**CYB2200 Lab 2 - Format string attack**

**Due: Friday, Sep 19th, 11:59 pm.**

**Turn in: This lab report.**

**Points: 30 points + 10 extra points**

**Objective: Understand format string vulnerability (topic relates to Metacharacters in class). And launch a format string attack to see what value you can reveal in the program.**

**Name: Gabriel Cummings**

Note: Do not make changes on the vul\_prog\_easy.c or vul\_prog\_hard.c.

Note: Give explanations with details. Short answers are not helpful.

(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.

Please use vul\_prog\_easy.c for the following tasks:

* (8 points) Task 1 - crash the program
  + Note: The crash in this lab should be triggered by exploiting the format string vulnerability. Specifically, the print(user\_input); (on line 42). The crash should not be caused by using very long input or other invalid input we tried in lab 1.
  + [Screenshot here] to show your program crashed.

A computer screen with white text

AI-generated content may be incorrect.

* + Please explain in detail why the program crashed and which exact %s triggered the crash.

My program fails on the second %s.  
When I use one %s, it returns the user-input variable, which in this case is the %s itself.  
When I use two, the program tries to access a place in memory only available to the operating system (memory address 30), which throws the “segmentation fault” error.

* (5 points) Task 2 - print out 20 values on the stack
  + [Screenshot here]  
      
    A screenshot of a computer

    AI-generated content may be incorrect.
  + Please explain what you did and why.  
    I used %x, which prints out the hexadecimal value of the memory address stored on the stack.  
    I input %x 20 times, using a period to separate each %x.  
    This printed out the top 20 memory addresses from the stack.  
    You’ll notice the 2nd one is “30”, which proves what I said in Task 1 about %s attempting to access a memory value it doesn’t have permission to access.
* (10 points) Task 3 – Format string attack: Use a format tag to print out at least one secret value. The screenshot should have the letter ‘C,’ a letter ‘S,’ or both letters ‘C’ and ‘S’ shown.
  + [Screenshot here]

A computer screen shot of a computer

AI-generated content may be incorrect.

* + Please explain what you did and why.  
    I used two %x’s before using a bunch of %s’s in my input.  
    This avoided the issue where my program tried to get the information at the 2nd memory address (“30”), and then printed out the string values of all the other memory addresses on the stack.  
    This revealed “C”, one of the secret characters mentioned in the description of Task 3.
* (7 points) Conclusion
  + What have you learned from this lab? Anything interesting?  
      
    Well, I’ll certainly be sure to use input validation everywhere I possibly can, and not make the mistake of simply printing out the user’s input without checking it.  
    I don’t want my information or my program’s secrets stolen!
  + What is a format string attack? What is the reason the leads to the format string vulnerability? What are the countermeasures?  
      
    A format string attack is when a user inputs a “format tag” as their input, and the program makes the mistake of trying to print that out directly.  
    The program then tries to “execute” the user input so to speak, and tries to grab whatever information is stored on the stack, formatted based on the format tag used.  
    For instance, %s attempts to return the string ((const) (unsigned) char \*) stored at a memory address, and %x returns the hexadecimal (unsigned int) of whatever’s on the stack (so the memory address in this case).  
    This can give the user information they definitely aren’t supposed to have.

--------------------- Extra points Task --------------------

Note: this task is not required. However, finishing the task here will get you extra points.

Please use vul\_prog\_hard.c for this task:

Hint: This program uses dynamic memory, memory allocated with malloc(), which is stored on the heap, not on the stack. Exploiting a format string vulnerability only shows the values on the stack. How can we print the value on the heap? Your goal is to find out where those two letters are located on the heap and grab/print the value on the heap.

* (Not required) (10 pts) Task 4 – Format string attack: Use format tags to print out at least one secret value.

Assume you don’t know the two letters. Manually writing the value to the stack doesn’t count. In other words, you cannot write 0x45 or 0x56 to the stack and print them.

* + (5 pts) The screenshot shows the letter ‘E’
  + [Screenshot here]  
      
    A computer screen shot of a black screen

    AI-generated content may be incorrect.
  + Please explain what you did and why.  
    I input %x many times, followed by a %s as the end.  
    I started with one %x, then a %s, then I did it again, adding a %x to the end right before the %s, and I did this over and over.  
    Eventually, I ended up printing out one of the secret values!
  + (5 pts) The screenshot shows both letters ‘E’ and ‘V’.
  + [Screenshot here]
  + Please explain what you did and why.