Will Pond

**TASK1**

This is the C code of catall.c using the system() method that I will use in the next step.

A screenshot of a computer program

Description automatically generated

In this section I made directories for Lab1 and make a duplicate of catall.c named task1 then, begin to change the ownership to root.

A screenshot of a computer program

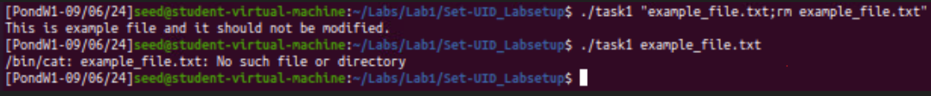
Description automatically generated

In this section I modified the permissions of task1 to be executable and make example file to use when compiling the C program.

A screenshot of a computer program

Description automatically generated

In this section I prove that you can remove files that you do not have permission to be writable.



This is the C code I use for catall.c that uses the execve() method and I will use this in the following step.

A screenshot of a computer program

Description automatically generated

In this section I made a duplicate copy of catall.c and named it task2 and begin changing the ownership to root.



In this section I modified the permissions of task1 to be executable and make example file to use when compiling the C program. Then prove that a file cannot be deleted because does not invoke the shell, so it stops the attack.

A screenshot of a computer program

Description automatically generated

**TASK2**

This is the C code for cap.leak.c and I will use this in following steps.

A screenshot of a computer program

Description automatically generated

In this section I am opening /etc/zzz and modifying it, then creating environment variable name test for /etc/zzz. Finally beginning to change it to root.

A computer screen with white text

Description automatically generated

In this section I create a duplicate of cap\_leak.c named task3, then compile the program to get the descriptor value. Next, put that in the environment variable of test. After that I did the cat command to display the context of the file of etc/zzz. Then did another environment variable name test2 with the same descriptor value to override root to access the file.

A screenshot of a computer screen

Description automatically generated

Reflection

In this Task1 was able to Invoke external programs using system() and exeve(). I observed how system() can run new programs but can be dangerous in the Set-UID programs because someone can gain access to your device and can delete files which surprised me. In the exeve() I observed that it does not only person to gain access to your device and delete files because it does not invoke shell in Set\_UID programs which I found interesting. In Task2 I observed that setuid() system can revoke privileges but, in that process, it doesn’t close the file calling to leak and a normal user can exploit this vulnerability. This is the reason why you always want to close your files during your programs, so this won’t happen.