Matthew Kachar

COSC4931

2/5/16

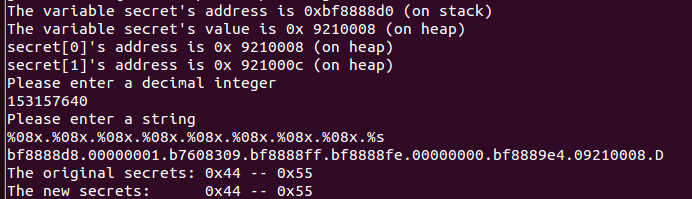
Homework 2 Format String Vulnerability Lab

Lab Objective:

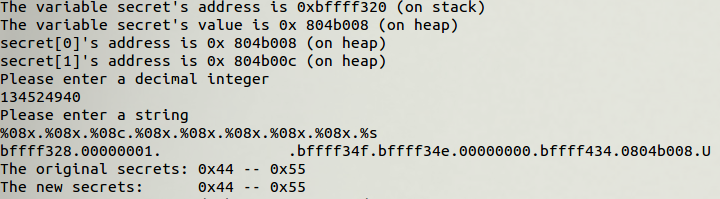
Exploit vulnerability of format-string function and develop protection scheme.

The first exercise is to exploit the vulnerability in the format-string function in the following ways; crash the program, print out the value of secret[1], modify the value of secret[1], and modify the value of secret[1] to a pre-determined value.

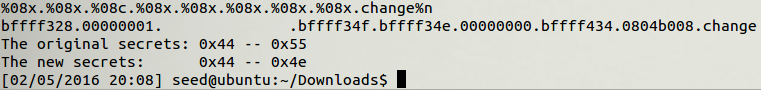
An easy way to crash the program is to input the string format specifier %s multiple times and the program will reach a segmentation fault trying to access unreachable memory.



To find the value of secret[1] the process to do so is to input the decimal value of secret[1]’s memory location then use %08x specifiers to print out the hexadecimal address before the string specifier which prints out secret char ‘U’



To change the value of secret[1] access the address the same way as state using the hex specifier but instead of the string specifier at the end use the %n specifier which will adjust the value of by a number of characters.

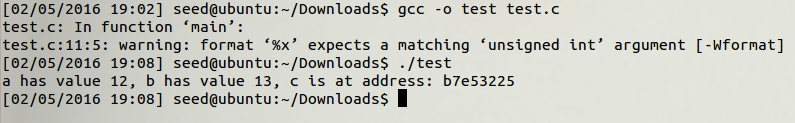


As shown above the value of secret[1] has been changed.

The quickest easiest fix to the code is to add the string modifier to the printf( ) function as to print simply what has been entered without performing any specifier commands

When the memory location is randomized it becomes much more difficult to assume the location of secret[1] and is much harder concatenate the address in the printf( ) function. For the second part of this section adds another level of difficulty to sort out the string address in the *mystring* file.

Although if enough subsequent addresses are correctly passed through vul\_prog.c then the *mystring* file can basically print out the entire stack. However it is much more difficult to dynamically route out the stack pointer using the file, and in fact I had more segmentation faults then successful attempts.



As for the miss-match potential, the compiler will proceed, but prompt a warning. The printf( ) function also seems to have no set method for fetching the *c* address call. In other words, one of the other variables’ address is called, but there is no telling which one. Sometimes it is *a* and sometimes it is *b* which is called by the %08x specifier.

Careful considerations must be made with the printf( ) function to avoid unintended consequences which can occur from implementing the albeit useful, yet easily exploited function. For instance, adding certain specifiers can stop memory addresses from being printed and methods for direct addressing can change the way in which things are stored in memory to something less predictable.