Matthew Kachar

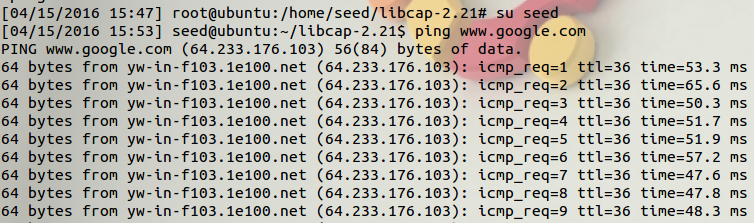
COSC4931

Assignment 7

Linux Capabilities Lab

Objective : Gaining experience with capabilities, the advantages of access control, least privilege techniques and analyze overall design. Understanding how capabilities effect process execution.

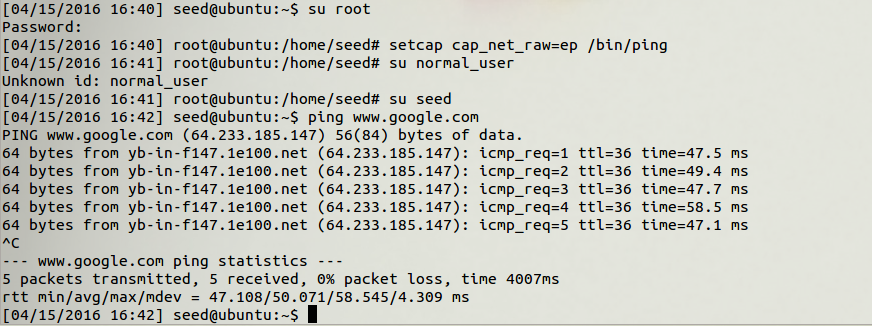
The ping command in Linux has capabilities to that of a *Set-UID* program. The ping command needs root user access to open a RAW socket.



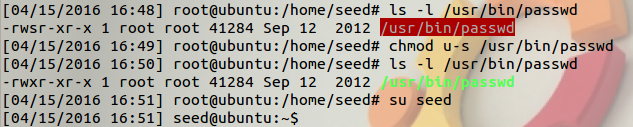
When the capability to change user ID, the command essentially fails



However we can give the ping command the capability of opening a RAW socket without acquiring root user access.



We can do the same with other programs with *Set-UID* capabilities such as *passwd*.

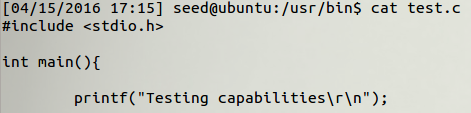


Using a simple C program to test the list of Linux capabilities for the second part of this task:

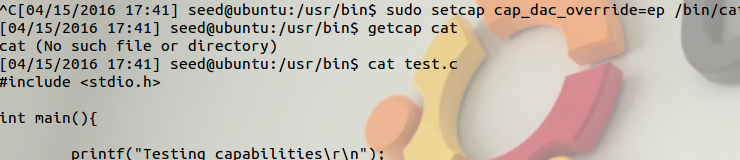
cap\_dac\_read\_search –This capability overrides the discretionary access control restrictions for read and search commands on a given file. To test this we can attempt to read a file with the restrictions enabled then set then set the capabilities to allow the *cat* read command.



Setting permission to execute *cat* # setcap cap\_dac\_read\_search=ep /bin/cat—we can read the code in the test.c file.

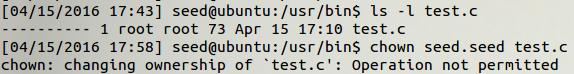


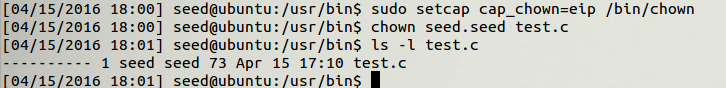
cap\_dac\_override –The override capability can be used in the same way for discretionary access control permissions allowing us to read the contents of test.c



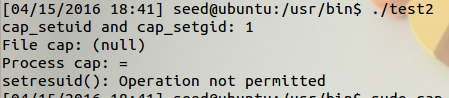
Although, unlike the read\_search capabilities, the dac\_override extends execute capabilities.

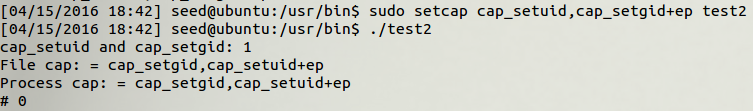
cap\_chown –The change ownership capability allows the program to change user ID by removing the change UID restrictions. For this test of the cap\_chown capability we can try to change the ownership with the restriction enable then set the new program capabilities and observe the difference after.





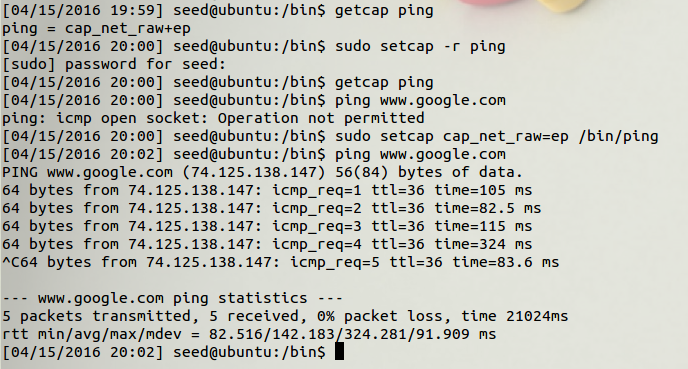
cap\_setuid –This capability allows the program to acquire and manipulate the user ID designated to the program during execution. For the test we can write a simple set UID program which attempts to change the UID similar to the environmental variables lab. With the restrictions in place we are denied permission, then setting the capabilities we can access the UID and manipulate it.



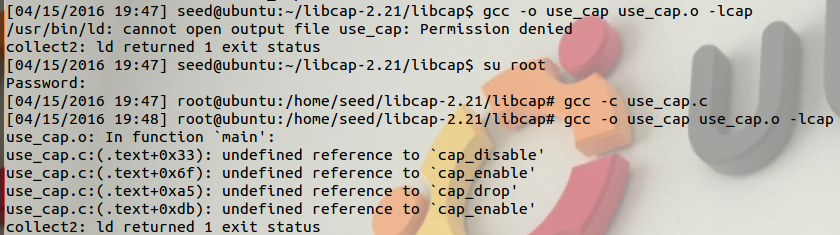


cap\_kill –This capability lifts restrictions on sending signals and gives a program the permission to terminate other processes. Different from termination notification, the kill command can arbitrarily dismiss a process without accounting for status in essence dumping any current data.

The cap\_net\_raw –removes the restrictions on acquiring the RAW socket. For this test we can remove the capability and attempt to ping google like the first example then restore the capability.



Adjusting privileges task; we see how execution of programs in Linux can be manipulated by access to different capabilities. And this can even extend to the process of compiling.



The cap\_proc.c and use\_cap.c cannot even be compiled unless the correct capabilities are enabled to allow the program access to the necessary directory. The program cannot open the files in libcap with given permission. Once given the read capability the program can open the shadow files.

We can dynamically adjust privileges by changing capabilities but if it is not an aspect of routine processes or built into commands it is more convenient to change to root user since access is more or less free and does not require accounting for privileges.

A user injecting malicious code can always enable capabilities that may have been disabled during execution, but if the capability is deleted it may not be accessed.

If there is a race condition vulnerability, a change in the programs capabilities with always be executed before the race condition, that is while the program may have stalled out in some point of execution, whatever level of access the attacker desires can be configured.