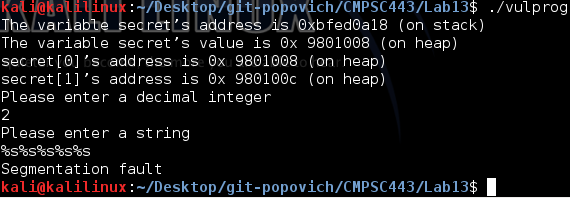
**2.1 Task 1: Exploit the Vulnerability**

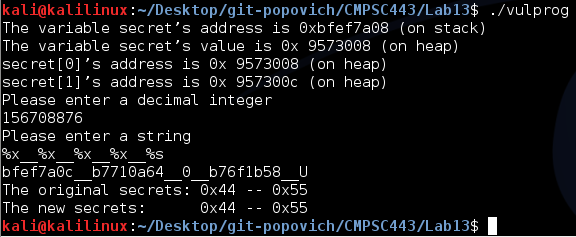
For this task, we will be exploiting printf vulnerabilities. Our goals are to

* crash the program
* print out the value of secret[1]
* modify the value of secret[1]
* and modify the secret[1] value to a pre-determined value.

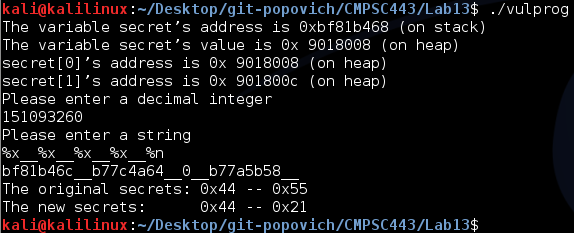
To start, we will crash the program by telling printf that we are sending it multiple arguments when in reality, we are not. This will cause it to take arguments off the stack and examine those locations in memory. Eventually, one of the locations it’s bound to examine will be off limits or an invalid location in memory, crashing the program. A screenshot of this happening is below:



Next, we plan on printing out the value of secret[1]. Because the program prints out secret[1]’s address for us (0x94b800c), we can use that as our format string. We will convert the address (0x94b800c) to decimal and get 155942920. We will use this number for our decimal integer. By using the format string “%x\_\_%x\_\_%x\_\_%x\_\_%s”, the program will print out addresses up the stack until it hits the decimal address that we entered. This address is converted to hex and printed out as a string. The program prints out “U” which is 0x55 = secret[1]! A screenshot is below:



Next, we will modify the value of secret[1]. This is very similar to the last program only we will change the “%s” in the format string to “%n”. This will take the address that it finds and write the current number of bytes written in the current call of printf. A screenshot of the program is below. Notice that it changes the value to 0x21 = 33 bytes printed, which checks out.



Finally, to write any value I want, it will be the number of bytes printed out by printf. Because we are printing out the addresses as we go up the stack, the smallest value we can print is 25 = 0x19. I will be printing out a 0x33 = 51, see the screenshot below:

