CIS 643 Computer Security

Lab 6 Linux Capability Exploration Lab

**Instructor: Wenliang (Kevin) Du**

Contents

Task 1: Experiencing Capabilities 2

Question 1: Explore non-Set-UID programs 2

Question 2: Explore capabilities 2

Task 2: Adjusting Privileges 6

Install disable/enable/delete functionality for capability 6

Question 3: cap\_dac\_read\_search capability Adjustment 7

Question 4: Capabilities vs. ACL 8

Question 5: Capabilities vs. Buffer-overflow 8

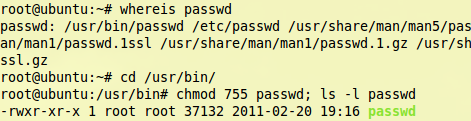
Question 6: Capabilities vs. Condition-Attack 8

## Task 1: Experiencing Capabilities

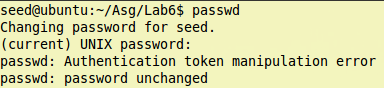
### Question 1: Explore non-Set-UID programs

**Description: Turn the following Set-UID programs into non-Set-UID programs, without affectingthe behaviors of these programs.**

Login as root, change /usr/bin/passwd to mode 755



Login as normal user, and type “passwd” command



We could see that we do not have authentication to change the password for the user now.

### Question 2: Explore capabilities

1. cap\_dac\_read\_search

**Functionality**: open a file owned by root as the normal user

**Program for test**: “grep” command

“grep” command could search text in a file, but if the user is not file owner, it would not allow user to search in the file.

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 9.12.27 PM.png

Let’s set cap\_dac\_read\_search capability to grep command to see differencesMacintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 9.13.36 PM.png

Type “grep” command again:

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 9.29.08 PM.png

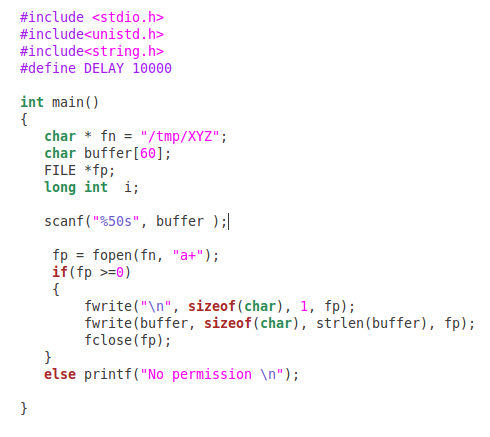
Now, we could use “grep” to get content from the file which own by root, and other user do not have permission to read.

1. cap\_dac\_override

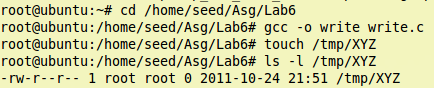
**Functionality**: write to a file owned by root, using a normal user

**Program for test**:

The following program will open file /tmp/XYZ which is owned by root, then if open successfully, it will append text that input by user.



Compile it as root, and then run the program as normal user.



From the following result we could see that normal user do not have permission to write text into file that owned by user.

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 9.52.51 PM.png

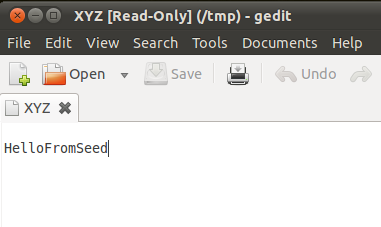
Now, let’s assign “cap\_dac\_override” capability to “write” program.

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 10.01.33 PM.png

Write the program again：

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 10.00.40 PM.png

No permission warning, if we open the /tmp/XYZ file, we could also see that we have written text to the file successfully.

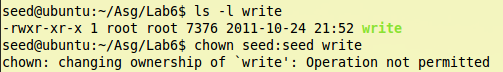


1. cap\_chown

**Functionality**: assign capability that changes the file own to normal user.

**Program for test**: “chown” command

Usually, normal user could not change the file owner and group



But if root set cap\_chown to chown commandMacintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 10.30.34 PM.png

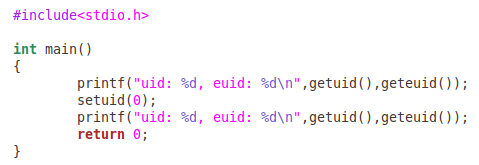
Normal user would get ability to change the owner and group of files and programs.

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 10.30.59 PM.png

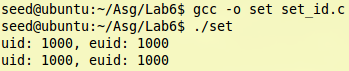
1. cap\_setuid

**Functionality**: allow program to change real and effective user to other user

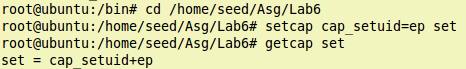
**Program for test**:



Use normal user to compile the program and run



We saw that the id is still the same. setuid() function does not affect the program.

Now use root to set cap\_setuid to the program

This time, the program set the uid to 0 successfully

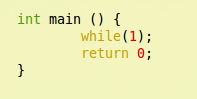
Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 11.01.06 PM.png

1. cap\_kill

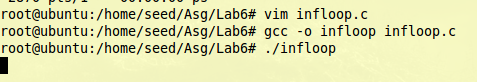
**Functionality**: allow normal user to kill process run by root

**Program for test**: “kill” command

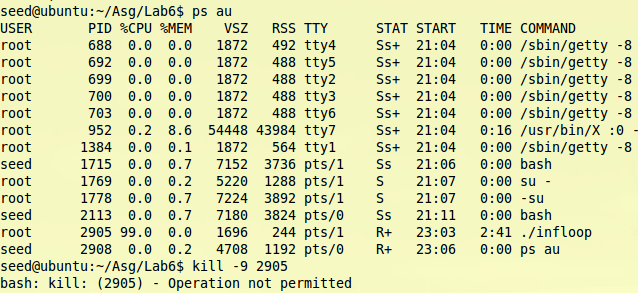
Write an infinite loop program as following



Use root to compile and run the program



Find the pid of “infloop” as normal user.

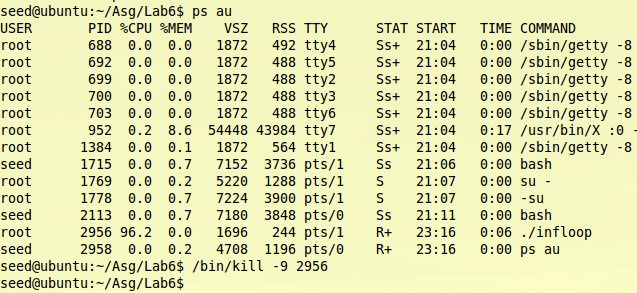


We try to kill the process, but it is not permitted, as the process is owned by root.

Now assign cap\_kill capability to /bin/kill

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 11.18.34 PM.png

Run the loop again, and let normal user to kill it



Now this time we really kill the process owned by root, and in root window, the loop also stop.

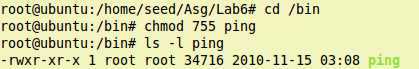
Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 11.21.24 PM.png

1. cap\_net\_raw

**Functionality**: allow normal user send request to net hosts.

**Program for test**: “ping” command

The /bin/ping is a Set-UID program, so we need to change it to normal program.

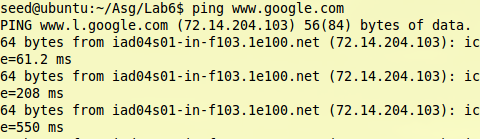


Try to ping [www.google.com](http://www.google.com) as normal user

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 11.30.44 PM.png

Assign cap\_net\_raw to /bin/pingMacintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 11.32.14 PM.png

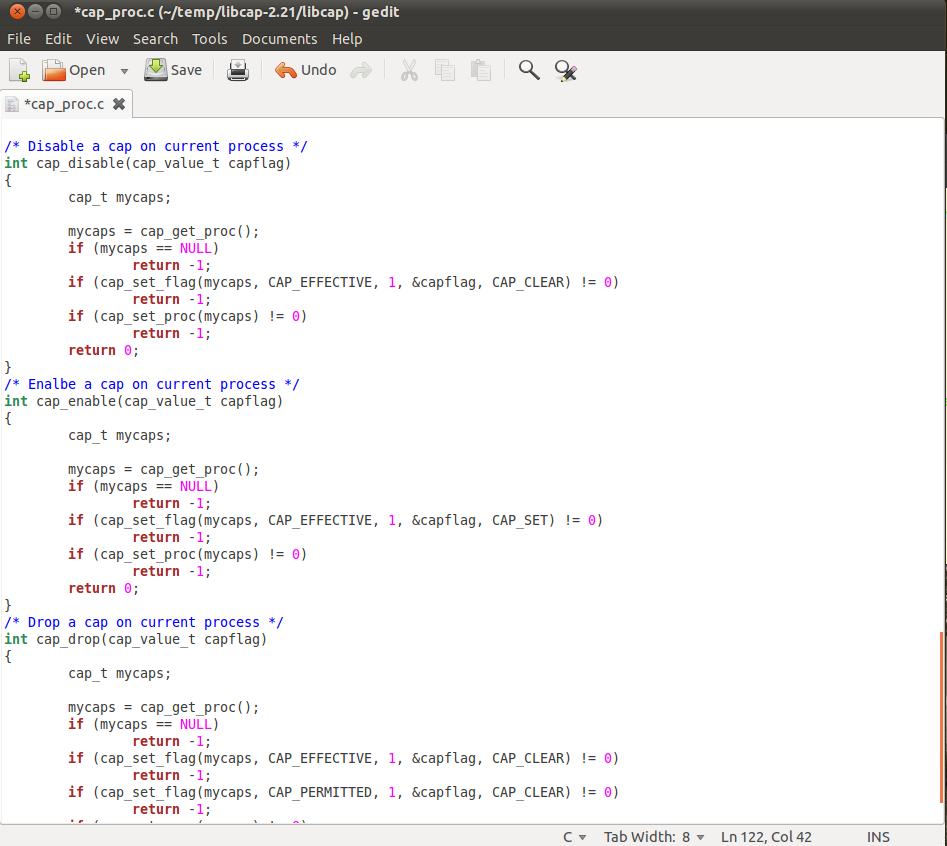
Ping again:



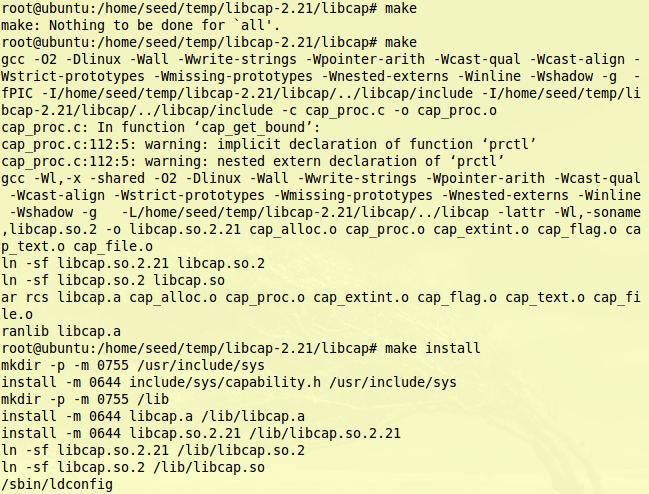
We got response from host this time

## Task 2: Adjusting Privileges

### Install disable/enable/delete functionality for capability

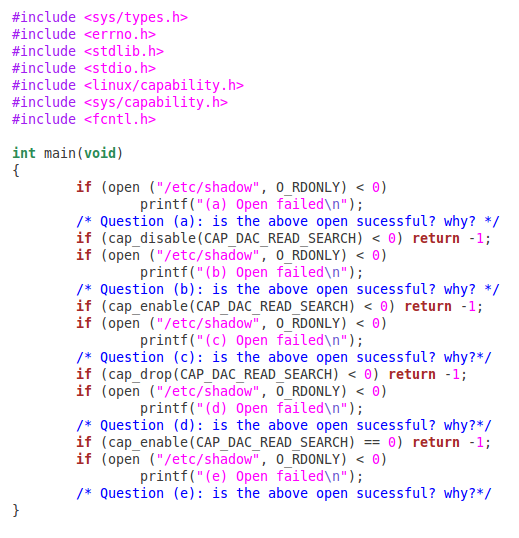
Add functions to cap\_pro.c

Make the source codes and install them



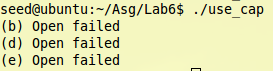
### Question 3: cap\_dac\_read\_search capability Adjustment

Program to run



Compile it and assign cap\_dac\_read\_search capability to it.Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-10-24 at 11.48.47 PM.png

Run the program as normal user



|  |  |  |
| --- | --- | --- |
| Statement | Result | Reason |
| (a) | Succeed | The process has cap\_dac\_read\_rearch capability |
| (b) | Failed | The capability has been disable, so the program lost the capability temporary |
| (c) | Succeed | The capability has been resumed |
| (d) | Failed | The process lost the capability |
| (e) | Failed | Since the capability has been dropped, it could not been enabled anymore |

### Question 4: Capabilities vs. ACL

**Description: 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?**

When we want to dynamically adjust the privilege, the capability would be convenience as it could be disabled temporary. And it makes privilege with more subsections. So if we lost or disable one capability, we still could use other capability.

### Question 5: Capabilities vs. Buffer-overflow

**Description: 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 uses the capability?**

Since the enable/disable capability functions are library function, so attacker may find the address of enable\_cap function to enable the capability, then use the capability to do evil thing.

But if the capability is deleted, it could not be enabled again, so I think in this situation, attacker would not have ability to use the capability.

### Question 6: Capabilities vs. Condition-Attack

**Description: 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?**

The program only checks the capability when it calls specific process (like open a file. So if the disable\_cap is called after action, there will be a chance for attacker to get capability. If the disable\_cap is called before action, attacker would not get chance to use capability.

If the capability is deleted, the situation would be the same as it disabled.