

Linux Capability Exploration Lab

Extra Credit Assignment for CS5460

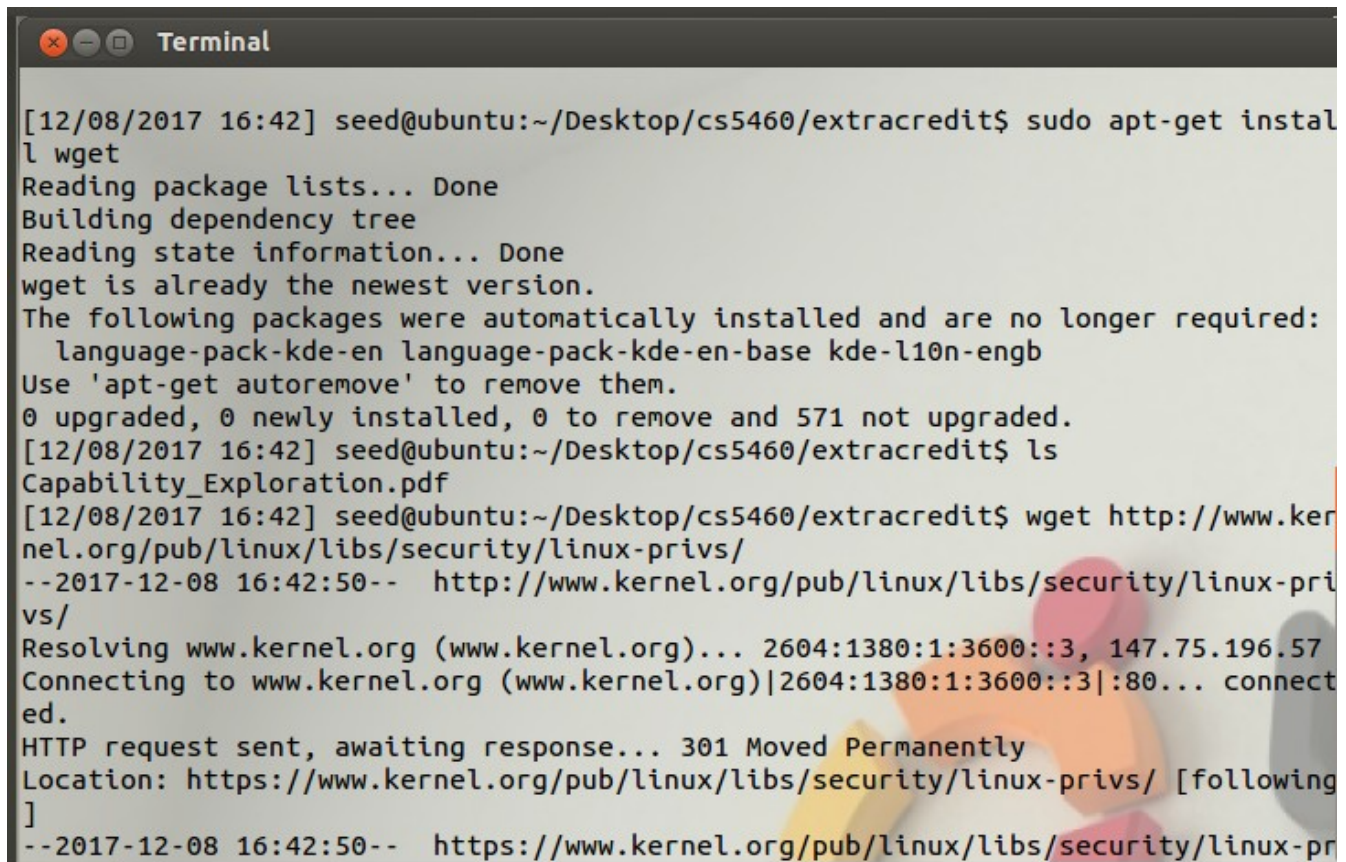
Austin Derbique A01967241s

This is a *Medium Seed Lab*. The seed Lab can be found here:

http://www.cis.syr.edu/~wedu/seed/Labs_12.04/System/Capability_Exploration/

2.1 Install Libcap

I installed libcap from the terminal. Screenshot here:

A terminal window titled "Terminal" showing a series of commands and their outputs. The user is at a prompt in a directory ~/Desktop/cs5460/extracredit. They run 'sudo apt-get install wget', which shows that wget is already installed and lists some packages to be removed. Then they run 'ls', showing 'Capability_Exploration.pdf'. Finally, they run 'wget http://www.kernel.org/pub/linux/libs/security/linux-privs/' which starts downloading the file, showing the URL being resolved and the connection status.

```
[12/08/2017 16:42] seed@ubuntu:~/Desktop/cs5460/extracredit$ sudo apt-get install
l wget
Reading package lists... Done
Building dependency tree
Reading state information... Done
wget is already the newest version.
The following packages were automatically installed and are no longer required:
  language-pack-kde-en language-pack-kde-en-base kde-l10n-engb
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 571 not upgraded.
[12/08/2017 16:42] seed@ubuntu:~/Desktop/cs5460/extracredit$ ls
Capability_Exploration.pdf
[12/08/2017 16:42] seed@ubuntu:~/Desktop/cs5460/extracredit$ wget http://www.ker
nel.org/pub/linux/libs/security/linux-privs/
--2017-12-08 16:42:50--  http://www.kernel.org/pub/linux/libs/security/linux-pri
vs/
Resolving www.kernel.org (www.kernel.org)... 2604:1380:1:3600::3, 147.75.196.57
Connecting to www.kernel.org (www.kernel.org)|2604:1380:1:3600::3|:80... connect
ed.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://www.kernel.org/pub/linux/libs/security/linux-privs/ [following
]
--2017-12-08 16:42:50--  https://www.kernel.org/pub/linux/libs/security/linux-pr
```

2.2 Put SELinux in Permissive Mode

My distribution of Ubuntu does not have SELinux which means I can skip this part. That is what the lab says to do.

3 Lab Tasks

3.1 Task 1: Experiencing Capabilities

This part discusses how the program ping is a SetUID program, meaning that the system changes the user to root to run the program. If there are any vulnerabilities with ping, the entire system can be compromised. This is a screenshot of changing the program to a non setuid program.

```
Terminal
[12/08/2017 16:48] seed@ubuntu:/etc$ ping google.com
PING google.com (172.217.11.238) 56(84) bytes of data.
64 bytes from den02s01-in-f14.1e100.net (172.217.11.238): icmp_req=1 ttl=54 time
=23.2 ms
64 bytes from den02s01-in-f14.1e100.net (172.217.11.238): icmp_req=2 ttl=54 time
=24.0 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 23.205/23.619/24.034/0.442 ms
[12/08/2017 16:50] seed@ubuntu:/etc$ vim /bin/ping
[12/08/2017 16:50] seed@ubuntu:/etc$ sudo chmod u-s /bin/ping
[12/08/2017 16:51] seed@ubuntu:/etc$ which ping
/bin/ping
[12/08/2017 16:51] seed@ubuntu:/etc$ ping google.com
^C
[12/08/2017 16:51] seed@ubuntu:/etc$ █
```

Although it is not a setuid program, we can use setcap to change the /bin/ping to be allowed to have cap_net_raw capability without the setuid power. This is seen below:

```
[12/08/2017 16:51] seed@ubuntu:/etc$ sudo su
[12/08/2017 16:53] root@ubuntu:/etc# setcap cap_net_raw=ep /bin/ping
[12/08/2017 16:53] root@ubuntu:/etc# exit
exit
[12/08/2017 16:54] seed@ubuntu:/etc$ ping google.com
PING google.com (172.217.11.238) 56(84) bytes of data.
64 bytes from den02s01-in-f14.1e100.net (172.217.11.238): icmp_req=1 ttl=54 time
=27.1 ms
64 bytes from den02s01-in-f14.1e100.net (172.217.11.238): icmp_req=2 ttl=54 time
=26.0 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 26.014/26.583/27.152/0.569 ms
[12/08/2017 16:54] seed@ubuntu:/etc$ █
```

Question 1: Please turn the following set_uid programs into non-set-uid programs without affecting the behaviors of these programs: /usr/bin/passwd

To solve question one, these are the commands I executed:

```
Terminal
[12/08/2017 16:56] seed@ubuntu:/etc$ sudo chmod u-s /usr/bin/passwd
[12/08/2017 16:56] seed@ubuntu:/etc$ sudo setcap cap_net_raw=ep /usr/bin/passwd
[12/08/2017 16:57] seed@ubuntu:/etc$ passwd
Changing password for seed.
(current) UNIX password:
Enter new UNIX password:
Retype new UNIX password:
Password unchanged
Enter new UNIX password:
Retype new UNIX password:
Password unchanged
Enter new UNIX password:
Retype new UNIX password:
Password unchanged
passwd: Authentication token manipulation error
passwd: password unchanged
[12/08/2017 16:57] seed@ubuntu:/etc$ █
```

Question 2: Explain the purpose of this capability.

1. cap_dac_read_search - reads the capabilities of the program
2. cap_dac_override - Overrides the default cap permissions
3. cap_chown - Changes the ownership capabilities
4. cap_setuid - changes the user ID to something else
5. cap_kill - kills all capabilities for the program
6. cap_net_raw - enables the raw socket for the capabilities

3.2 Task 2: Adjusting Privileges This task asks to run the following commands to compile and install the updated libcap. I updated libcap here:

```
Terminal
done
/sbin/setcap cap_setfcap=i /sbin/setcap
make[1]: Leaving directory `/home/seed/Desktop/libcap2.22/libcap-2.22/progs'
make -C doc install
make[1]: Entering directory `/home/seed/Desktop/libcap2.22/libcap-2.22/doc'
mkdir -p -m 755 /usr/share/man/man1 /usr/share/man/man3 /usr/share/man/man8
for man in \
    /usr/share/man/man1 capsh.1 \
    /usr/share/man/man3 cap_init.3 cap_free.3 cap_dup.3 cap_clear.3
cap_clear_flag.3 cap_get_flag.3 cap_set_flag.3 cap_compare.3 cap_get_proc.3 cap_
get_pid.3 cap_set_proc.3 cap_get_file.3 cap_get_fd.3 cap_set_file.3 cap_set_fd.3
cap_copy_ext.3 cap_size.3 cap_copy_int.3 cap_from_text.3 cap_to_text.3 cap_from
_name.3 cap_to_name.3 capsetp.3 capgetp.3 libcap.3 cap_get_bound.3 cap_drop_boun
d.3 \
do \
    /usr/share/man/man8 getcap.8 setcap.8 \
    ; \
case $man in \
/*) sub=$man ; continue ;; \
esac; \
install -m 644 $man $sub ; \
done
make[1]: Leaving directory `/home/seed/Desktop/libcap2.22/libcap-2.22/doc'
[12/08/2017 17:02] root@ubuntu:/home/seed/Desktop/libcap2.22/libcap-2.22# █
```

Question 3. Compile the following program and assign the `cap_dac_read_search` capability to the executable. Login as a normal user and run the program. Describe and explain your observations.

The name of this file is `use_cap.c` located in the main directory.

Compiling the code:

```
Dash home [12/08/2017 17:09] root@ubuntu:/home/seed/Desktop/cs5460/extracredit# gcc -c use_cap_final.c
[12/08/2017 17:09] root@ubuntu:/home/seed/Desktop/cs5460/extracredit# gcc -o use_cap_final use_cap_final.o -lcap
use_cap_final.o: In function `main':
use_cap_final.c:(.text+0x35): undefined reference to `cap_disable'
use_cap_final.c:(.text+0x73): undefined reference to `cap_enable'
use_cap_final.c:(.text+0xb1): undefined reference to `cap_drop'
use_cap_final.c:(.text+0xec): undefined reference to `cap_enable'
collect2: ld returned 1 exit status
[12/08/2017 17:09] root@ubuntu:/home/seed/Desktop/cs5460/extracredit# ls
Capability_Exploration.pdf  ping3.png          use_cap.c
index.html                 ping.png           use_cap_final.c
index.html.1               problem1.png       use_cap_final.c~
installing_lib_cap.png     udate_libcap.png  use_cap_final.o
[12/08/2017 17:09] root@ubuntu:/home/seed/Desktop/cs5460/extracredit#
```

The code itself:

```
*use_cap.c x use_cap_final.c x
#include <sys/types.h>
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <linux/capability.h>
#include <sys/capability.h>
int main(void)
{
    if (open ("/etc/shadow", O_RDONLY) < 0)
        printf("(a) Open failed\n");

    /*Question (a): is the above open sucessful? why?*/

    if (cap_disable(CAP_DAC_READ_SEARCH) < 0) return -1;
    if (open ("/etc/shadow", O_RDONLY) < 0)
        printf("(b) Open failed\n");
    /*Question (b): is the above open sucessful? why?*/

    if (cap_enable(CAP_DAC_READ_SEARCH) < 0) return -1;
    if (open ("/etc/shadow", O_RDONLY) < 0)
        printf("(c) Open failed\n");
    /*Question (c): is the above open sucessful? why?*/

    if (cap_drop(CAP_DAC_READ_SEARCH) < 0) return -1;
    if (open ("/etc/shadow", O_RDONLY) < 0)
        printf("(d) Open failed\n");
    /*Question (d): is the above open sucessful? why?*/

    if (cap_enable(CAP_DAC_READ_SEARCH) == 0) return -1;
    if (open ("/etc/shadow", O_RDONLY) < 0)
        printf("(e) Open failed\n");
    /*Question (e): is the above open sucessful? why?*/
}
```

- a. It's interesting. It was not able to open the shadow file unless you gave it root permissions from the setuid.
- b. This failed as well
 - c. This one successful. Because `cap_dac_read_search` was true.
 - d. This was also successful because `cap_dac_read_search` is now true
 - e. With `cap_enable` true, the shadow file successfully opened here as well.

Question 4: If we want to dynamically adjust the amount of privileges in ACL-based access control, what should we do? Compared to capabilities, which access control is more convenient to do so?

It seems the most convenient method is to use a SetUId for the program, but that is not nearly as safe. You'd better off use the `use_cap` to handle capabilities for a given program.

Question 5. After a program (running as normal user) disables a capability A, it is compromised by a buffer-overflow attack. The attacker successfully injects his malicious code into this program's stack space and starts to run it. Can this attacker use the capability A? What if the process deleted the capability, can the attacker use the capability?

They may be able to run the code.. but it will not do anything as the capability is gone. This means that the attacker has no power to do anything with that system. The attacker can only use the process if there is a capability there.

Question 6. The same as the previous question, except replacing the buffer overflow attack with the race condition attack. Namely, if the attacker exploits the race condition in this program, can he use the capability A if the capability is disabled? What if the capability is deleted?

This one would seem to work better than the buffer overflow attack as the race condition attack will be performing two or more operations at the same time. If the capability is disabled, there is more than one process so one will still remain. This means that the attack can still be carried out.