ubnt@19cdcb763510:~/scavenger-hunt$ cat clues\_output.txt

### \*-: Moving Day ###

#### Making Space ####

We've been exploring the directories that already exist on the computer. But

what if we want to make our own folders and files? The first thing we need to

do is create a new directory. First go home: `cd ~/scavenger-hunt`. Then do

mkdir saved-clues

What we're going to do is save off all the clues we find in a separate folder

that we created with `mkdir` (make directory). Since the README is clue 1 we

don't need to worry about it. If you've been writing down all the clue

locations, this next part should be easy.

#### Stop Copying Me ####

Let's copy all of the clues we've found so far to our saved-clues folder:

cp clues/15316/clue saved-clues/clue2

cp clues/12345/clue saved-clues/clue3

This copies (`cp`) each clue to the new folder and gives them new names. If we

had just done this

cp clues/15316/clue saved-clues/

cp clues/12345/clue saved-clues/

The second file would overwrite the first, because they have the same name.

#### Keep Your Options Open ####

Linux commands often have options that change how they behave. For instance,

compare `ls -l` to ordinary `ls`. Here the `-l` is an option. You can group

options together like this

ls -lahS

The best way to find out about options is the manpage.

#### Moving On ####

Now let's say we don't like the folder name `saved-clues`. We can just move

(`mv`) it:

mv saved-clues [pick a new name]

Now do an `ls` to see the results of the move. Be careful with `mv`: you can

easily overwrite an existing folder.

#### Find Clue 5 ####

Read the man page for `mv` and find an

option to prevent overwriting. That option is your next hint.

---

### Clue 7: Make Me a Sandwhich ###

https://xkcd.com/149/

#### `sudo` ####

Linux has the concept of a `root` user, which is similar to the administrator

user on Windows. This user is sometimes called the super user. If you want to

do something as the super user, but stay logged in as yourself, there is a

command for that: `sudo`. It stands for "super-user do".

#### Installing Software ####

Sometimes you need a new program. To install software on some versions of Linux

(Ubuntu and Debian), you use the command `apt-get`. On other versions (Fedora,

CentOS) you use the command `yum`. Let's install a text editing program

called `vim`.

apt-get install vim

You should get an error message asking if you are root. This means you don't

have the ability to install software. Instead, try

sudo apt-get install vim

It will ask for a password.

#### Getting the password

The password of your user is the answer of Challenge 0.1 of the course.

It is an easily guessable password found among the list of 60 credentials of the malware Mirai. You can find it online either in a list of bad credentials used in dictionary attacks, or among the source of Mirai. (You can look for a file named `scanner.c`). Another option is to check the Dockerfile I wrote to create the image you are currently using. This can be done by checking DockerHub or GitHub. You can also guess it pretty easily.

Either way, if you find it, you can submit it on iLearn. Or you can continue without sudo.

#### Installing Software 2 ####

Now we have the ability to edit files. Try

vim README.md

from the `scavenger-hunt` directory. Some of the commands for `vim` are a little

strange. For now, just type `:q!` to quit.

#### Finding Clue 8 ####

The hint is the word contained in `/root/secret`.

#### Help. I can't sudo ####

Depending on the system you are using, you may not have permission to use `sudo`.

In this case you can use the hint `denied`.

---

### Clue 10: Pipes ###

#### Piping Information ####

Many commands will print their output. This is called "standard output" or

`stdout`. We saw earlier how you can redirect standard output to a file (`>`).

There is also standard input (`<`). For example, `cat < README.md` is the same

as `cat README.md`. But standard input and output can be chained together using

pipe (`|`). For example, you can count the number of files and folders in a

directory like this:

ls | wc -w

This works by taking the output of `ls` and using it as the input to `wc`.

Another example:

grep ^sand /usr/share/dict/words | wc -l

will print the number of words that start with "sand". The carat `^` symbol

is a regular expression that means "starts with". You can also use `$` for

"ends with".

#### Sort ####

Sometimes you need to sort data alphabetically. You'll notice that the

dictionary file is already sorted. You can create your own unsorted copy like

this:

sort -R /usr/share/dict/words > random\_words

Now you can `sort random\_words` to get back to alphabetical order, or

`sort -r random\_words` for reverse alphabetical order.

#### Finding Clue 11 ####

Use the command `ls -la /usr/bin` to get a big list of files. The 5th column in

that list is the size of the file in bytes. Find the sort command to print the

list of files with the largest file first, and then the rest indescending order.

Your hint is the options you had to use. You'll need to use double quotes for

your hint. For example, if your command was `sort -a -b -c`, your hint would

be

python3 next\_clue.py [secret] 11 "-a -b -c"

---

### Clue 5: Is There an Echo in Here? ###

#### `echo` ####

Sometimes we want the computer to repeat back the results of some command. Try

echo hello

The most basic thing `echo` will do is repeat back whatever you type.

#### Redirect ####

You can use this to create a small file, or start a new file:

echo My bologna has a first name > oscar.txt

If you look in oscar.txt you will see exactly what you typed. The `>` symbol

used here is called a redirect. It redirects whatever would normally be printed

to the screen to a file. You can try it with other commands:

ls > my\_directory.txt

You can also use `echo` to display what are called environment variables

echo $PATH

echo $HOME

The `HOME` variable should make sense at this point.

#### Find Clue 6 ####

The `PATH` variable tells the computer where programs are. Each path that could

contain a program is placed between colons. Your hint for the next clue is the

first path listed in your `PATH`.

---

### Clue 2: The Lay of the Land ###

#### `pwd` ####

What if we get lost and need to know where we are? Just type `pwd` (print

working directory). This should print something like this:

/home/user/scavenger-hunt/clues/123456

We are five folders deep, in a folder named `123456`.

#### `cd` ####

Change directory is extremely useful, but it can also be confusing. We

already saw how you can move up one directory (`cd ..`) or down one directory

(`cd [dir]`). You can move up or down any number of directories in a single

command like this (won't actually work here):

cd ../../../one/two/

You would navigate up 3 directories relative to where you are, then down

into directory one and then two. This is what's known as a relative path: it

depends on where you start where you will end up. The other way to change

directories is with absolute paths. Try this:

cd /

Look around and see what's there. This is known as the root path. You can

explore the entire file system from here, using just `ls` and `cd`.

#### Find Clue 3 ####

To find the next clue, go to the `/usr` directory and count the number of

subdirectories. This is a hint to your next clue location. Go to the

`scavenger-hunt` directory, and type

python3 next\_clue.py [secret number] [next clue number] [hint]

So, if there were 5 directories, we would type

python3 next\_clue.py 42 3 5

since our secret number is 42, we want to find clue 3, and our hint is 5.

The location of our next clue should be printed. If you get the hint wrong,

an incorrect clue will be printed. This is an example of what is known as GIGO

(garbage-in, garbage-out). Helpful tip: you can open more than one terminal

window, or more than one tab in your current window with `CTRL-SHIFT-T`.

#### `less` ####

Less is a program that allows you to view files in a terminal. Unlike `cat`,

you can scroll through the file using up, down, page up, \*etc\*. After you

navigate to clue 3, do `less clue`. The name Less is a play on More, a similar

program. More is older and is so named because you could press enter to

see more text.

---

### Clue 11: The Final Frontier ###

#### Finding the Final Clue ####

Using everything you've learned so far, and the fact that the real clues are

different from the fake clues, find the final clue!

---

### Clue 3: Humans vs. Machines ###

#### Binary vs. Text ####

There are two basic types of files: binary and text. Text can be read by both

humans and the computer, and is sometimes referred to as "human-readable". For

example, the file you are reading right now is text.

Binary is a number system that uses only 0 and 1 as digits. For example, 42 is

represented as 101010 in binary. Each digit is called a "bit". Eight bits is

called a "byte". There are 256 possible bytes (2^8). Bytes are a fundamental

unit of measurement in computing (\*e.g.\* file sizes are in bytes). Computers

use a shorthand for each byte called "hexadecimal" or more briefly "hex". In

hex there are sixteen digits, the usual 0-9 and also A-F. A is equal to 10,

B to 11, \*etc\*. Sometimes we write a `0x` in front of a hex number to indicate

we are using hex: 42 is `0x2A`.

If you ever look at a file and see a bunch of "garbage", you are probably

looking at a binary file. The content isn't intended for you: it's for the

computer. Binary files are sometimes referred to as "machine-readable".

#### `/bin` ####

One place you can always find binaries on a Linux system is in `/bin`. These

binaries are programs: if you `ls` in `/bin` you may recoginze some of them

(including `ls` itself). This is also a convenient way to get a list of

commands.

If you want to see the garbage view of a binary, you can `cat` or

`less` one of them. You can even `cat cat` or `less less`. On some Linux

systems you can see the hex itself with `hexdump`.

#### `/etc` ####

This directory is named after the latin \*et cetera\* but is usally pronounced

"et see". There are many text (and some binary) files here that are used to

configure the system. Humans and computers can both read these files to find

out how to configure the system.

For example, look at the `/etc/fstab` file. This describes how the filesystem

is mapped to the hard drive.

#### Find Clue 4 ####

Your hint for clue 4 is the file `/etc/hostname`. This file contains a single

word, which is the name of your computer. This name is your hint. Remember we

can find the next hint by typing

python3 next\_clue.py [secret number] [next clue number] [hint]

---

### Clue 12: Success! ###

#### You Found All the Clues ####

This is just the start of learning Linux. Explore, google things, break things.

You can also start learning shell scripting (executing multiple commands), or

even Python programming. Just type

python3

and you'll get an interactive programming environment. You can use `vim` or a

text editor to write your python code, then execute it:

python3 my\_code.py

#### Bash scripting ####

To create a bash script that automate any tasks you could do in your shell you can do the following

touch my\_script.sh

chmod +x my\_script.sh

Edit the file with the text editor of your choice (like vim) and copy the following lines

```

#!/bin/bash

echo "Hello world!"

```

To run the script you can type in your shell:

./my\_script.sh

#### Bonus question

Write a script that explores the `clues` and print files that are real clues (without printing the fake clues).

#### Challenges

Challenge 0.1 is finding the password of the user `ubnt`.

Challenge 0.2 is the script you wrote to find all clues at once.

---

### Clue 6: Which `which` is which? ###

#### `which` ####

In the previous clue we learned about `PATH`. This tells the computer where to

find binary programs we can run. But, how do we find where a specific program

is? The answer is `which`:

which mv

This should print `/bin/mv`. This tells us that the `mv` command is installed

in the `/bin/` directory. `which` itself is a program so you could try:

which which

#### Finding Clue 7 ####

`touch` is a program that creates an empty file. Your next hint is the location of the `touch` program.

---

### Clue 8: Counting Words ###

#### `wc` ####

Word count (`wc`) is a useful program. You can use it to tell how many lines,

words, and/or characters are in a file:

wc README.md

This will print the number of lines, words, and characters, in that order. If

you just want one of those, you can use `-l`, `-w`, or `-c`.

#### Finding Clue 9 ####

Check to see if you have the file `/usr/share/dict/words` installed. If not,

run this:

sudo apt-get install ispell

Now there is a file that acts as a dictionary for spell-checking:

`/usr/share/dict/words`. Your next clue is the number of words in the

dictionary.

---

### Clue 9: Searching High and Low ###

#### `grep` ####

Searching files is another useful trick. Try this:

grep secret README.md

This will print out every line that contains the word "secret". `grep` stands

for Gnu Regular Expression Parsing. Gnu is an umbrella organization that

publishes open source software. A regular expression is a pattern that matches

text. In this case our regular expression is just "secret", and will only find

exact matches. Regular expressions can be more powerful and/or complicated. For

example,

grep m.n README.md

will find any line where the letters m and n exist with a single character

between them. Check the man page for many interesting features of `grep`.

#### Finding Clue 10 ####

The next hint is the word that occurs after "tactful" in

`/usr/share/dict/words`. There's a specific option for `grep` that will make

this easy.