

# Rootkit – by

[Meghana D - 110739940]

[Prasoon Rai - 110921754]

[Shubhada Patil - 110284246]

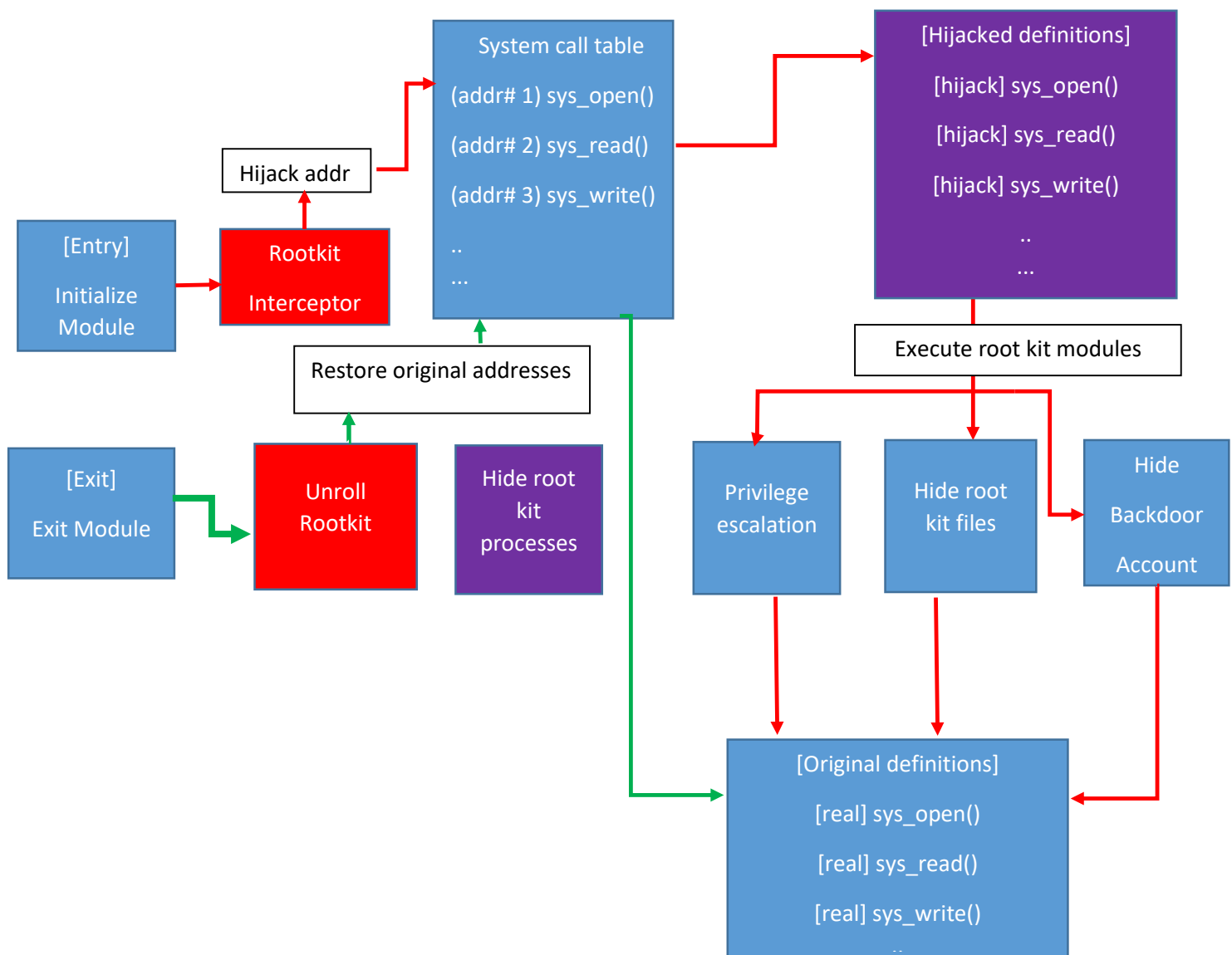
[Siddharth Kavar - 110739852]

## 1. Introduction:

Rootkit according to Wikipedia is a collection of computer software, typically malicious, that is designed to enable access to a computer or areas of its software that would not otherwise be allowed (for example, to an unauthorized user) and often masks its existence or the existence of other software. After attackers manage to gain access to remote machine, they want to keep access to the machine intact without exposing themselves. In this project, we have implemented a rootkit as a loadable kernel module which when loaded in the kernel will do the following:

- Hide the malicious process from showing up when a user does 'ps' or related command.
- Modify the /etc/passwd and /etc/shadow file to add a backdoor account and hide the backdoor entry when a user does cat.
- Give the ability to a malicious process to elevate its user id to root upon demand.
- Hide specific files and directories from showing up when a user does "ls".

## 2. Project schematic diagram:



### 3. System Specification

To use our module, you need to have Linux Kernel - 4.4.0-31-generic running on 32bit-X86 architecture.

### 4. Major Design Decisions:

As soon as we insert our module into the system we scan for the system call table address in the main memory. We do a linear scan of the memory and compare every address with the known address of the **sys\_close** system call (which is always exported). As soon as we find the match, we know that we have found the address of the system call table and then store the current function pointer of the system calls that we want to hijack and replace it with our modified function (**\*\_hijack**). We have mainly modified three system calls

- 1) ***long sys\_getdents64(unsigned int fd, struct linux\_dirent64 \_\_user \*dirent, unsigned int count)***
- 2) ***long sys\_read (unsigned int fd, char \_\_user \*buf, size\_t count)***
- 3) ***long sys\_chdir(const char \_\_user \*filename)***

Apart from this we have also modified **proc\_iterate** function for hiding malicious process when ps related commands are executed.

***int proc\_readdir\_new(struct file \*file, struct dir\_context \*ctx)***

- a) **Hiding Files and Directories:** In order to hide files, we have hijacked the **getdents64** system call. Please make a note that there are two system calls **getdents** and **getdents64**. We have hijacked **getdents64** because it was the one which was being called on 32bit Ubuntu 14.04 based on linux 4.4.0-31-generic. Here, before returning **dirent** structure back to userspace, we scan the buffer and look at each record. If any record contains the substring "rootkit" in directory name, we skip writing that record in the user struct **dirent** buffer. So, this way directory records containing substring "rootkit" are never returned to user. Therefore, such files and directories never shown up on **ls** and related commands.
- b) **Adding Backdoor:** In order to add the backdoor account, we have added the hardcoded string **"devil:x:1002:1002:devil lurking,,,:/home/devil:/bin/bash"**  
**"devil:\$1\$xyz\$jYhggMsejqNh4czc7hl8N/:17121:0:99999:7:::\n"**  
to **"/etc/passwd"** and **"/etc/shadow"** respectively. The **normal user** can see the contents of the **/etc/passwd** but the user cannot change it. Also the normal user cannot read the contents of the **/etc/shadow**. If we execute **"cat"** command on **"/etc/passwd"** the backdoor will not show up, provided that module is loaded.
- c) **Hiding malicious Processes:** In order to hide processes, we have hacked the **/proc** filesystem. We have modified **procfs** filesystem's **iterate** function to use our helper function **filldir**. This **filldir** function is responsible for filling the directory information in the user **dirent** buffer that is provided when the call is made to **getdents** system call. We skip records of our malicious

processes while filling user's dirent buffer. We identify our malicious processes/executables which contains the keyword "hack" in their name. We could have simply reused already modified **getdents** system call but that increases the execution time of the ps command and it is very noticeable. So, in order to not prolong the execution time of the ps command we chose to modify the procfs filesystems iterate method.

***int proc\_iterate(struct file \*file, struct dir\_context \*ctx)***

d) **Privilege escalation:** In privilege escalation, we modified the **chdir** system call.

***long sys\_chdir(const char \_\_user \*filename)***

To elevate to root privilege, process makes a call to this system call by passing "abrt" string as filename and upon successful execution it will become root.

Finally, when we remove our module, we restore all system calls to their original state.

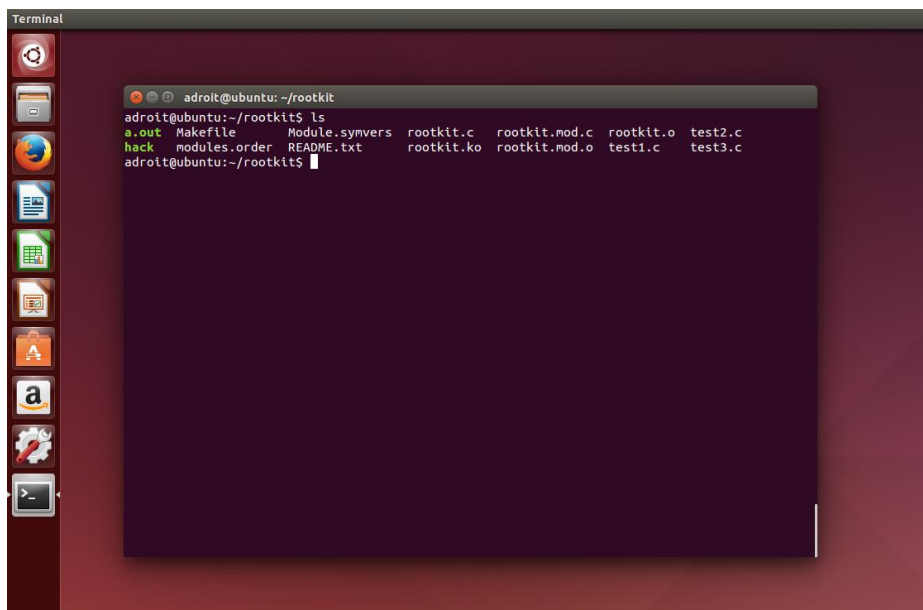
## 5. Module wise summary and execution outputs:

### 5.1 Hide root kit files and directories:

- This module is responsible for hiding files installed by the attacker as part of rootkit.
- Once active, files and folders with substring 'rootkit' do not list upon executing 'ls' command.

#### 5.1.1 Execution outputs:

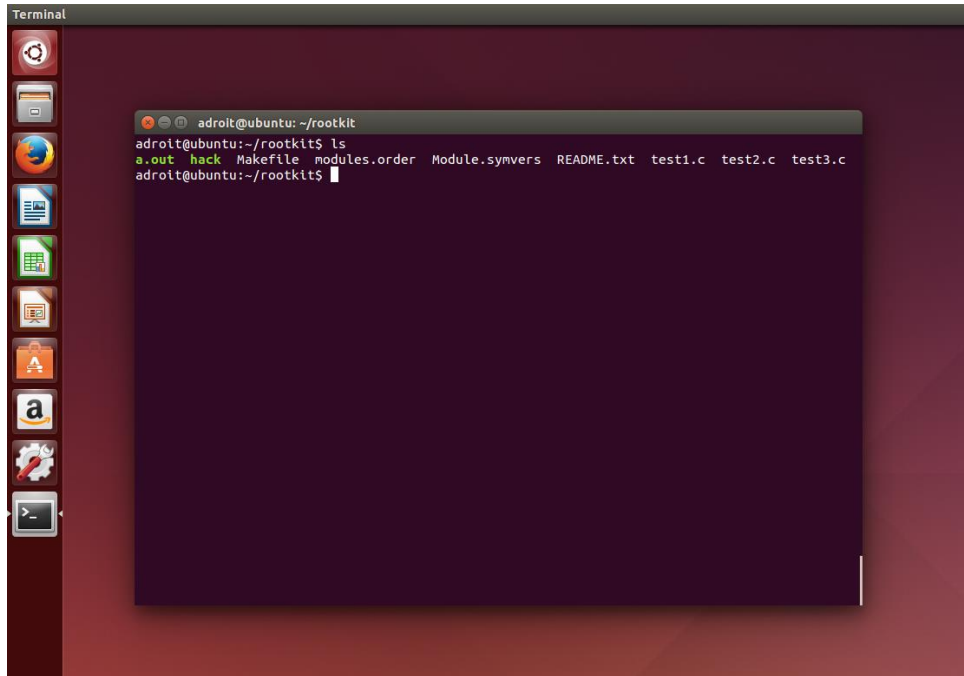
##### 1. Before loading rootkit, all the files are visible:



A terminal window titled "Terminal" is shown. The prompt is "adroit@ubuntu: ~/rootkit". The user has entered "ls" and the output is displayed in two lines. The first line shows "a.out", "Makefile", "Module.symvers", "rootkit.c", "rootkit.mod.c", "rootkit.o", and "test2.c". The second line shows "hack", "modules.order", "README.txt", "rootkit.ko", "rootkit.mod.o", "test1.c", and "test3.c". The prompt returns to "adroit@ubuntu: ~/rootkit\$".

```
adroit@ubuntu: ~/rootkit
adroit@ubuntu:~/rootkit$ ls
a.out  Makefile      Module.symvers  rootkit.c  rootkit.mod.c  rootkit.o  test2.c
hack   modules.order  README.txt      rootkit.ko  rootkit.mod.o  test1.c   test3.c
adroit@ubuntu:~/rootkit$
```

## 2. After installing rootkit module, files containing substring “rootkit” don’t list upon ls:



A terminal window titled 'Terminal' with a dark background and a vertical sidebar of application icons on the left. The terminal shows a user 'adroit' at 'ubuntu' in the directory '~/rootkit'. The command 'ls' has been executed, and the output lists several files: 'a.out', 'hack', 'Makefile', 'modules.order', 'Module.symvers', 'README.txt', 'test1.c', 'test2.c', and 'test3.c'. The prompt 'adroit@ubuntu:~/rootkit\$' is visible at the bottom of the terminal window.

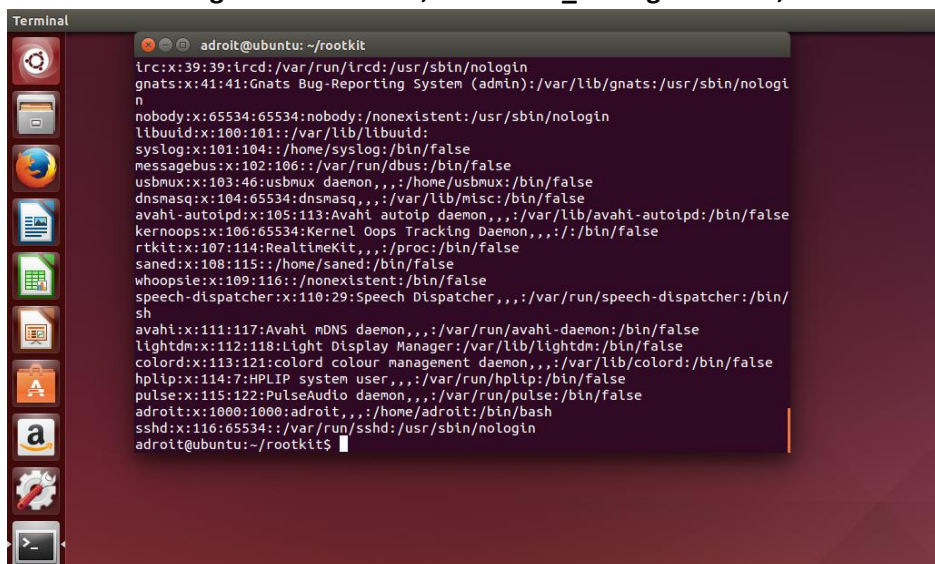
```
adroit@ubuntu:~/rootkit$ ls
a.out  hack  Makefile  modules.order  Module.symvers  README.txt  test1.c  test2.c  test3.c
adroit@ubuntu:~/rootkit$
```

## 5.2 Hide back door account:

- This module is responsible to hide the password entry of the attacker.
- Once active, /etc/passwd does not show attacker’s entry upon executing ‘cat’ command.

### 5.2.1 Execution outputs:

#### 1. After installing rootkit module, user devil\_lurking is added, but not visible on ‘cat /etc/passwd’:



A terminal window titled 'Terminal' with a dark background and a vertical sidebar of application icons on the left. The terminal shows a user 'adroit' at 'ubuntu' in the directory '~/rootkit'. The command 'cat /etc/passwd' has been executed, and the output lists the contents of the /etc/passwd file. The user 'devil\_lurking' is not visible in the output. The prompt 'adroit@ubuntu:~/rootkit\$' is visible at the bottom of the terminal window.

```
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101:/var/lib/libuuid:
syslog:x:101:104:/home/syslog:/bin/false
messagebus:x:102:106:/var/run/dbus:/bin/false
usbmux:x:103:46:usbmux daemon,,,:/home/usbmux:/bin/false
dnsmasq:x:104:65534:dnsmasq,,,:/var/lib/mtsc:/bin/false
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
kernoops:x:106:65534:Kernel Oops Tracking Daemon,,,:/bin/false
rtkit:x:107:114:RealtimeKit,,,:/proc:/bin/false
saned:x:108:115:/home/saned:/bin/false
whoopsie:x:109:116:/nonexistent:/bin/false
speech-dispatcher:x:110:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh
avahi:x:111:117:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
lightdm:x:112:118:Light Display Manager:/var/lib/lightdm:/bin/false
colord:x:113:121:colord colour management daemon,,,:/var/lib/colord:/bin/false
hplip:x:114:7:HPLIP system user,,,:/var/run/hplip:/bin/false
pulse:x:115:122:PulseAudio daemon,,,:/var/run/pulse:/bin/false
adroit:x:1000:1000:adroit,,,:/home/adroit:/bin/bash
sshd:x:116:65534:/var/run/sshd:/usr/sbin/nologin
adroit@ubuntu:~/rootkit$
```

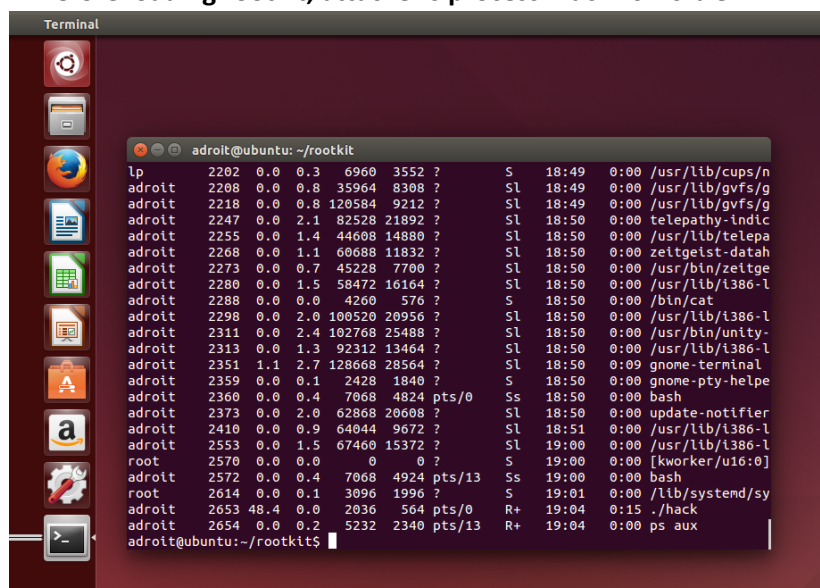
### 5.3 Hide root kit processes:

- This module is responsible for hiding attacker's processes.
- Upon execution, 'ps' command does not list specific processes when module is loaded.
- If there are child processes spawned by attacker processes, they get hidden as well.

#### 5.3.1 Execution outputs:

Experiment 1: Test file : test3.c executed as "gcc test3.c -o hack", "./hack"

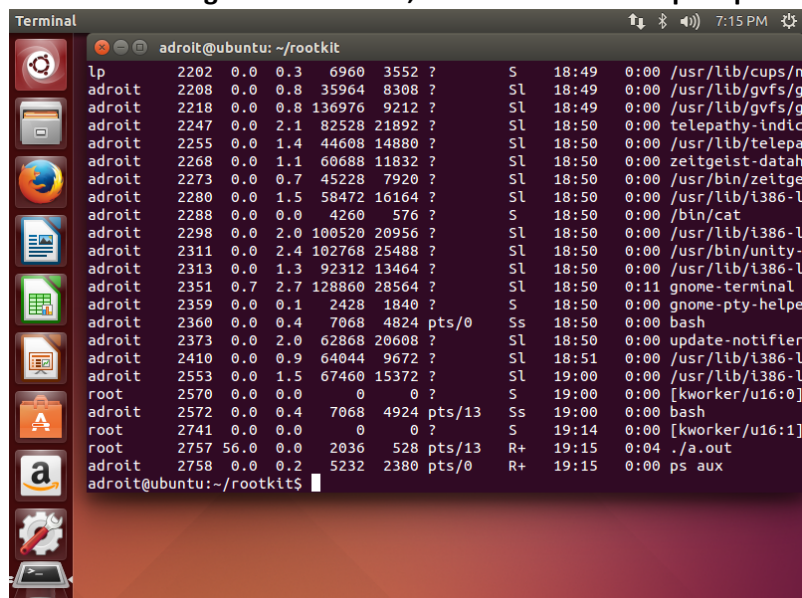
##### 1. Before loading rootkit, attacker's process 'hack' is visible:



A terminal window titled 'Terminal' showing the output of the 'ps' command. The output lists various system processes, including 'lp', 'adroit', 'root', and 'ps aux'. The 'hack' process is visible in the list, indicating it is not yet hidden by the rootkit.

```
adroit@ubuntu: ~/rootkit
lp      2202  0.0  0.3  6960  3552 ?        S    18:49  0:00 /usr/lib/cups/n
adroit  2208  0.0  0.8  35964  8308 ?        SL   18:49  0:00 /usr/lib/gvfs/g
adroit  2218  0.0  0.8  120584  9212 ?        SL   18:49  0:00 /usr/lib/gvfs/g
adroit  2247  0.0  2.1  82528  21892 ?        SL   18:50  0:00 telepathy-indic
adroit  2255  0.0  1.4  44608  14880 ?        SL   18:50  0:00 /usr/lib/telepa
adroit  2268  0.0  1.1  60688  11832 ?        SL   18:50  0:00 zeitgeist-datah
adroit  2273  0.0  0.7  45228  7700 ?        SL   18:50  0:00 /usr/bin/zeitge
adroit  2280  0.0  1.5  58472  16164 ?        SL   18:50  0:00 /usr/lib/i386-l
adroit  2288  0.0  0.0  4260  576 ?        S    18:50  0:00 /bin/cat
adroit  2298  0.0  2.0  100520  20956 ?        SL   18:50  0:00 /usr/lib/i386-l
adroit  2311  0.0  2.4  102768  25488 ?        SL   18:50  0:00 /usr/bin/unity-
adroit  2313  0.0  1.3  92312  13464 ?        SL   18:50  0:00 /usr/lib/i386-l
adroit  2351  1.1  2.7  128668  28564 ?        SL   18:50  0:09 gnome-terminal
adroit  2359  0.0  0.1  2428  1840 ?        S    18:50  0:00 gnome-pty-helpe
adroit  2360  0.0  0.4  7068  4824 pts/0    Ss   18:50  0:00 bash
adroit  2373  0.0  2.0  62868  20608 ?        SL   18:50  0:00 update-notifier
adroit  2410  0.0  0.9  64044  9672 ?        SL   18:51  0:00 /usr/lib/i386-l
adroit  2553  0.0  1.5  67460  15372 ?        SL   19:00  0:00 /usr/lib/i386-l
root    2570  0.0  0.0  0  0 ?        S    19:00  0:00 [kworker/u16:0]
adroit  2572  0.0  0.4  7068  4924 pts/13   Ss   19:00  0:00 bash
root    2614  0.0  0.1  3096  1996 ?        S    19:01  0:00 /lib/systemd/sy
adroit  2653  48.4  0.0  2036  564 pts/0    R+   19:04  0:15 ./hack
adroit  2654  0.0  0.2  5232  2340 pts/13   R+   19:04  0:00 ps aux
```

##### 2. After installing rootkit module, 'hack' does not list upon 'ps' command:



A terminal window titled 'Terminal' showing the output of the 'ps' command after the rootkit module has been installed. The output lists various system processes, but the 'hack' process is no longer visible, indicating it has been successfully hidden by the rootkit.

```
adroit@ubuntu: ~/rootkit
lp      2202  0.0  0.3  6960  3552 ?        S    18:49  0:00 /usr/lib/cups/n
adroit  2208  0.0  0.8  35964  8308 ?        SL   18:49  0:00 /usr/lib/gvfs/g
adroit  2218  0.0  0.8  136976  9212 ?        SL   18:49  0:00 /usr/lib/gvfs/g
adroit  2247  0.0  2.1  82528  21892 ?        SL   18:50  0:00 telepathy-indic
adroit  2255  0.0  1.4  44608  14880 ?        SL   18:50  0:00 /usr/lib/telepa
adroit  2268  0.0  1.1  60688  11832 ?        SL   18:50  0:00 zeitgeist-datah
adroit  2273  0.0  0.7  45228  7920 ?        SL   18:50  0:00 /usr/bin/zeitge
adroit  2280  0.0  1.5  58472  16164 ?        SL   18:50  0:00 /usr/lib/i386-l
adroit  2288  0.0  0.0  4260  576 ?        S    18:50  0:00 /bin/cat
adroit  2298  0.0  2.0  100520  20956 ?        SL   18:50  0:00 /usr/lib/i386-l
adroit  2311  0.0  2.4  102768  25488 ?        SL   18:50  0:00 /usr/bin/unity-
adroit  2313  0.0  1.3  92312  13464 ?        SL   18:50  0:00 /usr/lib/i386-l
adroit  2351  0.7  2.7  128860  28564 ?        SL   18:50  0:11 gnome-terminal
adroit  2359  0.0  0.1  2428  1840 ?        S    18:50  0:00 gnome-pty-helpe
adroit  2360  0.0  0.4  7068  4824 pts/0    Ss   18:50  0:00 bash
adroit  2373  0.0  2.0  62868  20608 ?        SL   18:50  0:00 update-notifier
adroit  2410  0.0  0.9  64044  9672 ?        SL   18:51  0:00 /usr/lib/i386-l
adroit  2553  0.0  1.5  67460  15372 ?        SL   19:00  0:00 /usr/lib/i386-l
root    2570  0.0  0.0  0  0 ?        S    19:00  0:00 [kworker/u16:0]
adroit  2572  0.0  0.4  7068  4924 pts/13   Ss   19:00  0:00 bash
root    2741  0.0  0.0  0  0 ?        S    19:14  0:00 [kworker/u16:1]
root    2757  56.0  0.0  2036  528 pts/13   R+   19:15  0:04 ./a.out
adroit  2758  0.0  0.2  5232  2380 pts/0    R+   19:15  0:00 ps aux
```

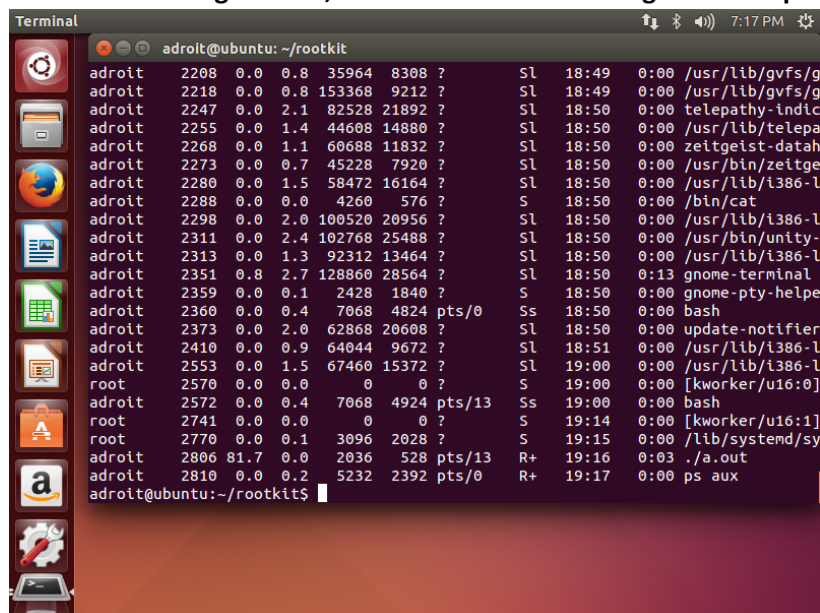
## 5.4 Privilege escalation:

- This module is responsible to enable the attacker acquire root privilege.
- Once active, the attacker would be able to gain root access.

### 5.4.1 Execution outputs:

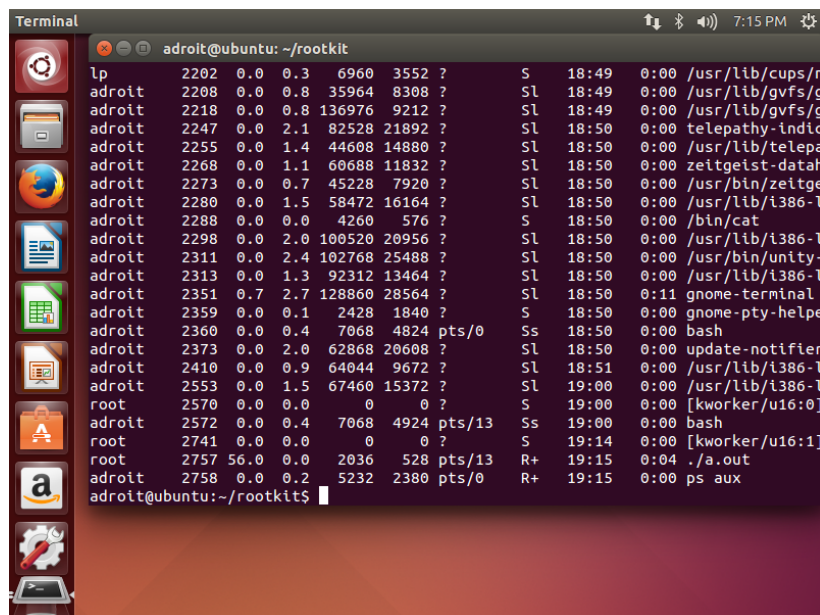
Experiment 1: Test File : test1.c compiled as “gcc test1.c” and executed with “./a.out”

1. Before loading rootkit, user is listed as running user for process ./a.out ('adroit' in our case):

A terminal window titled 'Terminal' with the prompt 'adroit@ubuntu: ~/rootkit'. It displays the output of the 'ps aux' command, showing a list of running processes. The process for './a.out' is listed with user 'adroit' and PID 2810.

```
Terminal
adroit@ubuntu: ~/rootkit
adroit 2208 0.0 0.8 35964 8308 ? SL 18:49 0:00 /usr/lib/gvfs/g
adroit 2218 0.0 0.8 153368 9212 ? SL 18:49 0:00 /usr/lib/gvfs/g
adroit 2247 0.0 2.1 82528 21892 ? SL 18:50 0:00 telepathy-indic
adroit 2255 0.0 1.4 44608 14880 ? SL 18:50 0:00 /usr/lib/telepa
adroit 2268 0.0 1.1 60688 11832 ? SL 18:50 0:00 zeitgeist-datah
adroit 2273 0.0 0.7 45228 7920 ? SL 18:50 0:00 /usr/bin/zeitge
adroit 2280 0.0 1.5 58472 16164 ? SL 18:50 0:00 /usr/lib/i386-l
adroit 2288 0.0 0.0 4260 576 ? S 18:50 0:00 /bin/cat
adroit 2298 0.0 2.0 100520 20956 ? SL 18:50 0:00 /usr/lib/i386-l
adroit 2311 0.0 2.4 102768 25488 ? SL 18:50 0:00 /usr/bin/unity-
adroit 2313 0.0 1.3 92312 13464 ? SL 18:50 0:00 /usr/lib/i386-l
adroit 2351 0.8 2.7 128860 28564 ? SL 18:50 0:13 gnome-terminal
adroit 2359 0.0 0.1 2428 1840 ? S 18:50 0:00 gnome-pty-helpe
adroit 2360 0.0 0.4 7068 4824 pts/0 Ss 18:50 0:00 bash
adroit 2373 0.0 2.0 62868 20608 ? SL 18:50 0:00 update-notifier
adroit 2410 0.0 0.9 64044 9672 ? SL 18:51 0:00 /usr/lib/i386-l
adroit 2553 0.0 1.5 67460 15372 ? SL 19:00 0:00 /usr/lib/i386-l
root 2570 0.0 0.0 0 0 ? S 19:00 0:00 [kworker/u16:0]
adroit 2572 0.0 0.4 7068 4924 pts/13 Ss 19:00 0:00 bash
root 2741 0.0 0.0 0 0 ? S 19:14 0:00 [kworker/u16:1]
root 2770 0.0 0.1 3096 2028 ? S 19:15 0:00 /lib/systemd/sy
adroit 2806 81.7 0.0 2036 528 pts/13 R+ 19:16 0:03 ./a.out
adroit 2810 0.0 0.2 5232 2392 pts/0 R+ 19:17 0:00 ps aux
adroit@ubuntu:~/rootkit$
```

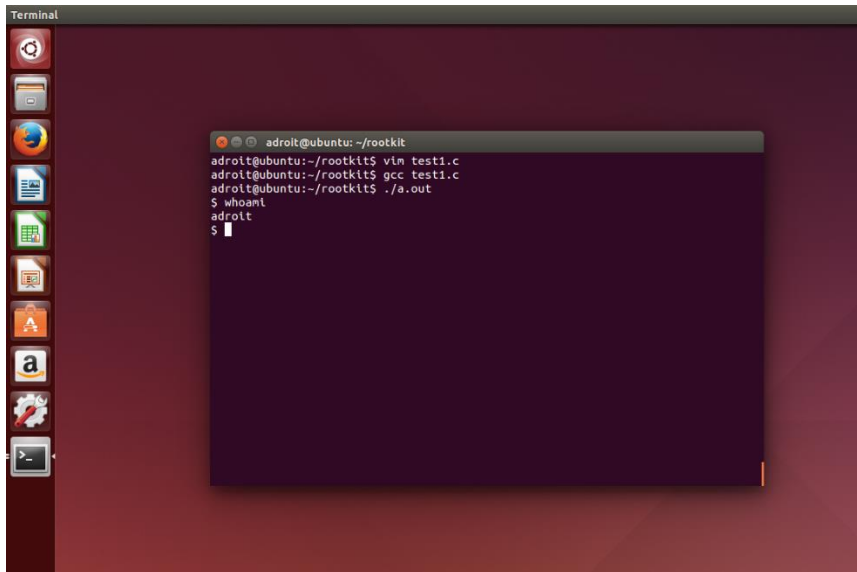
2. After loading rootkit module, user name lists as 'root' for process ./a.out:

A terminal window titled 'Terminal' with the prompt 'adroit@ubuntu: ~/rootkit'. It displays the output of the 'ps aux' command after the rootkit module has been loaded. The process for './a.out' now shows the user as 'root' with PID 2758.

```
Terminal
adroit@ubuntu: ~/rootkit
lp 2202 0.0 0.3 6960 3552 ? S 18:49 0:00 /usr/lib/cups/n
adroit 2208 0.0 0.8 35964 8308 ? SL 18:49 0:00 /usr/lib/gvfs/g
adroit 2218 0.0 0.8 136976 9212 ? SL 18:49 0:00 /usr/lib/gvfs/g
adroit 2247 0.0 2.1 82528 21892 ? SL 18:50 0:00 telepathy-indic
adroit 2255 0.0 1.4 44608 14880 ? SL 18:50 0:00 /usr/lib/telepa
adroit 2268 0.0 1.1 60688 11832 ? SL 18:50 0:00 zeitgeist-datah
adroit 2273 0.0 0.7 45228 7920 ? SL 18:50 0:00 /usr/bin/zeitge
adroit 2280 0.0 1.5 58472 16164 ? SL 18:50 0:00 /usr/lib/i386-l
adroit 2288 0.0 0.0 4260 576 ? S 18:50 0:00 /bin/cat
adroit 2298 0.0 2.0 100520 20956 ? SL 18:50 0:00 /usr/lib/i386-l
adroit 2311 0.0 2.4 102768 25488 ? SL 18:50 0:00 /usr/bin/unity-
adroit 2313 0.0 1.3 92312 13464 ? SL 18:50 0:00 /usr/lib/i386-l
adroit 2351 0.7 2.7 128860 28564 ? SL 18:50 0:11 gnome-terminal
adroit 2359 0.0 0.1 2428 1840 ? S 18:50 0:00 gnome-pty-helpe
adroit 2360 0.0 0.4 7068 4824 pts/0 Ss 18:50 0:00 bash
adroit 2373 0.0 2.0 62868 20608 ? SL 18:50 0:00 update-notifier
adroit 2410 0.0 0.9 64044 9672 ? SL 18:51 0:00 /usr/lib/i386-l
adroit 2553 0.0 1.5 67460 15372 ? SL 19:00 0:00 /usr/lib/i386-l
root 2570 0.0 0.0 0 0 ? S 19:00 0:00 [kworker/u16:0]
adroit 2572 0.0 0.4 7068 4924 pts/13 Ss 19:00 0:00 bash
root 2741 0.0 0.0 0 0 ? S 19:14 0:00 [kworker/u16:1]
root 2757 56.0 0.0 2036 528 pts/13 R+ 19:15 0:04 ./a.out
adroit 2758 0.0 0.2 5232 2380 pts/0 R+ 19:15 0:00 ps aux
adroit@ubuntu:~/rootkit$
```

## Experiment 2:

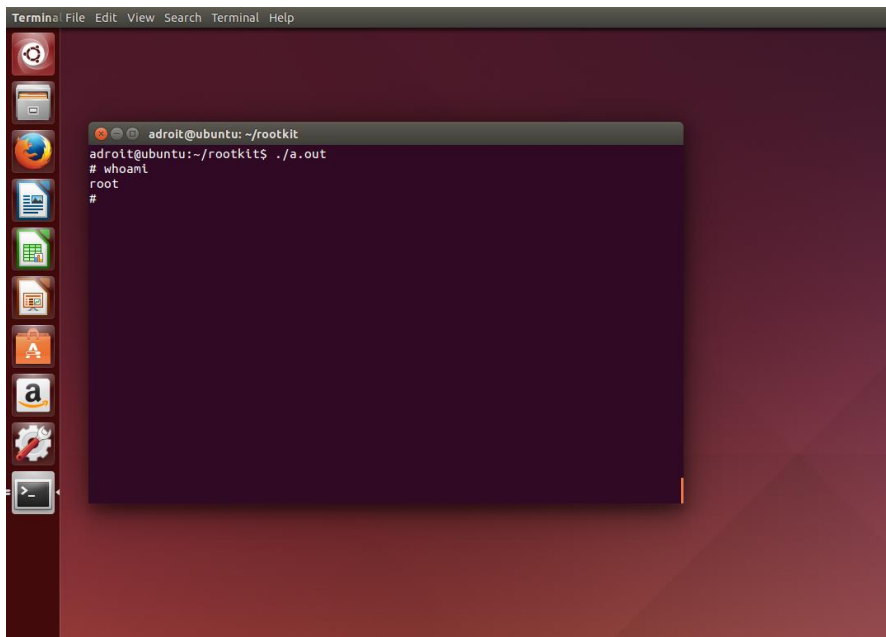
1. If rootkit is not loaded, executing test4.c does not escalate privilege ('whoami' outputs the running user):



A terminal window titled 'Terminal' is open on a Ubuntu desktop. The terminal shows the following commands and output:

```
adroit@ubuntu: ~/rootkit
adroit@ubuntu:~/rootkit$ vim test1.c
adroit@ubuntu:~/rootkit$ gcc test1.c
adroit@ubuntu:~/rootkit$ ./a.out
$ whoami
adroit
$
```

2. After loading rootkit and executing test4.c, privilege is escalated to root.



A terminal window titled 'Terminal' is open on a Ubuntu desktop. The terminal shows the following commands and output:

```
adroit@ubuntu: ~/rootkit
adroit@ubuntu:~/rootkit$ ./a.out
# whoami
root
#
```

## 6. Limitations

a) While removing our module, one needs to remove the backdoor account manually, which can be done easily by executing “sudo userdel -f devil”.

b) After removing our module, closing the terminal results in minor freezing of the window. We are not certain about the root cause and are exploring possible solutions.

## 7. References

- <http://www.cbs.dtu.dk/cgi-bin/nph-runsafe?man=getdents64>
- [https://kernelnewbies.org/Documents/Kernel-Docbooks?action=AttachFile&do=get&target=procfs-guide\\_2.6.29.pdf](https://kernelnewbies.org/Documents/Kernel-Docbooks?action=AttachFile&do=get&target=procfs-guide_2.6.29.pdf)
- <https://linux.die.net/lkmpg/x710.html>
- <https://www.kernel.org/doc/Documentation/security/credentials.txt>
- The Linux Kernel Module Programming Guide
- <http://phrack.org/issues/58/6.html>
- <http://turbochaos.blogspot.com/2013/09/linux-rootkits-101-1-of-3.html>
- <http://stackoverflow.com/questions/26000691/system-call-interception-via-loadable-kernel-module>