Carsten Hood
UIN: 922009787
CSCE 465-500
Due 2017/9/21

HW #1: Linux Capability Exploration Lab

Report

I ran Ubuntu through VirtualBox VM on a MacBook Pro. To note one thing I found interesting, I appreciated the fluidity/speed of this "virtual machine" running alongside my usual computer software; there were never hiccups or muddled user interaction.

==========

Question 1

In the first line we change *passwd* to a non-Set-UID program. Now the seed user can't execute *passwd*. (It doesn't throw an error until a new password is entered, which made me assume it was working at first). Then we assign *passwd* three capabilities to re-enable it: (1) *cap_dac_override* allows for bypassing file permission checks; (2) *cap_chown* lets the process make changes to files' UIDs and GIDs; and (3) *cap_fowner* grants access to inode flags, ACLs, and operations' permission checks when operations usually require matching UIDs. These capabilities allow the *passwd* process to access and modify the files */etc/passwd* and */etc/shadow* and their metadata in order to execute correctly for the seed user.

```
[09/21/2017 11:33] root@ubuntu:/home/seed# chmod u-s /usr/bin/passwd
[09/21/2017 11:34] root@ubuntu:/home/seed# setcap cap_dac_override,cap_c
hown,cap_fowner=+ep /usr/bin/passwd
[09/21/2017 11:34] root@ubuntu:/home/seed# getcap /usr/bin/passwd
/usr/bin/passwd = cap_chown,cap_dac_override,cap_fowner+ep
```

The outcome is that the seed user can successfully change their password (shown in the next screenshot) when *passwd* has these three capabilities (shown on the last line):

```
[09/21/2017 10:10] seed@ubuntu:~/Desktop/libcap2.22/libcap-2.22/libcap$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
[09/21/2017 10:11] seed@ubuntu:~/Desktop/libcap2.22/libcap-2.22/libcap$ getcap /usr/bin/
passwd
/usr/bin/passwd = cap_chown,cap_dac_override,cap_fowner+ep
```

==========

Question 2

(a) CAP DAC READ SEARCH

(1) The purpose of this capability is to let processes execute permission checks and bypass file/directory read permission checks.

(2) To demonstrate the functionality of *cap_dac_read_search* we execute the simple file-reading program *cat* on a test file *text.txt*. As shown below, we first remove all read/write permissions from the test file so that calling *cat* on *text.txt* as the seed user fails. After granting *cat* the capability *cap_dac_read_search* (line 6), *cat* is now permitted to read and successfully display *text.txt*'s content ("This file has no…").

```
[09/21/2017 13:23] root@ubuntu:/home/seed/Desktop/hw1# ls -l text.txt
----rw-r--. 1 seed seed 107 Sep 21 13:10 text.txt
[09/21/2017 13:23] root@ubuntu:/home/seed/Desktop/hw1# su seed
[09/21/2017 13:24] seed@ubuntu:~/Desktop/hw1$ cat text.txt
cat: text.txt: Permission denied
[09/21/2017 13:24] seed@ubuntu:~/Desktop/hw1$ sudo setcap cap_dac_read_search=ep /bin/cat
[sudo] password for seed:
Sorry, try again.
[sudo] password for seed:
[09/21/2017 13:25] seed@ubuntu:~/Desktop/hw1$ cat text.txt
This file has no read/write permissions and can be used to demonstrate the cap_dac_read_search
capability.
[09/21/2017 13:25] seed@ubuntu:~/Desktop/hw1$
```

(b) CAP DAC OVERRIDE

(1) This capability controls processes' "discretionary access control", i.e., their ability to read, write, and execute files.

(2) We demonstrate *cap_dac_override* with the *passwd* command. Initially *passwd* possesses the capability (shown in lines 1-2) and can be used successfully (lines 3-8). We then reset *passwd*'s capabilities minus *cap_dac_override* (line 9) and see that execution now fails (lines 10-17). This is because *passwd* is not authorized to read and write to the files */etc/passwd* and */etc/shadow* without *cap_dac_override* and hence cannot function correctly.

```
[09/21/2017 17:39] seed@ubuntu:~$ getcap /usr/bin/passwd
/usr/bin/passwd = cap_chown,cap_dac_override,cap_fowner+ep
[09/21/2017 17:39] seed@ubuntu:~$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
[09/21/2017 17:39] seed@ubuntu:~$ sudo setcap cap_chown,cap_fowner+ep /usr/bin/passwd
[sudo] password for seed:
[09/21/2017 17:40] seed@ubuntu:~$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: Authentication token manipulation error
passwd: password unchanged
```

(c) CAP FOWNER

(1) The purpose of this capability is to control access of processes to files whose UIDs they don't match when normally this access would be prohibited, as well as access to arbitrary files' metadata, ACLs, and inode flags.

(2) To demonstrate the workings of *cap_fowner* we show that the program *passwd* fails to execute without *cap_fowner* (lines 1-7) since it can't properly access */etc/passwd* and */etc/*shadow, but then succeeds after *cap_fowner* is added in line 8, granting full access.

```
[09/21/2017 17:57] seed@ubuntu:~$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: Authentication token manipulation error
passwd: password unchanged
[09/21/2017 17:57] seed@ubuntu:~$ sudo setcap cap_chown,cap_dac_override,cap_fowner+ep
/usr/bin/passwd
[09/21/2017 17:58] seed@ubuntu:~$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
```

(d) CAP CHOWN

(1) This capability controls allowing for making changes to files' GIDs & UIDs.

(2) To demonstrate *cap_chown*, we first attempt *chown* on a test file *text.txt* as the root user and are rejected. However, after granting *chown* the capability *cap_chown* (with the command *sudo setcap cap_chown=+ep /bin/chown*) the same operation runs successfully (in the second to last line below).

```
[09/21/2017 15:38] seed@ubuntu:~/Desktop/hw1$ ll text.txt
----rw-r--. 1 seed seed 107 Sep 21 13:10 text.txt
[09/21/2017 15:38] seed@ubuntu:~/Desktop/hw1$ chown root /home/seed/Desktop/hw1/text.txt
chown: changing ownership of `/home/seed/Desktop/hw1/text.txt': Operation not permitted
[09/21/2017 15:39] seed@ubuntu:~/Desktop/hw1$ sudo setcap cap_chown=+ep /bin/chown
[09/21/2017 15:39] seed@ubuntu:~/Desktop/hw1$ chown root /home/seed/Desktop/hw1/text.txt
[09/21/2017 15:39] seed@ubuntu:~/Desktop/hw1$ ll text.txt
```

(e) CAP FSETID

The purpose of *cap_fsetid* is to control processes' ability to abstain from clearing files' set-user-ID and set-group-ID mode bits upon modification and to modify the set-group-ID bit for files whose GIDs don't match.

```
[09/21/2017 15:38] seed@ubuntu:~/Desktop/hw1$ ll text.txt
----rw-r--. 1 seed seed 107 Sep 21 13:10 text.txt
[09/21/2017 15:38] seed@ubuntu:~/Desktop/hw1$ chown root /home/seed/Desktop/hw1/text.txt
chown: changing ownership of `/home/seed/Desktop/hw1/text.txt': Operation not permitted
[09/21/2017 15:39] seed@ubuntu:~/Desktop/hw1$ sudo setcap cap_chown=+ep /bin/chown
[09/21/2017 15:39] seed@ubuntu:~/Desktop/hw1$ chown root /home/seed/Desktop/hw1/text.txt
[09/21/2017 15:39] seed@ubuntu:~/Desktop/hw1$ ll text.txt
```

(f) CAP SYS MODULE

(1) The purpose of this capability is to control the ability of processes to load/unload kernel modules.

(2) First we run *lsmod* in the command line to list kernel modules and identify one to remove; we select *rfcomm*:

```
mac_hid                13078  0
rfcomm                 38104  0
bnep                   17791  2
```

Removal via the command *rmmod* as the seed user is denied (lines 1-2, "Operation not permitted") until we grant *rmmod* the capability *cap_sys_module* (lines 3-4), as shown below:

```
[09/21/2017 15:27] seed@ubuntu:~/Desktop/hw1$ rmmod rfcomm
ERROR: Removing 'rfcomm': Operation not permitted
[09/21/2017 15:28] seed@ubuntu:~/Desktop/hw1$ sudo setcap cap_sys_module=+ep /sbin/rmmod
[09/21/2017 15:28] seed@ubuntu:~/Desktop/hw1$ rmmod rfcomm
```

Running *lsmod* again, we see *rfcomm* is absent (it no longer appears between *mac_hid* and *bnep*.

```
mac_hid                13078  0
bnep                   17791  2
```

(g) CAP KILL

(1) This capability allows processes to send signals to other processes when otherwise this would be prohibited.

(2) To demonstrate *cap_kill*'s functionality we created a simple .c program *killer* that kills a program (in this case a simple infinite loop program) given its process ID through a command line argument.

Here's the *killer* code:

```c
int main(int argc, const char * argv[]) {
    int pid = atoi(argv[1]);
    printf("killer initiated\n");
    printf("killing: %d\n", pid);
    kill(pid, SIGKILL);
    return 0;
}
```

We also create a small program *loop* that simply runs an infinite loop until it gets signaled/killed.

```c
int main(void) {
        printf("infinite loop program running\n");
        while (1) { }
        printf("infinite loop program terminated\n");
}
```

We execute *loop* as the seed user and then execute *ps au* as user1 (a new user) to identify *loop*'s process ID as 13350 (first line below). We can try to execute "*./killer 13350*" as user1 (line 3), but currently user1 isn't authorized to signal a seed user program and so seed's *loop* process is unaffected. We then grant *killer* the capability *cap_kill* (in line 6) and attempt "*./killer 135350*" once more, with success:

```
seed     13350 95.0  0.0   2016   276 pts/0    R+   19:54   0:44 ./loop
user1    13358  0.0  0.1   4948  1152 pts/3    R+   19:55   0:00 ps au
user1@ubuntu:/home/seed/Desktop/hw1$ ./killer 13350
killer initiated
killing: 13350
user1@ubuntu:/home/seed/Desktop/hw1$ sudo setcap cap_kill=+ep ./killer
user1@ubuntu:/home/seed/Desktop/hw1$ getcap ./killer
./killer = cap_kill+ep
user1@ubuntu:/home/seed/Desktop/hw1$ ./killer 13350
killer initiated
killing: 13350
```

Because *killer* now has the capability to signal other users' processes, seed's *loop* program is cancelled and displays the text "Killed".

```
[09/21/2017 19:53] seed@ubuntu:~/Desktop/hw1$ ./loop
infinite loop program running
Killed
```

(h) CAP NET ADMIN

(1) The purpose of this capability is to control processes' ability to perform a host of network-related operations.
(2) We'll use the program *dumpcap* to demonstrate *cap_net_admin*. First run *dumpcap* as the seed user successfully (lines 1-4). Then remove *dumpcap*'s capabilities (including *cap_net_admin*) and attempt to run again (lines 5-7). Now *dumpcap* won't run since it lacks the ability to connect to network interfaces.

```
[09/21/2017 14:43] seed@ubuntu:~/Desktop/hw1$ dumpcap
File: /tmp/wireshark_eth13_20170921165938_2byr8C
Packets captured: 0
Packets received/dropped on interface eth13: 0/0
[09/21/2017 16:59] seed@ubuntu:~/Desktop/hw1$ sudo setcap -r /usr/bin/dumpcap
[sudo] password for seed:
[09/21/2017 17:01] seed@ubuntu:~/Desktop/hw1$ dumpcap
dumpcap: There are no interfaces on which a capture can be done
```

(i) CAP NET RAW

(1) The purpose of this capability is to control use of RAW/PACKET sockets and processes' ability to bind to addresses for "transparent proxying".
(2) To demonstrate *cap_net_raw* we first set *ping* to a non-Set-UID program (line 1) so that the seed user can't execute *ping* (lines 2-3). However, after granting *ping* the capability *cap_net* (line 4), which allows *ping* to control sockets, we are able to execute successfully (lines 5-7).

```
[09/21/2017 15:22] seed@ubuntu:~/Desktop/hw1$ sudo chmod u-s /bin/ping
[09/21/2017 15:22] seed@ubuntu:~/Desktop/hw1$ ping www.google.com
ping: icmp open socket: Operation not permitted
[09/21/2017 15:22] seed@ubuntu:~/Desktop/hw1$ sudo setcap cap_net_raw=+ep /bin/ping
[09/21/2017 15:23] seed@ubuntu:~/Desktop/hw1$ ping www.google.com
PING www.google.com (64.233.168.106) 56(84) bytes of data.
64 bytes from oj-in-f106.1e100.net (64.233.168.106): icmp_req=1 ttl=63 time=22.5 ms
```

(j) CAP SYS NICE

(1) The capability *cap_sys_nice* gives processes special privileges to modify processes' nice values, real-time scheduling policies, CPU affinities, scheduling classes, and migrate_pages/move_pages functionality.

```
[09/21/2017 14:43] seed@ubuntu:~/Desktop/hw1$ dumpcap
File: /tmp/wireshark_eth13_20170921165938_2byr8C
Packets captured: 0
Packets received/dropped on interface eth13: 0/0
[09/21/2017 16:59] seed@ubuntu:~/Desktop/hw1$ sudo setcap -r /usr/bin/dumpcap
[sudo] password for seed:
[09/21/2017 17:01] seed@ubuntu:~/Desktop/hw1$ dumpcap
dumpcap: There are no interfaces on which a capture can be done
```

(k) CAP SYS TIME

(1) This purpose of this capability is to control processes' ability to set the system/hardware clocks.
(2) To demonstrate *cap_sys_time*, we show the date (lines 1-2) and then show that the seed user can't modify the date using the *date* command (lines 3-5). We then assign *date* the capability *cap_sys_time* (line 6) and are able to change the date successfully (final lines).

```
[09/21/2017 15:16] seed@ubuntu:~/Desktop/hw1$ date
Thu Sep 21 15:16:07 PDT 2017
[09/21/2017 15:16] seed@ubuntu:~/Desktop/hw1$ date 091209121994
date: cannot set date: Operation not permitted
Mon Sep 12 09:12:00 PDT 1994
[09/21/2017 15:17] seed@ubuntu:~/Desktop/hw1$ sudo setcap cap_sys_time+ep /bin/d
ate
[sudo] password for seed:
[09/21/2017 15:18] seed@ubuntu:~/Desktop/hw1$ date 091209121994
Mon Sep 12 09:12:00 PDT 1994
```

==========

Question 3

First we add the *cap_dac_read_search* capability to *use_cap* as the root user:

```
[09/21/2017 11:05] root@ubuntu:/home/seed/Desktop/hw1# setcap cap_dac_read_search=+ep use_c
ap
[09/21/2017 11:05] root@ubuntu:/home/seed/Desktop/hw1# getcap use_cap
use_cap = cap_dac_read_search+ep
```

Then we execute *use_cap* as the seed user and see that sections (b), (d), and (e) fail:

```
[09/21/2017 11:08] seed@ubuntu:~/Desktop/hw1$ ./use_cap
(b) Open failed
(d) Open failed
(e) Open failed
```

(a) Yes, successful. Since we provided the *cap_dac_read_search* capability via the command line prior to execution, the program is able to access */etc/shadow* in read-only mode.

(b) No, unsuccessful ("open failed"). The capability *cap_dac_read_search* is disabled prior to the *open* attempt so no read access is permitted.

(c) Yes, successful. The capability *cap_dac_read_search* is enabled again prior to the *open* attempt.

(d) No, unsuccessful ("open failed"). The capability *cap_dac_read_search* is dropped/deleted prior to the *open* attempt.

(e) No, unsuccessful ("open failed"). The program attempts to re-enable *cap_dac_read_search* but is unable to do so because this capability was permanently dropped (instead of disabled temporarily); so the *open* attempt fails anyway.