

## HW #8: Buffer Overflow

CS 392/681: Computer Security  
Fall 2006

**[100pts] Demos Due starting 11am on 12/18/06**

Consider the following program “vuln.c”:

```
//BEGIN:A program to illustrate buffer overflow
//vulnerability

void func(char *str) {
    char buffer[8];
    int *ret;
    strcpy(buffer,str);
}

int main(int argc, char **argv) {
    int x;
    x = 0;
    func(argv[1]);
    x = 1;
    printf("%d\n",x);
}

//END
```

The program is very similar to one of the programs we studied in the class. Clearly, it has a static buffer overflow vulnerability. In this exercise, you have to modify the normal execution flow of the program in such a manner that the **instruction {x=1;} is skipped**, i.e., finally program prints out the value of x as 0 and not 1. You have to achieve this in two different ways:

1. **[50points]** Assume that you have write access to the program, i.e., you can modify the source code. Modify the function func(.) in so that the address to which the program returns after executing function(.) is changed in such a manner that the instruction {x=1;} is skipped. Use the pointer \*ret defined in funct() to modify the return address appropriately
2. **[50 points]** Assume that you don't have write access to the program, but you only have an executable access. Now, you have to exploit the buffer overflow vulnerability and execute the program by passing an argument "argv[1]" in such a manner that the return address is modified so that the instruction {x=1;} is skipped. You can fill up the buffer with NOP instruction (opcode 0x90) wherever needed. You can write an exploit program that runs the above program ("vuln.c") with an arbitrary input of your choice (use execl() function to do so), for testing.

## Instructions

You can use the objdump utility to get a map of the code section of the executable file. The command is "objdump -d <executable\_file\_name>". This will list all the address locations, opcodes and assembly code of the instructions, for all functions the program calls. For the exercise, you are mainly interested in the main function and the address locations of its various instructions.

## Demo

You have to demo and explain how you did your exercise to the TA and/or to me.