

Name: Gaurav Jaiswal

Reg no: 20BKT0028

CYBER SECURITY AND ETHICAL HACKING ASSESSMENT 2

Task 1: File and Directory Manipulation

COMMANDS:

```
mkdir my_directory  
cd my_directory  
touch my_file.txt  
ls  
mv my_file.txt new_file.txt  
less new_file.txt  
echo "Hello, World!" >> new_file.txt  
mkdir backup  
mv new_file.txt backup/  
ls backup/  
rm -r backup/
```

OUTPUT:

```
(kali㉿kali)-[~]
└─$ mkdir my_directory

(kali㉿kali)-[~]
└─$ cd my_directory

(kali㉿kali)-[~/my_directory]
└─$ touch my_file.txt

(kali㉿kali)-[~/my_directory]
└─$ ls

my_file.txt

(kali㉿kali)-[~/my_directory]
└─$ mv my_file.txt new_file.txt

(kali㉿kali)-[~/my_directory]
└─$ less new_file.txt

zsh:1: unmatched "
!done (press RETURN)

(kali㉿kali)-[~/my_directory]
└─$ mkdir backup

(kali㉿kali)-[~/my_directory]
└─$ mv new_file.txt backup/

(kali㉿kali)-[~/my_directory]
└─$ ls backup/

new_file.txt

(kali㉿kali)-[~/my_directory]
└─$ rm -r backup/
```

EXPLANATION:

`mkdir my_directory`: This command creates a new directory called "my_directory" in the current location.

`cd my_directory`: This command allows you to navigate into the "my_directory" directory.

`touch my_file.txt`: This command creates an empty file called "my_file.txt" in the current directory.

ls: This command lists all the files and directories in the current directory.

mv my_file.txt new_file.txt: This command renames the file "my_file.txt" to "new_file.txt".

less new_file.txt: This command displays the content of "new_file.txt" using the pager tool "less". You can scroll through the content using the arrow keys, and press "q" to exit.

echo "Hello, World!" >> new_file.txt: This command appends the text "Hello, World!" to the file "new_file.txt".

mkdir backup: This command creates a new directory called "backup" within the "my_directory" directory.

mv new_file.txt backup/: This command moves the file "new_file.txt" to the "backup" directory.

ls backup/: This command lists the files and directories in the "backup" directory to verify that "new_file.txt" is now located there.

rm -r backup/: This command deletes the "backup" directory and all its contents recursively (the "-r" flag is used to remove directories).

Task 2: Permissions and Scripting

COMMANDS:

touch my_script.sh

nano my_script.sh

#!/bin/bash

echo "Welcome to my script!"

echo "Today's date is \$(date)."

chmod +x my_script.sh

./my_script.sh

OUTPUT:

```
(kali㉿kali)-[~/my_directory]
└─$ touch my_script.sh

(kali㉿kali)-[~/my_directory]
└─$ nano my_script.sh

(kali㉿kali)-[~/my_directory]
└─$ chmod +x my_script.sh

(kali㉿kali)-[~/my_directory]
└─$ ./my_script.sh

Welcome to my script!
Today's date is Sun May 28 05:59:49 AM EDT 2023.

(kali㉿kali)-[~/my_directory]
└─$
```

EXPLANATION:

`touch my_script.sh`: This command creates a new file called "my_script.sh" in the current directory.

`nano my_script.sh`: This command opens the file "my_script.sh" in the nano text editor, allowing you to add and edit the script's content. You can use any text editor of your choice.

`#!/bin/bash, echo "Welcome to my script!", echo "Today's date is $(date)."`: These lines are added to the "my_script.sh" file. The first line is a shebang that specifies the interpreter to be used (in this case, bash). The subsequent lines are commands to be executed when the script is run.

`chmod +x my_script.sh`: This command makes the "my_script.sh" file executable by adding execute permissions.

`./my_script.sh`: This command runs the "my_script.sh" script, executing the commands within it and displaying the output.

Task 3: Command Execution and Pipelines

COMMANDS:

ps aux

ps aux | grep bash

ps aux | grep bash | wc -l

OUTPUT:

```
(kali㉿kali)-[~/my_directory]
$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.2	0.3	167940	12320	?	Ss	05:44	0:02	/sbin/init splash
root	2	0.0	0.0	0	0	?	S	05:44	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	05:44	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	05:44	0:00	[rcu_par_gp]
root	5	0.0	0.0	0	0	?	I<	05:44	0:00	[slub_flushwq]
root	6	0.0	0.0	0	0	?	I<	05:44	0:00	[netns]
root	8	0.0	0.0	0	0	?	I<	05:44	0:00	[kworker/0:0H-events_highpri]
root	10	0.0	0.0	0	0	?	I<	05:44	0:00	[mm_percpu_wq]
root	11	0.0	0.0	0	0	?	I	05:44	0:00	[rcu_tasks_kthread]
root	12	0.0	0.0	0	0	?	I	05:44	0:00	[rcu_tasks_rude_kthread]
root	13	0.0	0.0	0	0	?	I	05:44	0:00	[rcu_tasks_trace_kthread]
root	14	0.0	0.0	0	0	?	S	05:44	0:00	[ksoftirqd/0]
root	15	0.0	0.0	0	0	?	I	05:44	0:00	[rcu_preempt]
root	16	0.0	0.0	0	0	?	S	05:44	0:00	[migration/0]
root	18	0.0	0.0	0	0	?	S	05:44	0:00	[cpuhp/0]
root	19	0.0	0.0	0	0	?	S	05:44	0:00	[cpuhp/1]
root	20	0.0	0.0	0	0	?	S	05:44	0:00	[migration/1]
root	21	0.0	0.0	0	0	?	S	05:44	0:00	[ksoftirqd/1]
root	23	0.0	0.0	0	0	?	I<	05:44	0:00	[kworker/1:0H-events_highpri]
root	2931	0.0	0.0	0	0	?	I	05:53	0:00	[kworker/2:0-events]
root	2954	0.0	0.0	0	0	?	I	05:53	0:00	[kworker/u8:1-flush-8:0]
root	3020	0.0	0.0	0	0	?	I	05:56	0:00	[kworker/0:1-events]
root	3078	0.0	0.0	0	0	?	I	06:00	0:00	[kworker/2:1-cgroup_destroy]
root	3097	0.0	0.0	0	0	?	I	06:01	0:00	[kworker/0:2-ata_sff]
kali	3099	18.1	0.1	11200	4776	pts/0	R+	06:02	0:00	ps aux

```
(kali㉿kali)-[~/my_directory]
$ ps aux | grep bash
```

kali	3103	0.0	0.0	6332	2136	pts/0	S+	06:02	0:00	grep --color=auto bash
------	------	-----	-----	------	------	-------	----	-------	------	------------------------

```
(kali㉿kali)-[~/my_directory]
$ ps aux | grep bash | wc -l
```

1

```
(kali㉿kali)-[~/my_directory]
$
```

EXPLANATION:

ps aux: This command lists all the processes running on your system.

grep bash: This command filters the output of the previous command and displays only the processes that have "bash" in their name.

wc -l: This command counts the number of lines in the filtered output, providing the total count of processes with "bash" in their name.