**NAME *: LAKSHITHA RAJ VASANADU***

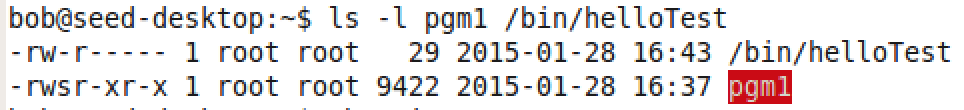
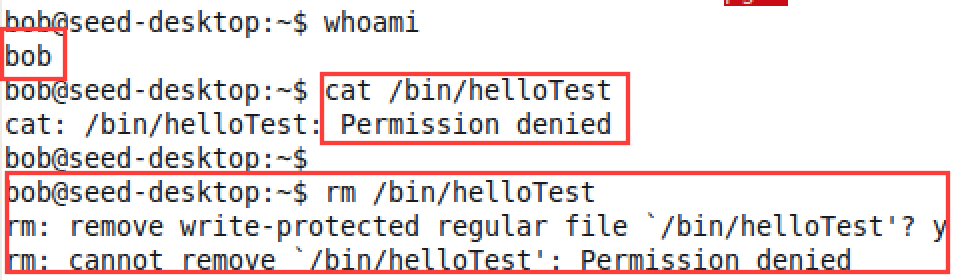
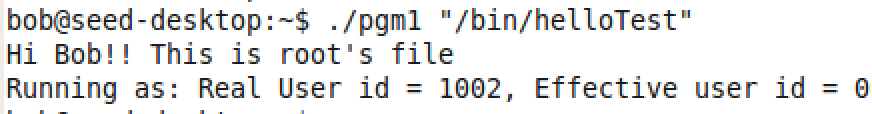
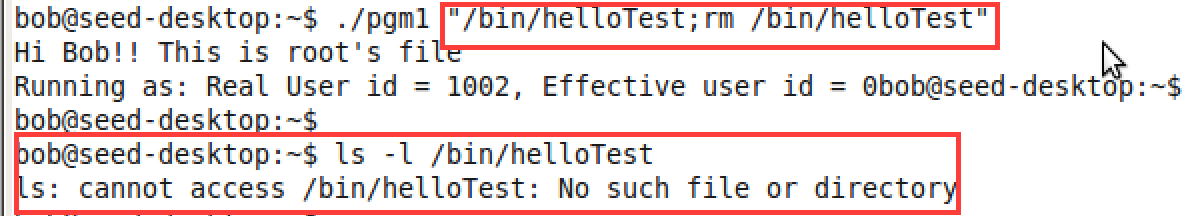
**ID *: 00001115006***

**COEN-225**

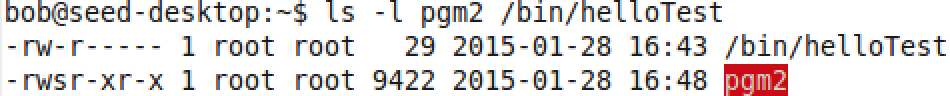
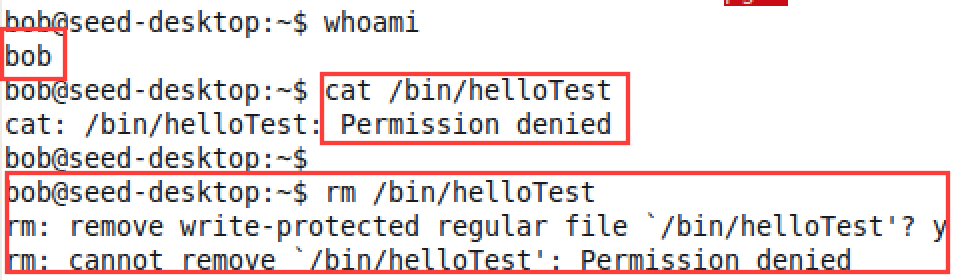
**LAB ASSIGNMENT 2**

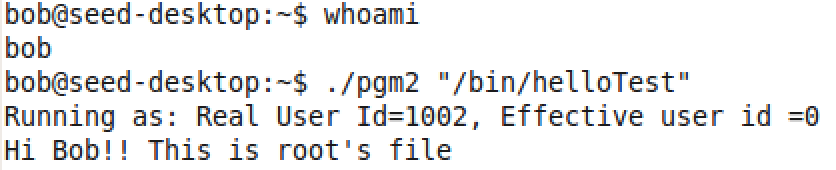
**Set-UID Program Vulnerability Lab**

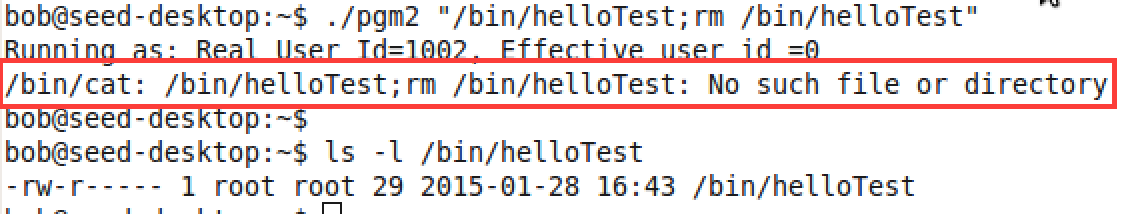
1. **system( )** versus **execve( )**
2. **Using system( )**

* **Steps Performed:**
* Create the given program with **q=0** and call it as **pgm1.c**. Lets its executable be **pgm1**, which is a set-UID program.
* Create another test file called **helloTest** in the root-owned **/bin** directory so that normal users can’t read/write to it.
* The permissions for the files are shown as:
* Bob, being a normal user tries to read/ delete the **helloTest** file but he is unable to do so:
* Bob now runs the set-UID program to display the **helloTest** file: 
* Bob is able to do so as he gains the root access while running the program. Bob tries to exploit this as follows:
* Bob **succeeded to delete** the file even though he does not have write permissions. Bob uses **“;”** character which is the command separator character in shell to exploit. This is called as command line injection.
* **Observation:**
* **system()** by default uses **/bin/sh** to execute the commands passed as arguments to it. Since **“;”** is a delimiter for commands on shell, the string before “;” is used as an argument to /bin/cat and the latter is executed as a separate command. Since, it’s a set-UID program these commands have root privilege. Malicious activities such as deleting root-owned files can happen.

1. **Using execve( )**

* **Steps Performed:**
* Create the given program with **q=1** and call it as **pgm2.c**. Lets its executable be **pgm2**, which is a set-UID program.
* Create another test file called **helloTest** in the root-owned **/bin** directory so that normal users can’t read/write to it.
* The permissions for the files are shown as:
* Bob, being a normal user tries to read/ delete the **helloTest** file but he is unable to do so:
* Bob now runs the set-UID program to display the **helloTest** file:

****

* Bob is able to do so as he gains the root access while running the program. Bob tries to exploit this as follows:
* Bob is still unable to delete the **helloTest** file even though he has root access and he repeats the same attack as in that of (a).
* **Observation:**
* **execv( )** uses **/bin/cat** as the path to the file to be executed. The second parameter is used as an argument to the first file. Here, it is an argument to the /bin/cat command. It does **not** recognize **“;”** as delimiter for the commands. As a result, it throws ***no such file error*** and hence could not be exploited in this scenario.

**Although system() and execv() are kind of similar in functionality, the way they achieve it differs. In the given context system() could be exploited but not execv().**

**6) The LD\_PRELOAD environment variable**

**7) Relinquishing privileges and cleanup**