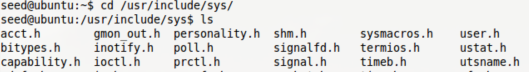
Linux Capability Exploration Lab

Yukui Ye

SUID: 439644268

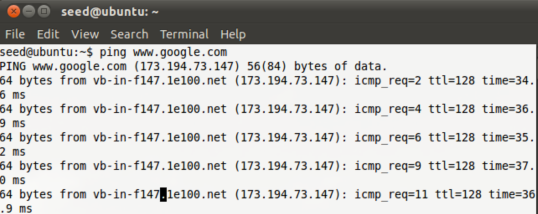
Lab setup: check to see wether capability.h has been installed in the system.



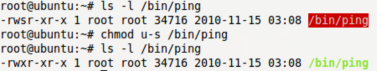
we found the capability.h file , so the libcap library has already been installed.

Task1: Experiencing Capabilities

1)login as formal user and run the command. We can tell from the screenshot below the program is successfully running.



2)login as root to turn /bin/ping into a non Set-UID program.

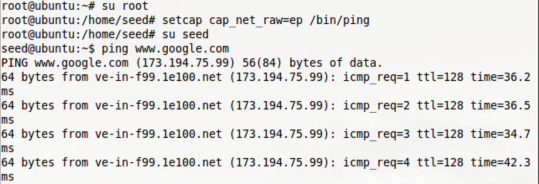


3)Run the program again, surprisingly, it cannot work out like before.

Macintosh HD:Users:yukuiye:Desktop:4.png

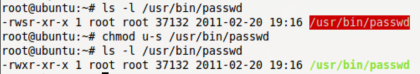
The failure of running the program is due to ping needs to open RAW socket, which can only conducted by root.

4) From previous test, we can tell that we do not need to give ping too many power so let us just assign cap\_net\_raw capability to ping.



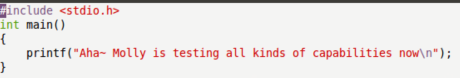
the above screenshot tells us that after we assigned cap\_net\_raw to ping，the program runs successfully .

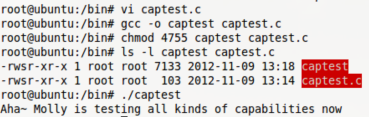
Question 1: login as root and turn /usr/bin/passwd from Set-UID program into non-Set-UID program.



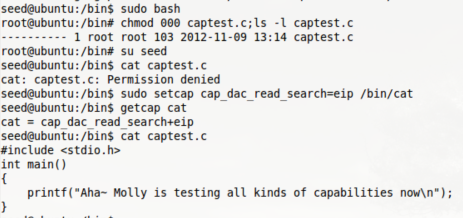
Question2:

Create own program named captest under directory /bin/

Compile it and set it as a set-uid program

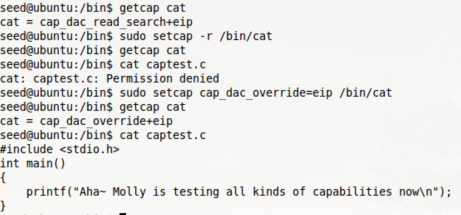


1) Cap\_dac\_read\_search: Bypass file read permission checks and directory read and execute permission checks. After enable this capability , it can run any program that without read permission.



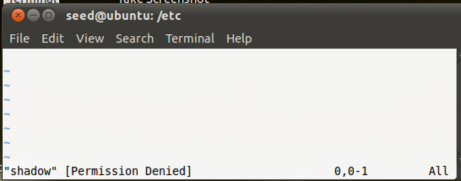
To test this command, I change captest.c to 000, when we run it as normal user, it shows permission denied. After that I setcap cap\_dac\_read\_search to read command ‘cat’, then run it again, surprisingly, we can read the content of the program. Therefore, cap\_dac\_read\_search really do bypass the read permission, it can run any program that without read permission.

2)Cap\_dac\_override: Bypass file read, write, and execute permission checks. (DAC is an abbreviation of "discretionary access control"). In first example, I am trying to read the content of captest.c.



First remove the capability that appeared in previous test, try to read the program, we found that permission denied, then enable cap\_dac\_override capability to read permission ’cat’, then we can successfully read the program.

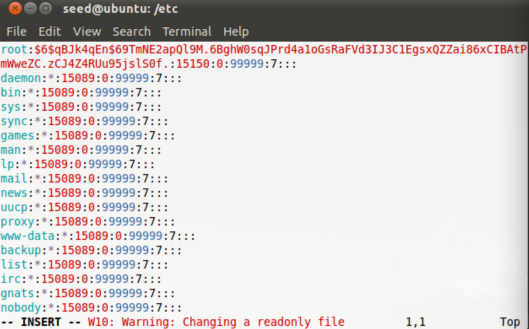
Another example is bypassing edit or write permission after we enable the capability of cap\_dac\_override. Before enable this capability, when we type “vi shadow”, we cannot even open the /etc/shadow file.



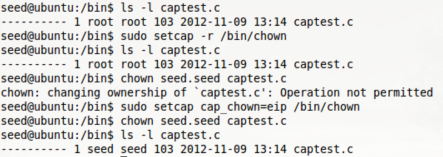
The following code is we enable the capability on “vi” command, and then try to open it.

Macintosh HD:Users:yukuiye:Library:Application Support:QQ:Users:394852457:QQ:Temp.db:3A6BB8FD-99EA-4CF6-B482-04C7C352E067.png

The following screenshot is the result, we successfully opened the file, and you can insert any sentence in this file.



3)Propose: cap\_chown: Make arbitrary changes to file UIDs and GIDs

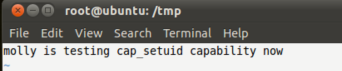


From above screenshot, first we remove the capability set by previous test, then intent to change the owner of the file captest.c from root to seed, it turns out that Operation not permitted, then we set cat\_chown capability to the command ‘chown’, after that, we do chown command again, it successfully change the owner of the file to seed.

4)Cap\_setuid: make forged UID when passing socket credentials via UNIX domain sockets.

First we create a file name sidcontent which contents the sentence that going to show out . Then create the program which will forge a uid when passing socket.

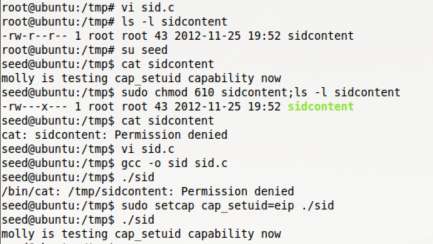
Macintosh HD:Users:yukuiye:Library:Application Support:QQ:Users:394852457:QQ:Temp.db:92ED3C22-5E6A-4331-8281-E719552EEF9F.png



Then create the program which will forge a uid when passing socket. The program named sid.c

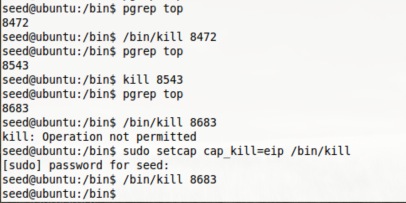


Then we set the content program sidcontent to no read permission for others, then do cat sidcontent, we can see from below picture, it shows permission denied. Then we go to compile the sid.c program and run it, it also says permission denied.

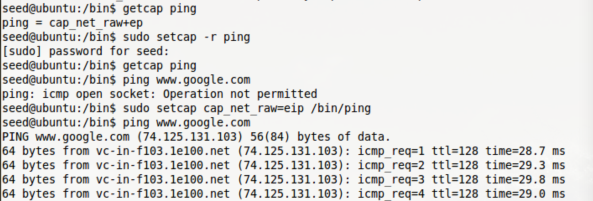


After we setcap cap\_setuid to ./sid, then run it , it successfully run the program by showing out the content in sidcontent file.

5)Propose: cap\_kill: Bypass permission checks for sending signals . In this task, I run top on other terminal in root permission, and then use seed try to kill the process, but as we observed in the following screenshot, it shows operation not permitted, then we setcap cap\_kill capability to kill, and run it again, it finally successfully kill the process.

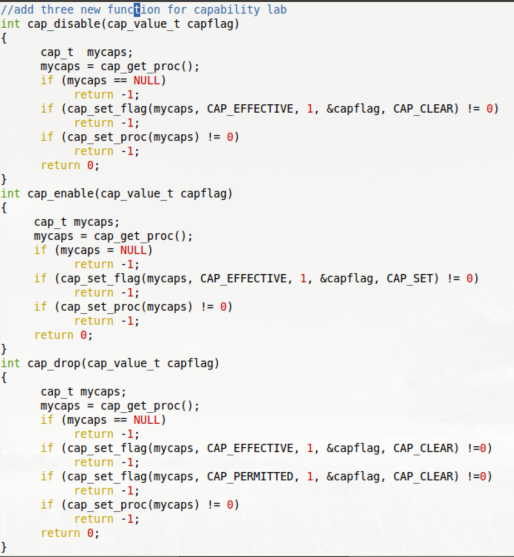


6)Propose: cap\_net\_raw:Enable normal user to use RAW and PACKET sockets.In this example, we first check whether ping has any capability, it shows it has one then we remove it, and run ping, it show permission denied, then we setcap cap\_net\_raw to ping , and run it again, it works !

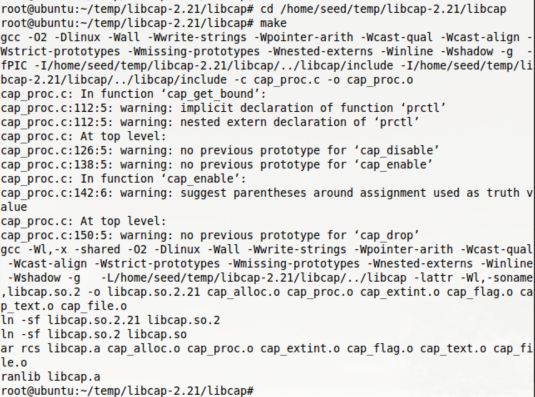


Task2: Adjusting Privileges

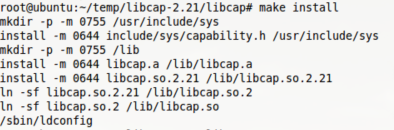
1)Add the following three function to /home/seed/temp/libcap-2.21/libcap/cap\_proc.c.



2) Run commands to compile and install the updated libcap.

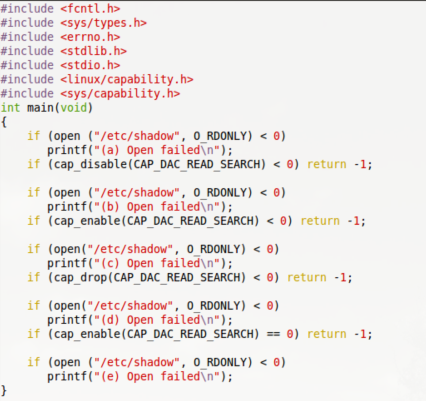


then install:



Question3:

First, create use\_cap.c file



Compile this program:

Macintosh HD:Users:yukuiye:Library:Application Support:QQ:Users:394852457:QQ:Temp.db:727D9A20-6204-4A63-A693-7FF666E6D019.png

Assign the cap\_dac\_read\_search capability to the executable file ” ./use\_cap”. And login as normal user, run the program.

Macintosh HD:Users:yukuiye:Library:Application Support:QQ:Users:394852457:QQ:Temp.db:EE1A9685-C6DC-41D6-8197-BB8E3EE713A0.png

Macintosh HD:Users:yukuiye:Library:Application Support:QQ:Users:394852457:QQ:Temp.db:4858E79D-8F20-4B23-956B-33F9535AE509.png

Explain the result:

Compare with the code and the result, program cannot open shadow file , it comes(b) open failed, then after we enable it , it can get the capability and it shows (c)is successfully open, if we drop the capability, (d)open failed, and we cannot enable it again after drop the capability, though, in the code, even we tried to enable its capability again, (e)still open failed.

Question4: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?

Answer: ACL is a list of access control entry, which give access permission to a user or group on a given file or folder. In ACL, if we want to grant permission to other user/group, we always need to login as root or superuser, and use “chmod” command to grand permission on file to the aimed user. While by using capabilities, we can bypass some permission check, even if we were not supposed to have permission on accessing this file. It is more convenient for normal user since you do not need to ask access permission from root, but it is more danger.

Question5: 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?

Answer: Yes. After normal user disables a capability A, the attacker can still use the capability A by enabling it in his malicious code, but if the process deleted the capability, the attacker cannot use the capability.

Question6: 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?

Answer: If the attacker exploits the race condition in this program, he can still use the capability A no matter the capability is disabled or deleted. That is because the malicious code will always run before the capability statement in the race condition attack.