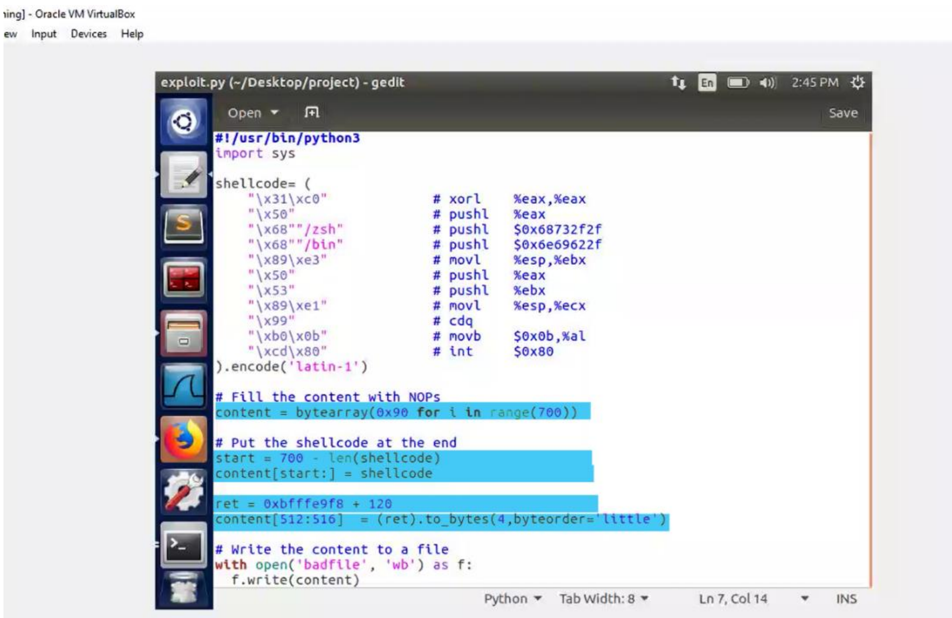
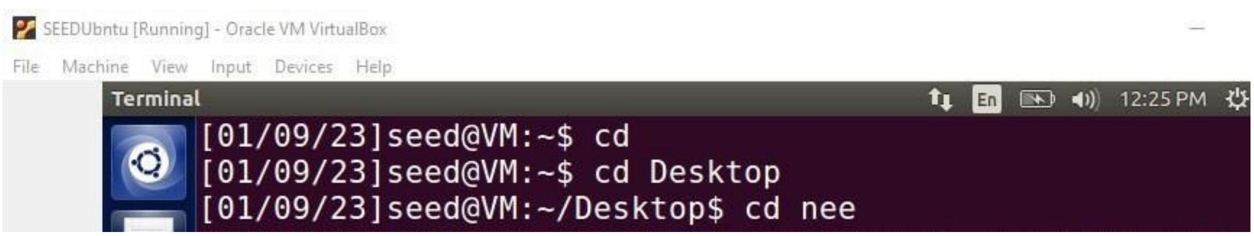
- Task one:

  - First, I have filled in the python class what it was missing and changing the buffer size to (700).



- Now, in the terminal first we have to change directory and go to the place the files are there so we can compile and run.



- Then, the first step we will run (sudo sysctl -w kernel.randomize_va_space=0) and turn off the randomization.

When set to zero it means the address space is not randomized, and the base address of the buffer remains, so the variable’s address will always be the same.

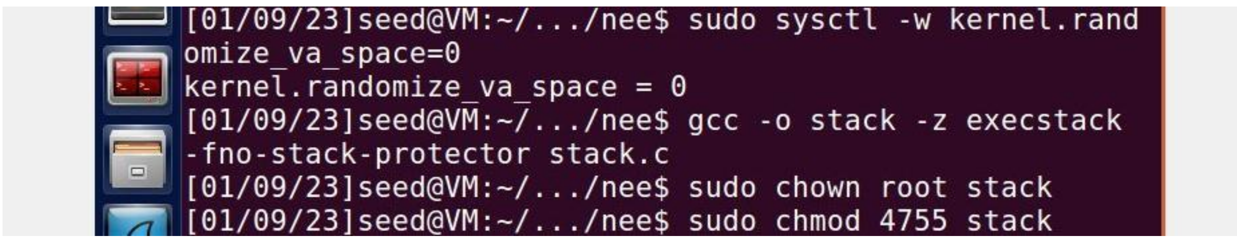
- After that we will compile (gcc -o stack -z execstack -fno-stack-protector stack.c)

(-z execstack) to sets the stack as executable as we need

(-fno-stack-protector) to disabled stack guard protection

- Now we will change the owner of the executable stack to root by (sudo chown root stack)

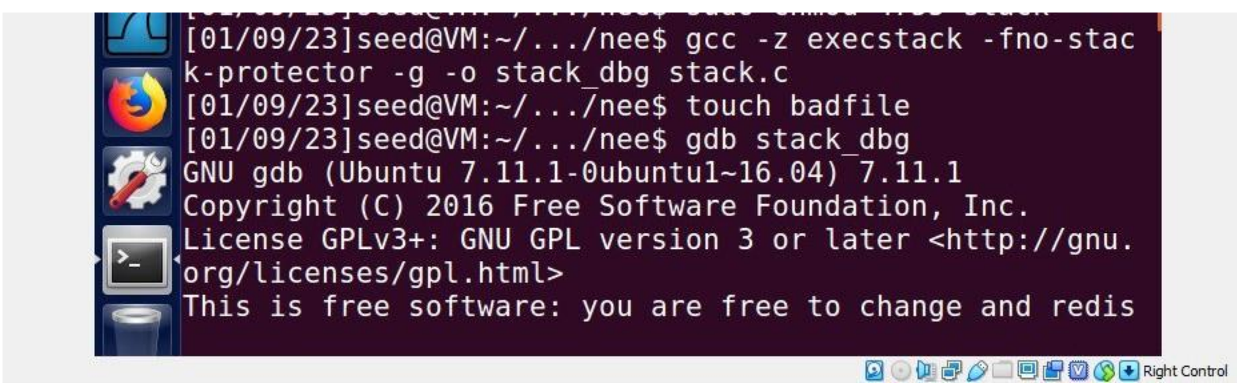
-Then we will change the mod to 4755 by writing (sudo chmod 4755 stack) to make the stack editable and executable, and the mod after that will be read and write execute.



- Now we have to find the distance between the base of the buffer and the return address by using gdb debugging tool.
- We will write (gcc *–z execstack –fno-stack-protector* –g –o stack_dbg stack.c)

  (–o stack_dbg) to make stack debug.

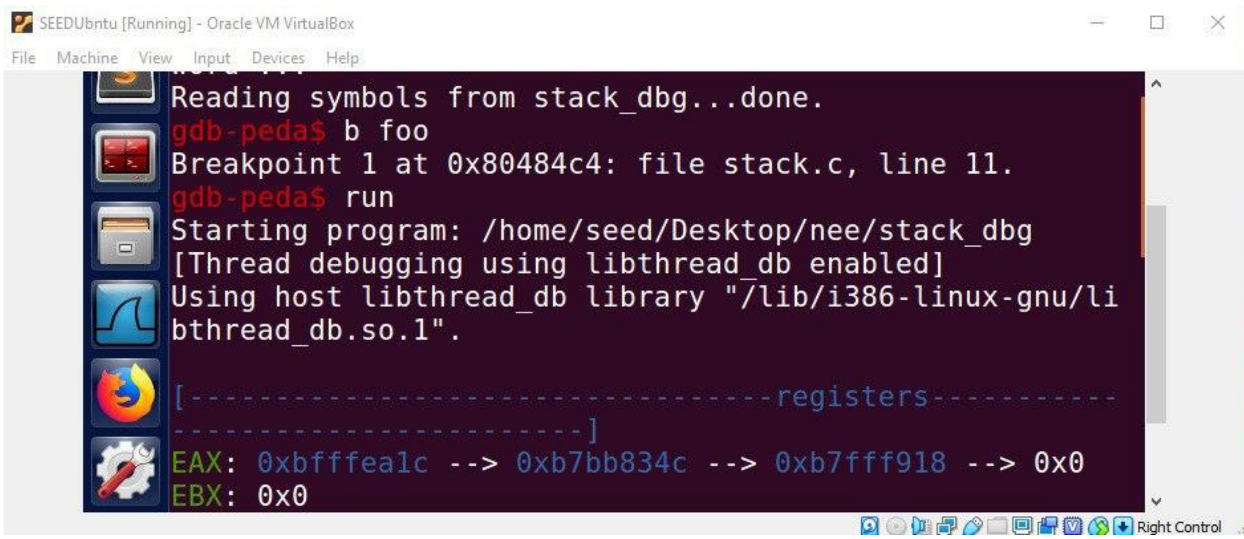
- Then, we will run (touch badfile), this will create badfile and open it so we can write on it.
- Now we will open debugging tool by (gdb stack_dbg)



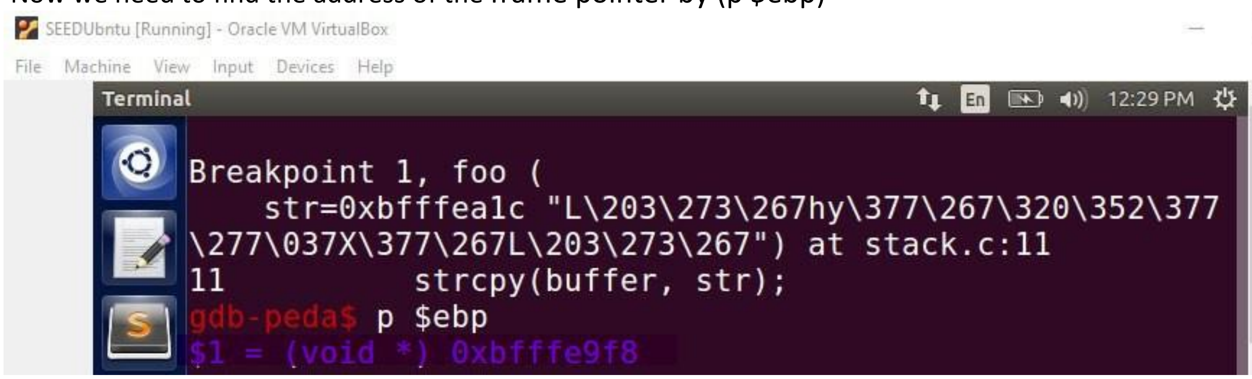
- Here inside the debug, we will set a break point to the function by (b foo)

  foo() is the name of our function, and the program will stop inside the foo function.

- And then (run)



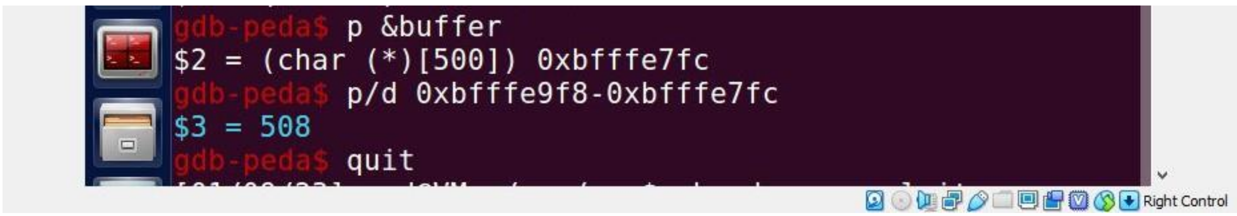
- Now we need to find the address of the frame pointer by (p $ebp)



- Also, we need to find the address of the buffer by (p &buffer)

- Now we need to find the distance between the base of the buffer and the return address by (p/d 0xbfffe9f8 – 0xbfffe7fc)

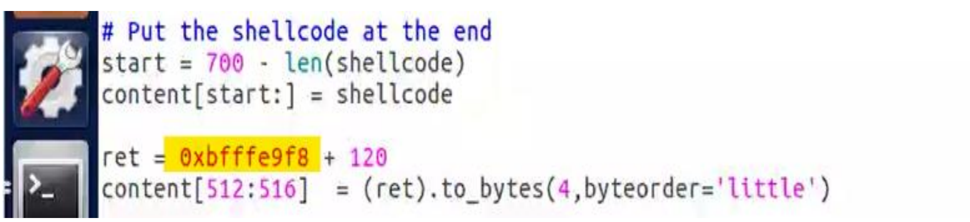
  (p/d) To give us unsigned integer



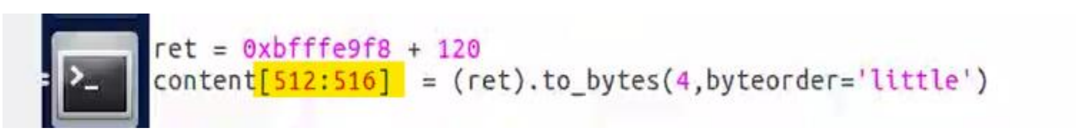
We can figure out the return address is = 508 + 4 = 512, and the return address will be between 512 and 516.

Now we will adjust on the python class:

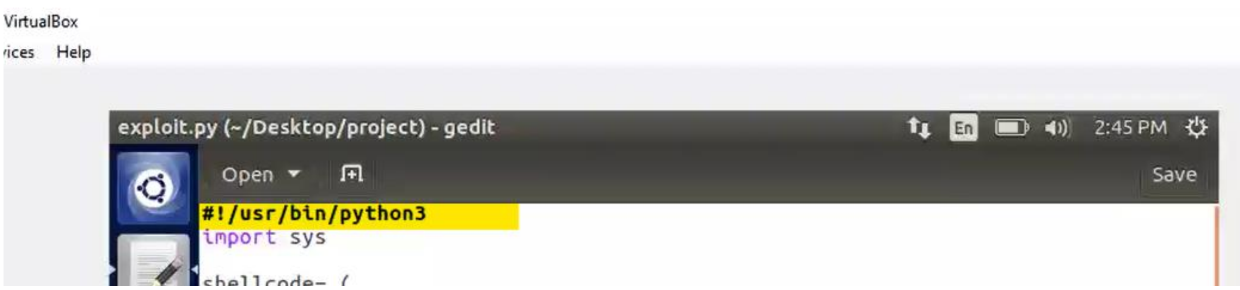
- First, we will change the ret to be the same address as the frame pointer



- Second, we will place the return address byte by byte from content [512] to content [516].

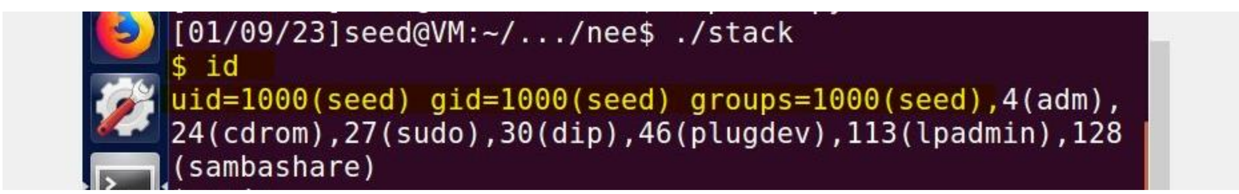


- After that I will make the script executable by adding (#!/usr/bin/python3) to the top of the script



- And in the terminal I will make the file executable with (chmod u+x exploit.py) and before running (./exploit.py) I will first remove the old badfile because the exploit will create a new file.

- Now we will run the vulnerable program by "./stack", to be able to get a shell.

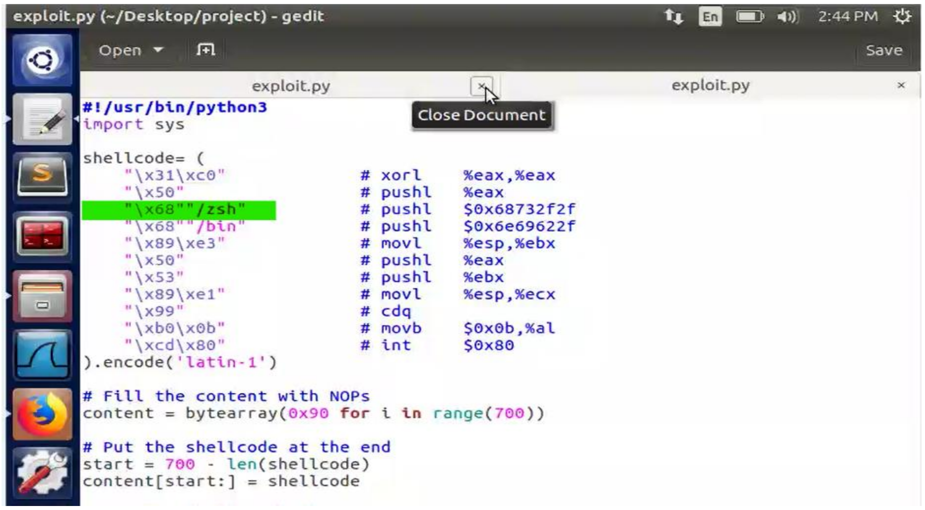


Got the shell.

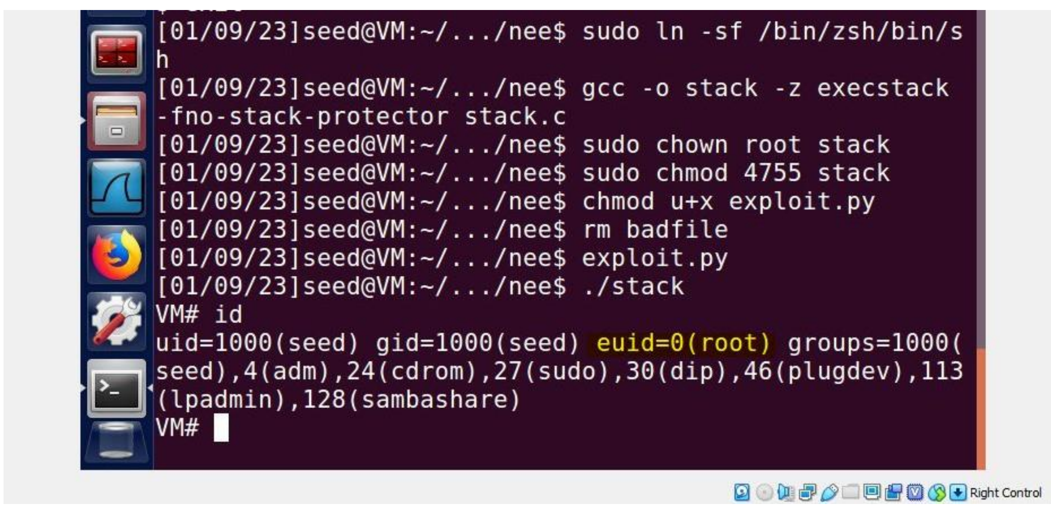
- Task tow:

We have to find a way to obtain root shell.

- We will change on the shellcode.



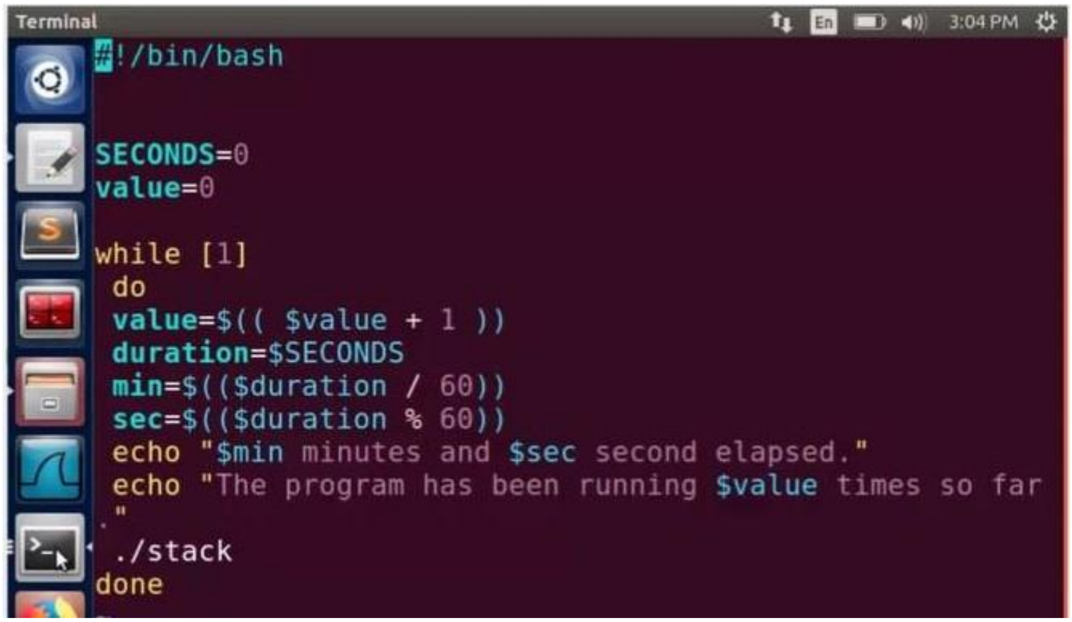
- Then we will run(sudo ln -sf /bin/zsh/bin/sh), and then the same command on task one to be able to get root.



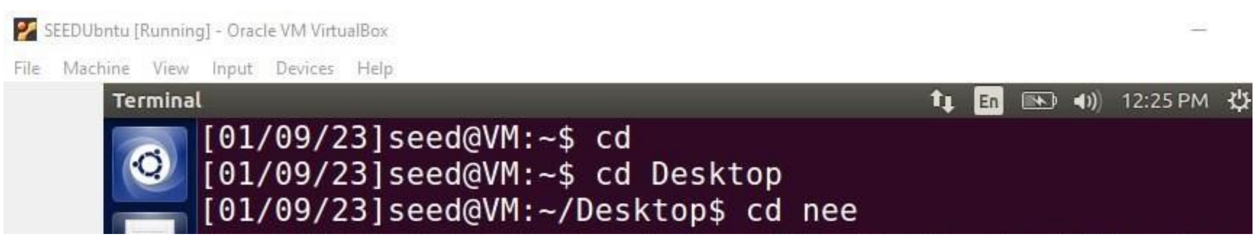
As we see we got the root shell.

- Task three:

- On 32-bit Linux machines, stacks only have 19 bits of entropy, which means the stack base address can have 2^19 = 524, 288 possibilities. This number is not that high and can be exhausted easily with the brute-force approach.
- We have defeated the address randomization countermeasure on our 32- bit VM.
- First, we will write a shell that opens our code in infinite loop and try it out many times against randomly assigned stack address.



- In the terminal we have to change directory and go to the place the files are there so we can compile and run.



- Now, we will turn on the Ubuntu's address randomization using this command: (sudo /sbin/sysctl -w kernel.randomize_va_space=2)

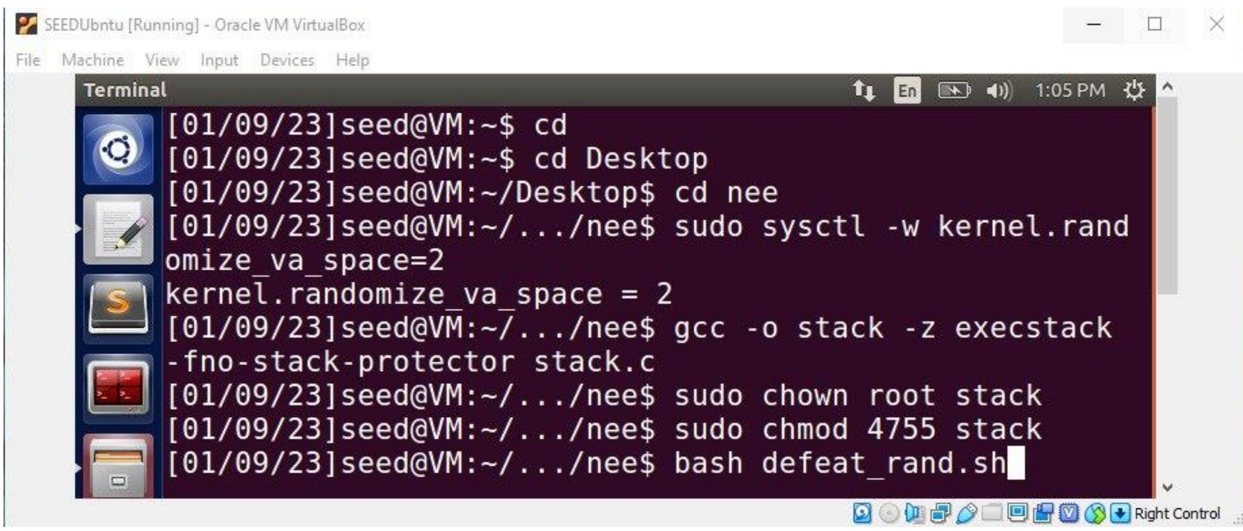
When set to 2, both stack and heap memory address is randomized.

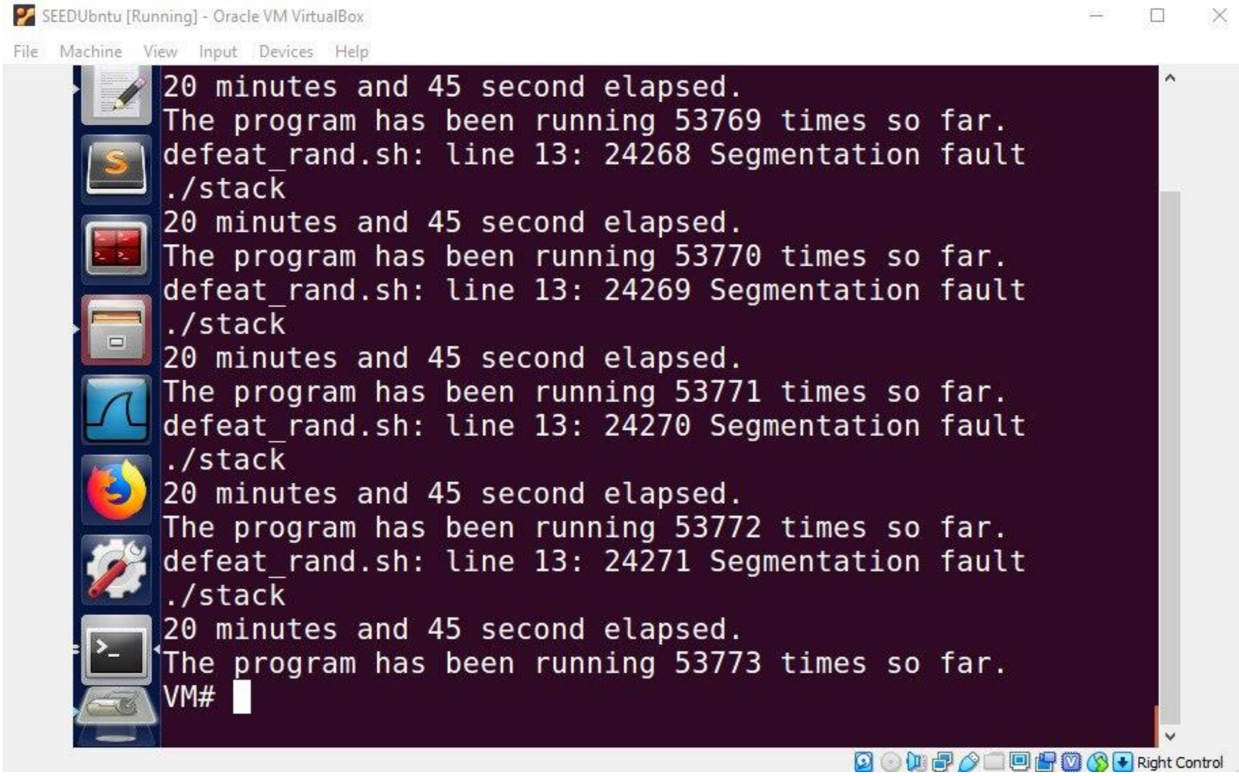
- After that we will compile (gcc -o stack -z execstack -fno-stack-protector stack.c)

(-z execstack) to sets the stack as executable as we need

(-fno-stack-protector) to disabled stack guard protection

- Now we will change the owner of the executable stack to root by (sudo chown root stack)
- Then we will change the mod to 4755 by writing (sudo chmod 4755 stack) to make the stack editable and executable, and the mod after that will be read and write execute.
- And to run the shell we will use (.sh), so we will use this command (bash defeat_rand.sh)





After 20 minutes and 45 second we got the root shell.