**Homework 2**

**EECS 221 Advanced System Security (Winter 2019)**

**Due on 02/28/2019 11:59 PM PST**

Submit the report through CANVAS.

Do not copy from the Internet, lecture notes, or others. Please use your words in your solutions.

Student Name:

Student ID:

**This assignment requires Ubuntu VM 16.04 to be installed on your machine. Please follow the instruction here:** [**http://www.cis.syr.edu/~wedu/seed/lab\_env.html**](http://www.cis.syr.edu/~wedu/seed/lab_env.html)**.**

You will follow the instructions of Buffer Overflow Vulnerability Lab (See “Description” of <http://www.cis.syr.edu/~wedu/seed/Labs_16.04/Software/Buffer_Overflow/>) and finish the lab tasks **1, 2, 4, 5, 6**. The goal is to gain experiences with buffer overflow vulnerability and learn how to mitigate this problem. The report should contain the steps you take, the code you write, the observations you see, and the necessary screenshots to justify your observations.

**Task 1: Running Shellcode:** Follow the instructions, compile call\_shellcode.c, run it and report what you have observed.

**Task 2: Exploiting the Vulnerability:** fill the missing code in exploit.c or exploit.py, compile and run it to generate badfile. Then, run stack and see if you can get the shell.

Hint: The shell code present in badfile will be copied into program’s stack due to buffer overflow. It will be executed only if the return address of bof() contains shellcode’s address. Therefore, the shellcode should be placed at the right distance from the start of badfile. To construct badfile, you need to

1. Find the address of buffer[] of bof()
2. Find the distance of the return address from buffer[]
3. Find the distance of shellcode from buffer[]

and then construct the missing instructions using those addresses in exploit.c or exploit.py

To learn the addresses, you need to compile stack.c with flag “-g” to generate debugging information and use gdb (here is a quick tutorial <http://www.cs.toronto.edu/~krueger/csc209h/tut/gdb_tutorial.html>). The stack frame pointer register is ebp.

**Task 4: Defeating Address Randomization:** The stack begins at a different location in the virtual address space if ASLR is turned on, but the entropy of randomness is not high enough. Follow the instructions and report your findings.

**Task 5: Turn on the StackGuard Protection:** Follow the instructions and report your findings.

**(Bonus)** can you defeat StackGuard using what we have learnt in lecture?

**Task 6: Turn on the Non-executable Stack Protection:** Follow the instructions and report your findings.