LINUX FUNDAMENTALS NOTES

NOTE: - ALL THE COMMANDS ARE CASE-SENSITIVE & ALSO THE INPUT ARGUMENTS OR VALUES

echo: IT OUTPUT ANY TEXT THAT WE INPUTS. E.g: "echo hello"

whoami: IT RETURNS CURRENT USERNAME WE ARE LOGGED IN AS

ls: list all files in the current directory or the given directory path as input

E.G. LS [FILE LOCATION]

cd: change the current working directory

E..G CD [FILE LOCATION]

cat: it is used to read and print the contents of a file

cat [filename]

pwd: it is used to find the current working directory and it's complete path also

find: the command is used to find the file name in each and every folder and there subfolder

find -name [filename] //you can special wildcards like *, ?, _

find -name *.[extension]

grep: It is used to find a particular text or value in a given file

grep [finding text] [file name]

wc: it is used to count the number of lines in the file or number of characters also abased on flag

wc -l [file name]

OPERATORS IN THE LINUX:

Operator "&"

This operator allows us to execute commands in the background. For example, let's say we want to copy a large file. This will obviously take quite a long time and will leave us unable to do anything else until the file successfully copies.

The "&" shell operator allows us to execute a command and have it run in the background (such as this file copy) allowing us to do other things!

Operator "&&"

This shell operator is a bit misleading in the sense of how familiar is to its partner "&". Unlike the "&" operator, we can use "&&" to make a list of commands to run for example command1 && command2. However, it's worth noting that command2 will only run if command1 was successful.

Operator ">"

This operator is what's known as an output redirector. What this essentially means is that we take the output from a command we run and send that output to somewhere else.

A great example of this is redirecting the output of the echo command that we learned in Task 4. Of course, running something such as echo howdy will return "howdy" back to our terminal — that isn't super useful. What we can do instead, is redirect "howdy" to something such as a new file!

Let's say we wanted to create a file named "welcome" with the message "hey". We can run echo hey > welcome where we want the file created with the contents "hey"

Using the > Operator

tryhackme@linux1:~$ echo hey > welcome

Using cat to output the "welcome" file

tryhackme@linux1:~$ cat welcome

hey

Note: If the file i.e. "welcome" already exists, the contents will be overwritten!

Operator ">>"

This operator is also an output redirector like in the previous operator (>) we discussed. However, what makes this operator different is that rather than overwriting any contents within a file, for example, it instead just puts the output at the end.

Following on with our previous example where we have the file "welcome" that has the contents of "hey". If were to use echo to add "hello" to the file using the > operator, the file will now only have "hello" and not "hey".

The >> operator allows to append the output to the bottom of the file — rather than replacing the contents like so:

Using the >> Operator

tryhackme@linux1:~$ echo hello >> welcome

Using cat to output the "welcome" file

tryhackme@linux1:~$ cat welcome

hey

hello

SSH: This protocol is called Secure Shell or SSH for short and is the common means of connecting to and interacting with the command line of a remote Linux machine. What is SSH & how Does it Work?

Secure Shell or SSH simply is a protocol between devices in an encrypted form. Using cryptography, any input we send in a human-readable format is encrypted for travelling over a network -- where it is then unencrypted once it reaches the remote machine, such as in the diagram below.

SSH allows us to remotely execute commands on another device remotely.

Any data sent between the devices is encrypted when it is sent over a network such as the Internet

Using SSH to Login to Your Linux Machine

The syntax to use SSH is very simple. We only need to provide two things:

1. The IP address of the remote machine

2. Correct credentials to a valid account to login with on the remote machine

Note: When you type a password into an ssh login prompt there is no visible feedback -- you will not be able to see any text or symbols appear as you type the password. It is still working, however, so just type the password and press enter to login.

Files and folders with "." are hidden files.

The Man(ual) Page

The manual pages are a great source of information for both system commands and applications available on both a Linux machine, which is accessible on the machine itself and online.

To access this documentation, we can use the mancommand and then provide the command we want to read the documentation for. Using our ls example, we would use man ls to view the manual pages for ls

Creating Files and Folders (touch, mkdir)

Creating files and folders on Linux is a simple process. First, we'll cover creating a file. The touch command takes exactly one argument -- the name we want to give the file we create. For example, we can create the file "note" by using touch note. It's worth noting that touch simply creates a blank file. You would need to use commands like echo or text editors such as nano to add content to the blank file.

Using touch to create a new file

tryhackme@linux2:~$ touch note

tryhackme@linux2:~$ ls

folder1 note

This is a similar process for making a folder, which just involves using the mkdir command and again providing the name that we want to assign to the directory. For example, creating the directory "mydirectory" using mkdir mydirectory.

Creating a new directory with mkdir

tryhackme@linux2:~$ mkdir mydirectory

tryhackme@linux2:~$ ls

folder1 mydirectory note

Removing Files and Folders (rm)

rm is extraordinary out of the commands that we've covered so far. You can simply remove files by using rm. However, you need to provide the -R switch alongside the name of the directory you wish to remove.

Using rm to remove a file

tryhackme@linux2:~$ rm note

tryhackme@linux2:~$ ls

folder1 mydirectory

Using rm recursively to remove a directory

tryhackme@linux2:~$ rm -R mydirectory

tryhackme@linux2:~$ ls

folder1

Copying and Moving Files and Folders (cp, mv)

Copying and moving files is an important functionality on a Linux machine. Starting with cp, this command takes two arguments:

1. the name of the existing file

2. the name we wish to assign to the new file when copying

cp copies the entire contents of the existing file into the new file. In the screenshot below, we are copying "note" to "note2".

Using cp to copy a file

tryhackme@linux2:~$ cp note note2

tryhackme@linux2:~$ ls

folder1 note note2

Moving a file takes two arguments, just like the cp command. However, rather than copying and/or creating a new file, mv will merge or modify the second file that we provide as an argument. Not only can you use mv to move a file to a new folder, but you can also use mvto rename a file or folder. For example, in the screenshot below, we are renaming the file "note2" to be named "note3". "note3" will now have the contents of "note2".

Using mv to move a file

tryhackme@linux2:~$ mv note2 note3

tryhackme@linux2:~$ ls

folder1 note note3

Determining File Type

What is often misleading and often catches people out is making presumptions from files as to what their purpose or contents may be. Files usually have what's known as an extension to make this easier. For example, text files usually have an extension of ".txt". But this is not necessary.

So far, the files we have used in our examples haven't had an extension. Without knowing the context of why the file is there -- we don't really know its purpose. Enter the file command. This command takes one argument. For example, we'll use file to confirm whether or not the "note" file in our examples is indeed a text file, like so file note.

Using file to determine the contents of a file

tryhackme@linux2:~$ file note

note: ASCII text

Switching Between Users

Switching between users on a Linux install is easy work thanks to the su command. Unless you are the root user (or using root permissions through sudo), then you are required to know two things to facilitate this transition of user accounts:

The user we wish to switch to

The user's password

The su command takes a couple of switches that may be of relevance to you. For example, executing a command once you log in or specifying a specific shell to use.

Simply, by providing the -l switch to su, we start a shell that is much more similar to the actual user logging into the system - we inherit a lot more properties of the new user, i.e., environment variables and the likes.

Using su to switch to user2 interactively

tryhackme@linux2:~$ su user2

Password:

user2@linux2:/home/tryhackme$

For example, when using su to switch to "user2", our new session drops us into our previous user's home directory.

Using su to switch to user2 interactively

tryhackme@linux2:~$ su -l user2

Password:

user2@linux2:~$ pwd

user2@:/home/user2$

Where now, after using -l, our new session has dropped us into the home directory of "user" automatically.

Note:- ls -lh is used to see the user having access or permission to a file

<u>Common directories</u>

/etc

This root directory is one of the most important root directories on your system. The etc folder (short for etcetera) is a commonplace location to store system files that are used by your operating system.

For example, the sudoers file highlighted in the screenshot below contains a list of the users & groups that have permission to run sudo or a set of commands as the root user.

Also highlighted below are the "passwd" and "shadow" files. These two files are special for Linux as they show how your system stores the passwords for each user in encrypted formatting called sha512.

Some notable contents of the /etc directory

tryhackme@linux2:/etc$ ls

shadow passwd sudoers sudoers.d

/var

The "/var" directory, with "var" being short for variable data, is one of the main root folders found on a Linux install. This folder stores data that is frequently accessed or written by services or applications running on the system. For example, log files from running services and applications are written here (/var/log), or other data that is not necessarily associated with a specific user (i.e., databases and the like).

Some notable contents of the /var directory

tryhackme@linux2:/var$ ls

backups log opt tmp

/root

Unlike the /home directory, the /root folder is actually the home for the "root" system user. There isn't anything more to this folder other than just understanding that this is the home directory for the "root" user. But, it is worth a mention as the logical presumption is that this user would have their data in a directory such as "/home/root" by default.

tmp

This is a unique root directory found on a Linux install. Short for "temporary", the /tmp directory is volatile and is used to store data that is only needed to be accessed once or twice. Similar to the memory on your computer, once the computer is restarted, the contents of this folder are cleared out.

What's useful for us in pentesting is that any user can write to this folder by default. Meaning once we have access to a machine, it serves as a good place to store things like our enumeration scripts.

Some notable contents of the /tmp directory

root@linux2:/tmp# ls

todelete trash.txt rubbish.bin

Terminal Text Editors

Nano

It is easy to get started with Nano! To create or edit a file using nano, we simply use nano filename -- replacing "filename" with the name of the file you wish to edit.

Once we press enter to execute the command, nano will launch! Where we can just begin to start entering or modifying our text. You can navigate each line using the "up" and "down" arrow keys or start a new line using the "Enter" key on your keyboard.

Nano has a few features that are easy to remember & covers the most general things you would want out of a text editor, including:

Searching for text

Copying and Pasting

Jumping to a line number

Finding out what line number you are on

You can use these features of nano by pressing the "Ctrl" key (which is represented as an ^ on Linux) and a corresponding letter. For example, to exit, we would want to press "Ctrl" and "X" to exit Nano.

VIM

VIM is a much more advanced text editor. Some of VIM's benefits, albeit taking a much longer time to become familiar with, includes:

Customisable - you can modify the keyboard shortcuts to be of your choosing

Syntax Highlighting - this is useful if you are writing or maintaining code, making it a popular choice for software developers

VIM works on all terminals where nano may not be installed

Downloading Files

Wget

This command allows us to download files from the web via HTTP -- as if you were accessing the file in your browser. We simply need to provide the address of the resource that we wish to download. For example, if I wanted to download a file named "myfile.txt" onto my machine, assuming I knew the web address it -- it would look something like this:

wget https://assets.tryhackme.com/additional/linux-fundamentals/part3/myfile.txt

Transferring Files From Your Host - SCP (SSH)

Secure copy, or SCP, is just that -- a means of securely copying files. Unlike the regular cp command, this command allows you to transfer files between two computers using the SSH protocol to provide both authentication and encryption.

Working on a model of SOURCE and DESTINATION, SCP allows you to:

Copy files & directories from your current system to a remote system

Copy files & directories from a remote system to your current system

Provided that we know usernames and passwords for a user on your current system and a user on the remote system.

With this information, let's craft our scp command (remembering that the format of SCP is just SOURCE and DESTINATION)

scp important.txt ubuntu@192.168.1.30:/home/ubuntu/transferred.txt

And now let's reverse this and layout the syntax for using scpto copy a file from a remote computer that we're not logged into

The command will now look like the following:

scp ubuntu@192.168.1.30:/home/ubuntu/documents.txt notes.txt