Scott Christensen

12/05/16

C.S. 465

Project #8: Extra Credit Option 7

Learn about format string vulnerabilities and demonstrate how they work:

You can use C’s printf function to read data from the stack “%x”, read character strings from the process’s memory “%s”, and write an integer to locations in the process’s memory “%n”.

Here is the original C program:

int main (int argc, char \*\*argv)

{

char buf [100];

int x = 1 ;

snprintf ( buf, sizeof buf, argv [1] ) ;

buf [ sizeof buf -1 ] = 0;

printf ( “Buffer size is: (%d) \nData input: %s \n” , strlen (buf) , buf ) ;

printf ( “X equals: %d/ in hex: %#x\nMemory address for x: (%p) \n” , x, x, &x) ;

return 0 ;

}

If the format string parameter “%x %x” is inserted in the input string, it looks like so:

./formattest “Bob %x %x”

When the format function parses the argument, the output will display the name Bob, but instead of showing the %x string, the application show the contents of a memory address. Like so:

Buffer size is (14)

Data input : Bob bffff 8740

X equals: 1/ in hex: 0x1

Memory address for x (0xbffff73c)

Using this vulnerability, one can smash the stack or does a buffer overflow attacks.