COMP4108- Assignment 2
Isaiah Gratwohl, 101152162
w

Part A

3. Address: ffffffffbc8013c0. I used `cat /proc/kallsyms | grep sys_call_table`

5/6/7: `dmesg`

```
[1349340.188494] Rootkit module is loaded!
[1349451.406290] Rootkit module is unloaded!
[1349451.406294] Rootkit module cleanup copmlete.
```

8. With the rootkit loaded, any program that makes use of openat() should use our function instead. To test if it works, I made a blank file called “test.txt”, then after loading the module, ran cat `./test.txt`. `dmesg` then gives as output:

```
[1371262.375784] openat() called for test.txt
```

and we can see that our function is called, where `cat` would normally use openat() to read the file.

9. Since kernel modules run as root, there is little we could do to protect the system once a rootkit is installed. So we should focus on prevention, and detection.

Principle of least privilege (P6) would help with prevention in the first place, as granting only the minimum permissions necessary to various programs would decrease the chance that they are exploited for privilege escalation.

Evidence production (P14) would aid detection if a rootkit were to be installed. Detection tools could monitor sensitive data, such as the control register, and sound an alert if a modification is made. Rootkits that are unaware of this protection would then be revealed.

Part B

execve() function signature:

```
int execve(const char *pathname, char *const _Nullable argv[], char *const _Nullable envp[]);
```

In our hook, we need to know the pathname of the file being executed so we can print it. The first argument of is passed through the rdi register. So we obtain its value with: regs→di.

Then we must compare our eUID to the root_uid provided by insert script. We do this with:

```
if ((int)current_euid().val == root_uid) {...
```

if so, we use `commit_creds` with the new uid/euid of 0.

1. output:

```
[ 8838.928999] Executing /usr/bin/tail
[ 8838.929003] Current UID: 0
[ 8838.929026] Executing /bin/dmesg
[ 8838.929029] Current UID: 0
```

2. output:

```
root@COMP4108-a2:~/a2# su student
student@COMP4108-a2:~/a2$ whoami
student
student@COMP4108-a2:~/a2$ ./insert.sh
insmod: ERROR: could not insert module rootkit.ko: Operation not
permitted
student@COMP4108-a2:~/a2$ sudo ./insert.sh
student@COMP4108-a2:~/a2$ whoami
root
```

Note:the above output doesn’t really prove anything since I found it impossible to insert the module without automatically promoting myself to root.

Part C

Getdents64 function signature:

```
int getdents(unsigned int fd, struct linux_dirent *dirp,unsigned int count);
```

My code contains comments which detail the following:

We retrieve dirp from ret→si. We call the original getdents64 which copies all the directory entries into buffer pointed to by dirp. But even with the pointer, we can’t access those directories since they are in user space. So we must first copy them into kernel memory with copy_from_user. Then, we can just use pointer arithmetic to read each dentry and check if it starts with the magic prefix. If a dentry does not start with the prefix, we copy it into another buffer. Then, we copy this new buffer back into user space with copy_to_user, who will not see the hidden dentries.

1. output:

```
[ 9657.656932] getdents64() hook invoked
[ 9657.656998] entry: rootkit.o
[ 9657.657001] entry: .rootkit.mod.o.cmd
[ 9657.657004] entry: ..
[ 9657.657007] entry: insert.sh
[ 9657.657009] entry: rootkit.c
[ 9657.657012] entry: rootkit.mod.c
[ 9657.657015] entry: rootkit.ko
[ 9657.657019] entry: isaiahgratwohl-assignment2.pdf
[ 9657.657022] entry: .rootkit.ko.cmd
[ 9657.657024] entry: test.txt
[ 9657.657027] entry: Makefile
```

```
[ 9657.657030] entry: modules.order
[ 9657.657033] entry: rootkit.mod.o
[ 9657.657036] entry: .rootkit.o.cmd
[ 9657.657039] entry: eject.sh
[ 9657.657041] entry: .
[ 9657.657044] entry: .rootkit.mod.cmd
[ 9657.657047] entry: $sys$_lol_hidden.txt
[ 9657.657050] entry: Module.symvers
[ 9657.657053] entry: rootkit.mod
[ 9657.657056] entry: .txt
```

2. output:

```
student@COMP4108-a2:~/a2$ ll
ls: cannot access '': No such file or directory
total 196
c????????? ? ?       ?           ?            ? ''
drwxrwxr-x 2 student student  4096 Oct 18 23:21  ./
drwxr-xr-x 9 student student  4096 Oct 18 23:00  ../
-rwxrwxr-x 1 student student   107 Feb  1  2024  eject.sh*
-rwxrwxr-x 1 student student   219 Oct 18 22:30  insert.sh*
-rw-rw-r-- 1 root    student 20480 Oct 18 23:00  isaiahgratwohl-assignment2.pdf
-rw-rw-r-- 1 student student   174 Feb  1  2024  Makefile
-rw-r--r-- 1 root    root       28 Oct 18 23:21  modules.order
-rw-r--r-- 1 root    root        0 Oct 14 12:56  Module.symvers
-rw-rw-r-- 1 student student  9455 Oct 18 23:21  rootkit.c
-rw-r--r-- 1 root    root    13648 Oct 18 23:21  rootkit.ko
-rw-r--r-- 1 root    root      238 Oct 18 23:21  .rootkit.ko.cmd
-rw-r--r-- 1 root    root       28 Oct 18 23:21  rootkit.mod
-rw-r--r-- 1 root    root     1492 Oct 18 23:21  rootkit.mod.c
-rw-r--r-- 1 root    root      112 Oct 18 23:21  .rootkit.mod.cmd
-rw-r--r-- 1 root    root     4536 Oct 18 23:21  rootkit.mod.o
-rw-r--r-- 1 root    root    30946 Oct 18 23:21  .rootkit.mod.o.cmd
-rw-r--r-- 1 root    root    10480 Oct 18 23:21  rootkit.o
-rw-r--r-- 1 root    root    49769 Oct 18 23:21  .rootkit.o.cmd
-rw-r--r-- 1 root    root       20 Oct 14 19:20  test.txt
-rw-r--r-- 1 root    root        0 Oct 17 16:15  .txt
```