Isai Mercado Oliveros
misaie
November 3, 2015

Format String Vulnerabilities

C functions that print format strings such as %s, %d, etc have vulnerabilities. The vulnerability is that when the C print function is called with one string format, it will spec the printing variable to be positioned in the stack above the print function, so that the print function grabs that variables formats the string representation according to the format string and displays it in the terminal. The problem comes when the print function is given a string format, and it is not given the variable to print, then the print function would print the value in the stack that is right above the print function. For example,

```
char* variable = “hello world”;
printf(“%s”, variable);

// output is hello world
```

If the code just calls the printf(“%s”) without any variable it will try to print whatever is above it, and it might crash, but if you print the binary value as an hexadecimal string representation, such as printf(“%x”) it will print the binary value of above position of the stack as a hexadecimal value. For example,

```
int a = 1;
printf(“%x”);

// output would be something like 00000001 which is the value of a
```

Moreover, the problem increases when an attacker finds that print functions are being used, and as input he enters two hundred “%x\n” such as “%x \n%x \n%x \n%x \n%x \n%x \n%x \n%x \n...”into the print function. The print function will read every single value in the stack up to the place 200 and it will print 200 hexadecimal representations of what is in the stack, which is giving the whole stack to the attacker. If the attacker knows the stack, he can calculate return addresses, and he can reverse engineer to find how to open a shell, and continue exploiting the system.

Showing the whole stack is very easy, if the attacker uses a little bash script to generate all the “%x /n” that he needs. For example,

```
Perl -E ‘say “%x /n”x200’

Or

for i in {001...200}; do echo -n “$i = “ ; ./programToBeHacked
“%$i$p”; echo; done
```