CSCD 437
Lab 3
Smash the Stack

PART 1

1) I have provided you an AWS virtual machine running Ubuntu 20.04
2) This might help with some of the stack protections https://stackoverflow.com/questions/2340259/how-to-turn-off-gcc-compiler-optimization-to-enable-buffer-overflow
3) You may need to
    a. Turn off ASLR with `echo 0 | sudo tee /proc/sys/kernel/randomize_va_space`
4) When you compile stackOverload.c -o stackOverload
    a. You may need to turn stack protection off with `-fno-stack-protector`
    b. You may have to modify other protections
5) Run the stackOverload executable via ./stackOverload
    a. You will need to pass the executable a String
    b. **Note:** The Borland compiler I used to demonstrate is 32 bits and Ubuntu gcc compiler is 64 bits
6) Run perl hackOverrun.pl
    a. Change the $args line for the string and the address
    b. Note: You may need to pad the initial set of letters in hackOverrun.pl to overflow to the return address location on the stack
    c. Note: This lab presumes hackOverrun and stackOverload are in the same directory

NOTE

- The only changes you can make to stackOverload.c is to add more %p\n to the following two printf statements in the *foo* function.
    - printf("My stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
    - printf("Now the stack looks like:\n%p\n%p\n%p\n%p\n%p\n%p\n\n");
- In HackOverrun.pl you will need to change line #1 to the contents of line #2
    1. $cmd = "stackOverrun ".$arg; to
    2. $cmd = "./stackOverload ".$arg;
- The only line you can change in HackOverrun.pl is:

```
$args = "ABCDEFGHIJKLMNOP"."\x90\x11\x40";
```

NOTE: Completing only step 1 will get you 85% of the points

WHAT TO TURN IN FOR STEP 1:

- Output captures in a single PDF that show your exploit worked/didn't work in AWS. Note: There will be output in either case of it worked or didn't work.
- Screen captures of hackOverrun.pl and stackOverload.c in the single pdf – this is the same PDF as the bullet above.
- Name your PDF your last name first letter of first name-step1.pdf (Example: steiners-step1.pdf)
- The hackOverrun.pl and stackOverload.c files
- Included in the single PDF should be a narrative explaining:
    - settings you had to turn off
    - struggles you had/websites you used
    - Your process to getting the buffer overload to work properly. If you couldn't get it to work then a narrative explaining what happened and what needs to be done so it might work.

There is a second page keep reading

STEP 2

- You can work in teams of two for Step2
- Create a folder named step2 and
- Copy the hackOverrun.pl and stackOverload.c from step 1 and paste them into the step2 folder
- Instead of printing to the screen in the bar function actually inject code that opens a terminal.
- Read chapter 2 in the Shellcoders Handbook
- You can't use the system command or any system calls.
- You need to keep the provided code as working code in stackOverload.c – You can add code

WHAT TO TURN IN FOR STEP 2:

- Output captures in a single PDF that show your exploit worked/didn't work in AWS. Note: There will be output in either case of it worked or didn't work.
- Screen captures of hackOverrun.pl and stackOverload.c in the single pdf – this is the same PDF as the bullet above.
- Name your PDF your last name first letter of first name-step2.pdf (Example: steiners-step2.pdf)
- The step2 folder including hackOverrun.pl and stackOverload.c files
- Included in the single PDF should be a narrative explaining:
    - The name of who you worked with on part 2. If solo then state "no teammate"
    - settings you had to turn off
    - struggles you had/websites you used
    - Your process to getting the buffer overload to work properly. If you couldn't get it to work then a narrative explaining what happened and what needs to be done so it might work.

Name your zip your last name first letter of first name-lab3.zip (Example: steiners-lab3.zip)