**21CY682 – Secure Coding lab – I**

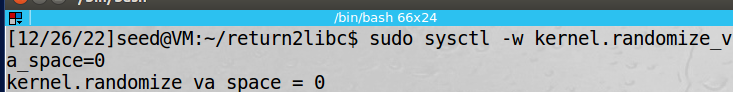
**Assignment Topic: Return to libc**

**Name: Ramya Ajay**

**Roll Number: CYS22004**

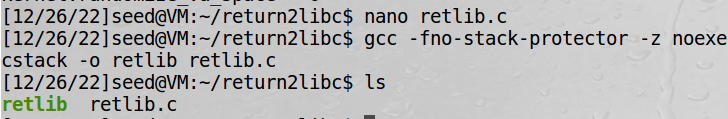
**Date: 7/01/2023**

Ubuntu and several other Linux-based systems use address space randomization to randomize the starting address of heap and stack, making guessing the exact addresses difficult. We can disable this feature using command .



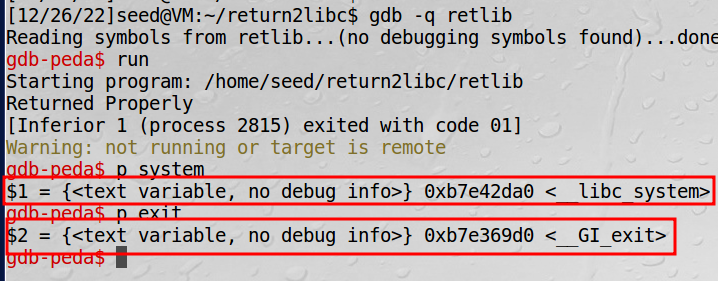
**Task 1: Finding out the addresses of libc functions:**

* gcc -fno-stack-protector -z noexecstack -o retlib retlib.c this command compiles retlib.c. After it is compiled we need to change the owner of the file to root using command sudo chown root retlib then make it executable using command sudo chmod 4755 retlib
* Now we need to create a badfile with whatever content we like or maybe leave it empty.
* Next we can run gdb compiler on retlib using command gdb -q retlib
* Now that we are inside gdb, we need to run the program using command run
* Now we can get the address of system() and exit() using command p system and p exit respectively.





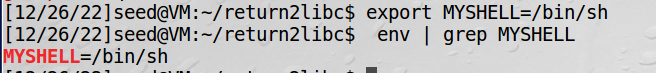


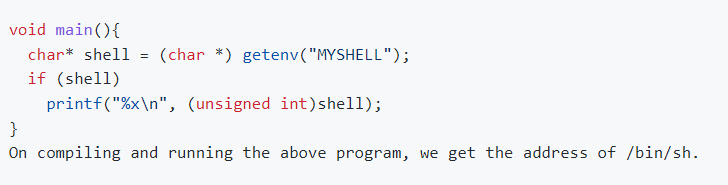


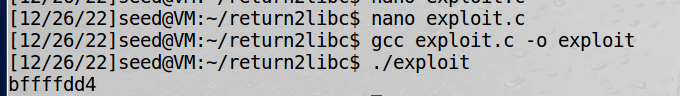
Now we got the values for system and exit address location.

### Task 2: Putting the shell string in the memory

* Our attack strategy is to jump to the system() function and get it to execute an arbitrary command.
* Since we want the shell prompt, we want the system() function to execute the “/bin/sh” program.
* For that, we need to place /bin/sh into the memory and know its address so that it can be passed to the system() function.
* We can define a new variable MYSHELL and let it contain the string /bin/sh using command export MYSHELL=/bin/sh.
* We will use the address of MYSHELL as the argument to the system() call. The following program gives the location of MYSHELL.







### Task 3: Exploiting the buffer-overflow vulnerability

### 

### 

### Task 4: Turning on address randomization

* We can turn on the address randomization using command sudo sysctl -w kernel.randomize\_va\_space=2
* This time ./retlib gives segmentation fault. This is because buffer overflow occurred but address of system(), exit() and /bin/sh varied every time. So we can not get a hold on for an exact address. This is why attack was not successful.

### 