**CYB220 Lab 2 Format string attack**

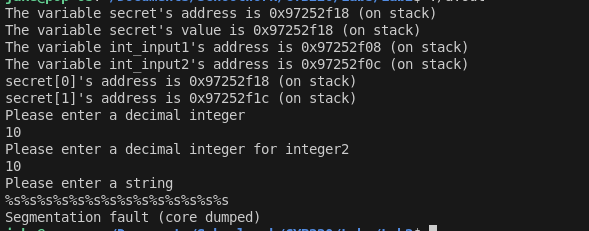
**Due: Friday Sep 13th, 11:59pm.**

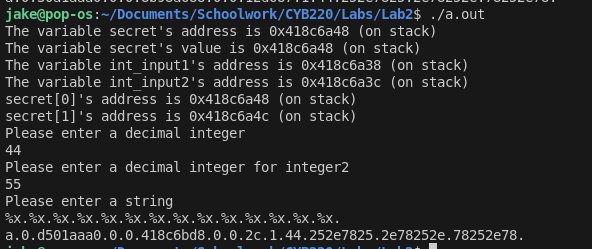
**Turn in: This lab report**

**Points: 30 points**

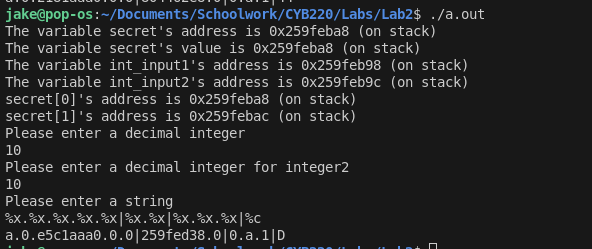
**Name: Jake Gendreau**

(30 points) Write a detailed report about the Format string attack lab. Explain what you have done and what you have observed. Include screenshots for each step.

* (5 points) Task 1 - crash the program
  + By putting %s%s...%s for the string field, the program attempted to fetch n strings from memory, but ended up accessing invalid memory and crashing.
* (5 points) Task 2 - print out 14 values on the stack



* + By using %x. 14 times, the program printed 14 values on the stack, as represented by hex integers
* (10points) Task 3 – Use a string to print out at least one secret values.



* + By using the string that I gave with all of the %x. entries, I was able to identify which address had the 0x44, which was the one with the secret. I was able to then swap out the corresponding %x for a %c to get a character to print. In this case, D.
* (10 points) Conclusion
  + What have you learned from this lab? Anything interesting?
    - I learned that you can use printf() without specifying a format string. When I learned printf(), I was taught that it has to have an input string.
  + What is format string attack? What is the reason of the format string vulnerability? What are the countermeasures?
    - A format string attack occurs when a user input is fed directly into a printf() statement, which can lead to memory access that shouldn't be accessed. It occurs because the printf() statement will read from the stack if no format string is specified. The easiest countermeasure is to first validate input, then ensure that a format string is put into the pritnf() statement by default