Overview:
Shellcode is widely used in many attacks that involve code injection. Writing shellcode is quite challenging. Although we can easily find existing shellcodes from the Internet, there are situations where we have to write a shellcode that satisfies certain specific requirements. Moreover, to be able to write our own shellcode from scratch is always exciting. The purpose of this lab is to help you understand these techniques so you can write their own shellcode.

Description:
For this lab, you will need to create a shellcode that can execute some program, which can be "/bin/cat /etc/passwd". We also created an "execve.c" program to do the execve system call, in order to help you find out what arguments you will need. And after you execute the binary, the passwd file will be displayed.
In your shellcode, you need to use system call execve, which is 59, and below is a hint you can use in your shellcode to invoke execve(). But you need to figure out the rest. We also provide the Makefile to help you generate the .sc file. Once you finish your shellcode, you can directly use "make" to get both binary and .sc files. Please name you .s file end with "*64.s"(ex. shellcode64.s), then you can do make.

Useful Link: (To find corresponding syscall number)
https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/

After you finish the shellcode, we have prepared the payload generator python file under:
$ cd lab07/sc_exploit
$ python3 exploit.py example.sc > payload
So you only need to give the buffer start address and the number of bytes to overwrite the return address.
The last step for you will be using GDB to find out those information above and use the payload you generated to overflow the vulnerable program "exercise.c" and with executing your chosen program.

Task:
1. Create shellcode to execute any program with execve()
2. Overflow the "exercise.c" with your payload, and executing the program you decide to use

Submission:
Please submit a zipped file with your shellcode, modified exploit.py file and a report with screenshots to show you have successfully executed the program by overflowing the "exercise.c" using your generated payload.