COMP 4108 Assignment 2

Due 11:59PM on **Oct 8th**

Name: Michael Macdougall
Student #: 101197828

Section A:

1. No marks
2. No marks
3.

   Command:
   cat kallsyms|grep sys_call_table
   Address: ffffffff92a013c0

```
root@COMP4108-a2:/proc# cat kallsyms|grep sys_call_table
ffffffff92a002a0 R x32_sys_call_table
ffffffff92a013c0 R sys_call_table
ffffffff92a02400 R ia32_sys_call_table
```

4.

```
unsigned long * get_syscall_table_bf(void){
unsigned long *syscall_table;
syscall_table = (unsigned long*)kallsyms_lookup_name("sys_call_table");
return syscall_table;
}
```

5.

   Command:
   make

```
student@COMP4108-a2:~/a2$ make
make -C /lib/modules/5.4.0-171-generic/build M=/home/student/a2 modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-171-generic'
  CC [M]  /home/student/a2/rootkit.o
/home/student/a2/rootkit.c:74:14: warning: ‘magic_prefix’ defined but not used [-Wunuse
d-variable]
   74 | static char* magic_prefix;
      |              ^~~~~~~~~~~~
/home/student/a2/rootkit.c:62:12: warning: ‘root_uid’ defined but not used [-Wunused-va
riable]
   62 | static int root_uid;
      |            ^~~~~~~~
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/student/a2/rootkit.mod.o
  LD [M]  /home/student/a2/rootkit.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-171-generic'
```

6.

   Commands:
   sudo ./inject.sh

lsmod

```
student@COMP4108-a2:~/a2$ sudo ./insert.sh
[sudo] password for student:
student@COMP4108-a2:~/a2$ lsmod
Module                  Size  Used by
rootkit                16384  0
```

7.

Commands:
sudo ./eject.sh
lsmod

```
student@COMP4108-a2:~/a2$ sudo ./eject.sh
student@COMP4108-a2:~/a2$ lsmod
Module                  Size  Used by
intel_rapl_msr         20480  0
intel_rapl_common      24576  1 intel_rapl_msr
kvm_intel             286720  0
kvm                   667648  1 kvm_intel
```

8.

See rootkit.c for this part
Commands:
touch cloth.txt
cat cloth.txt
sudo tail /var/log/syslog

```
student@COMP4108-a2:~/a2$ touch cloth.txt
student@COMP4108-a2:~/a2$ cat cloth.txt
student@COMP4108-a2:~/a2$ sudo tail /var/log/syslog
Oct  8 11:20:24 COMP4108-a2 kernel: [  556.813284] Rootkit module cleanup copmlete.
Oct  8 11:20:34 COMP4108-a2 kernel: [  565.964100] rootkit: unknown parameter 'root_uid' ignored
Oct  8 11:20:34 COMP4108-a2 kernel: [  565.964106] rootkit: unknown parameter 'magic_prefix' igno
red
Oct  8 11:20:34 COMP4108-a2 kernel: [  565.964256] Rootkit module initializing.
Oct  8 11:20:34 COMP4108-a2 kernel: [  565.977708] Rootkit module is loaded!
Oct  8 11:21:12 COMP4108-a2 anacron[337]: Job `cron.weekly' started
Oct  8 11:21:12 COMP4108-a2 anacron[3540]: Updated timestamp for job `cron.weekly' to 2024-10-08
Oct  8 11:21:12 COMP4108-a2 anacron[337]: Job `cron.weekly' terminated
Oct  8 11:21:38 COMP4108-a2 kernel: [  630.290943] openat() called for cloth.txt
Oct  8 11:21:42 COMP4108-a2 kernel: [  634.521428] openat() called for cloth.txt
```

9.

P5 Isolated Compartments - If we compartmentalize both the user space and the kernel space this can prevent the access to syscalls that allow rootkits to hide files.

P6 least privilege - Rootkits use escalation of privilege in order to gain access to root level permissions and by providing least privilege it can make it much harder for rootkits to be effective.

Section B:

1. See rootkit.c for this part specifically the new_execve() hook

```
[12953.809908] executing: /bin/sleep
[12953.809910] effective UID: 0
[12954.812735] executing: /bin/sed
[12954.812739] effective UID: 0
[12954.818622] executing: /bin/cat
[12954.818625] effective UID: 0
[12958.200032] executing: /bin/dmesg
[12958.200036] effective UID: 1001
[12958.200537] executing: /usr/bin/tail
[12958.200543] effective UID: 1001
```

2. See rootkit.c and insert.sh for this part

```
student@COMP4108-a2:~/a2$ whoami
student
student@COMP4108-a2:~/a2$ sudo ./insert.sh
UID=1001
student@COMP4108-a2:~/a2$ whoami
root
student@COMP4108-a2:~/a2$
```

Explanation: The way that this method of backdooring works is by first saving the original execve() syscall so we can call it later. Unprotect the memory in order to be able to write to the sys_call_table, Then by intercepting the execve() syscall point it to a custom execve() hook. This hook checks if the current effective uid is equal to the root_uid param passed in using insmod in the insert.sh script. If that is true it prepares new credentials using the prepare_kernel_cred() function and commits them to the user using commit_creds(). Finally we return the original execve() allowing the program to run with elevated permissions and call protect_memory() to prevent future and unintended edits.

Section C:

1. See rootkit.c for this part specifically the new_getdents64() hook

```
Oct  8 13:53:51 COMP4108-a2 kernel: [ 9763.418559] getdents64() hook invoked.
Oct  8 13:53:51 COMP4108-a2 kernel: [ 9763.418563] entry: .
Oct  8 13:53:51 COMP4108-a2 kernel: [ 9763.418566] entry: ..
Oct  8 13:53:51 COMP4108-a2 kernel: [ 9763.418569] entry: 6212
```

2. See rootkit.c and insert.sh for this part

```
student@COMP4108-a2:~/a2$ ls
'$sys$_lol_hidden.txt'   insert.sh       Module.symvers   rootkit.mod     rootkit.o
 cloth.txt               Makefile        rootkit.c        rootkit.mod.c
 eject.sh                modules.order   rootkit.ko       rootkit.mod.o
student@COMP4108-a2:~/a2$ sudo ./insert.sh
UID=1001
student@COMP4108-a2:~/a2$ ls -al
total 168
drwxrwxr-x  2 student student  4096 Oct  8 11:44 .
drwxr-xr-x 10 student student  4096 Oct  8 11:13 ..
-rw-rw-r--  1 student student     0 Oct  8 11:21 cloth.txt
-rwxrwxr-x  1 student student   107 Feb  1  2024 eject.sh
-rwxrwxr-x  1 student student   258 Oct  8 11:14 insert.sh
-rw-rw-r--  1 student student   174 Feb  1  2024 Makefile
-rw-rw-r--  1 student student    28 Oct  8 11:44 modules.order
-rw-rw-r--  1 student student     0 Oct  8 11:15 Module.symvers
-rw-rw-r--  1 student student  6395 Oct  8 11:44 rootkit.c
-rw-rw-r--  1 root    root    13312 Oct  8 11:44 rootkit.ko
-rw-rw-r--  1 student student   238 Oct  8 11:44 .rootkit.ko.cmd
-rw-rw-r--  1 student student    28 Oct  8 11:44 rootkit.mod
-rw-rw-r--  1 student student  1430 Oct  8 11:44 rootkit.mod.c
-rw-rw-r--  1 student student   112 Oct  8 11:44 .rootkit.mod.cmd
-rw-rw-r--  1 root    root     4408 Oct  8 11:44 rootkit.mod.o
-rw-rw-r--  1 student student 30946 Oct  8 11:44 .rootkit.mod.o.cmd
-rw-rw-r--  1 root    root    10224 Oct  8 11:44 rootkit.o
-rw-rw-r--  1 student student 49769 Oct  8 11:44 .rootkit.o.cmd
```

Explanation: The way that this method of hiding files works is by first saving the original getdents64() syscall so we can call it later. Unprotect the memory in order to be able to write to the sys_call_table. Then by making a copy of the dirents to a kernel buffer we are able to manipulate the dirents. Loop through all the entries and print them out while also checking if the current entry has the magic_prefix in this case 'sys'. If the first entry has the magic prefix, subtract the size of the current entry from the total and shift each entry forward by one to cover the first entry. Otherwise, change the size of the previous entry to be the size of the sum of the previous and current entry, essentially skipping over the current entry and hiding it. Finally copy the kernel buffer back to the userspace and then return the original getdents64 syscall.