, enter *cp permissions.txt /home/analyst/logs*.

Note: The *mv* command can also be used to rename files. To rename a file, pass the new name in as the second argument instead of the new location. For example, entering *mv permissions.txt perm.txt* renames the *permissions.txt* file to *perm.txt*.

nano text editor

nano is a command-line file editor that is available by default in many Linux distributions. Many beginners find it easy to use, and it's widely used in the security profession. You can perform multiple basic tasks in nano, such as creating new files and modifying file contents.

To open an existing file in nano from the directory that contains it, enter *nano* followed by the file name. For example, entering *nano permissions.txt* from the */home/analyst/reports* directory opens a new nano editing window with the *permissions.txt* file open for editing. You can also provide the absolute file path to the file if you're not in the directory that contains it.

You can also create a new file in nano by entering *nano* followed by a new file name. For example, entering *nano authorized_users.txt* from the */home/analyst/reports* directory creates the *authorized_users.txt* file within that directory and opens it in a new nano editing window.

Since there isn't an auto-saving feature in nano, it's important to save your work before exiting. To save a file in nano, use the keyboard shortcut *Ctrl + O*. You'll be prompted to confirm the file name before saving. To exit out of nano, use the keyboard shortcut *Ctrl + X*.

Note: Vim and Emacs are also popular command-line text editors.

Standard output redirection

There’s an additional way you can write to files. Previously, you learned about standard input and standard output. **Standard input** is information received by the OS via the command line, and **standard output** is information returned by the OS through the shell.

You’ve also learned about piping. **Piping** sends the standard output of one command as standard input to another command for further processing. It uses the pipe character (|).

In addition to the pipe (|), you can also use the right angle bracket (>) and double right angle bracket (>>) operators to redirect standard output.

When used with *echo*, the > and >> operators can be used to send the output of *echo* to a specified file rather than the screen. The difference between the two is that > overwrites your existing file, and >> adds your content to the end of the existing file instead of overwriting it. The > operator should be used carefully, because it’s not easy to recover overwritten files.

When you’re inside the directory containing the *permissions.txt* file, entering *echo* "*last updated date*" >> *permissions.txt* adds the string “last updated date” to the file contents. Entering *echo* "*time*" > *permissions.txt* after this command overwrites the entire file contents of *permissions.txt* with the string “time”.

Note: Both the > and >> operators will create a new file if one doesn’t already exist with your specified name.

Key takeaways

Knowing how to manage the file system in Linux is an important skill for security analysts. Useful commands for this include: *mkdir*, *rmdir*, *touch*, *rm*, *mv*, and *cp*. When security analysts need to write to files, they can use the nano text editor, or the > and >> operators.

Revision #1
Created 2023-07-06 13:08:28 UTC by naruzkurai
Updated 2023-07-06 13:08:39 UTC by naruzkurai