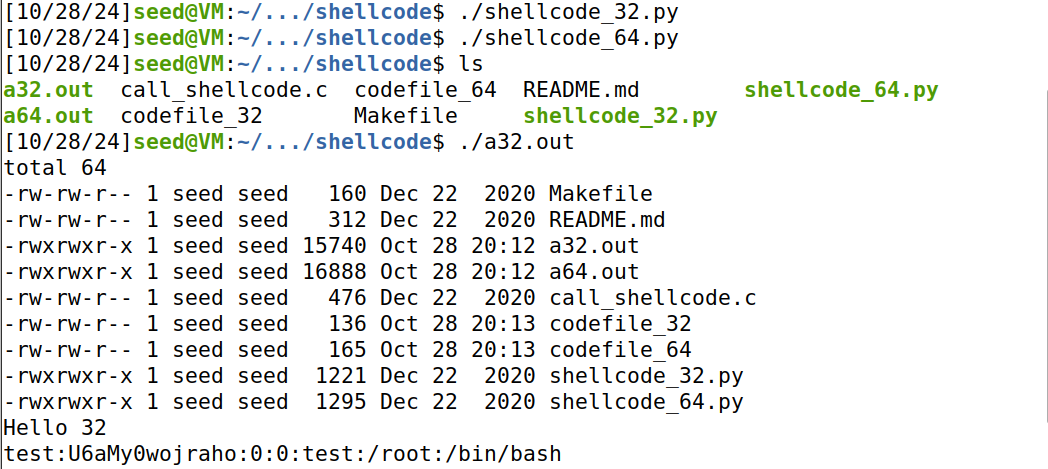
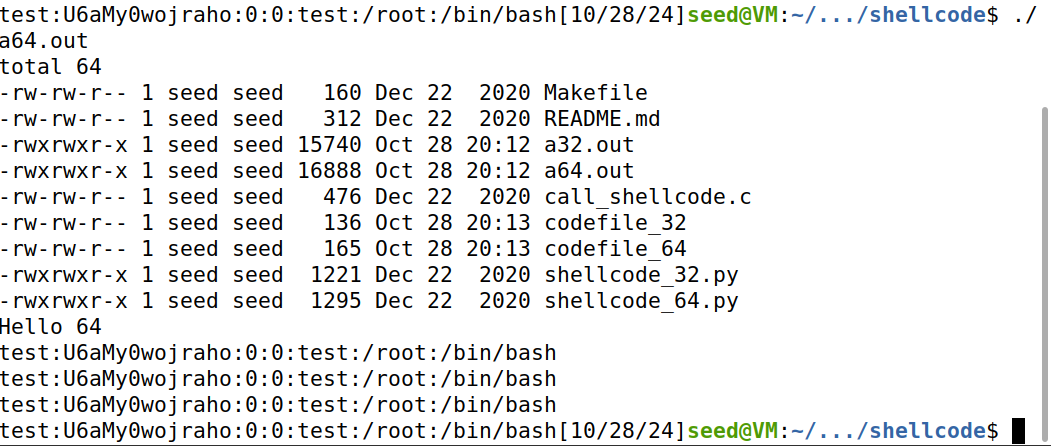
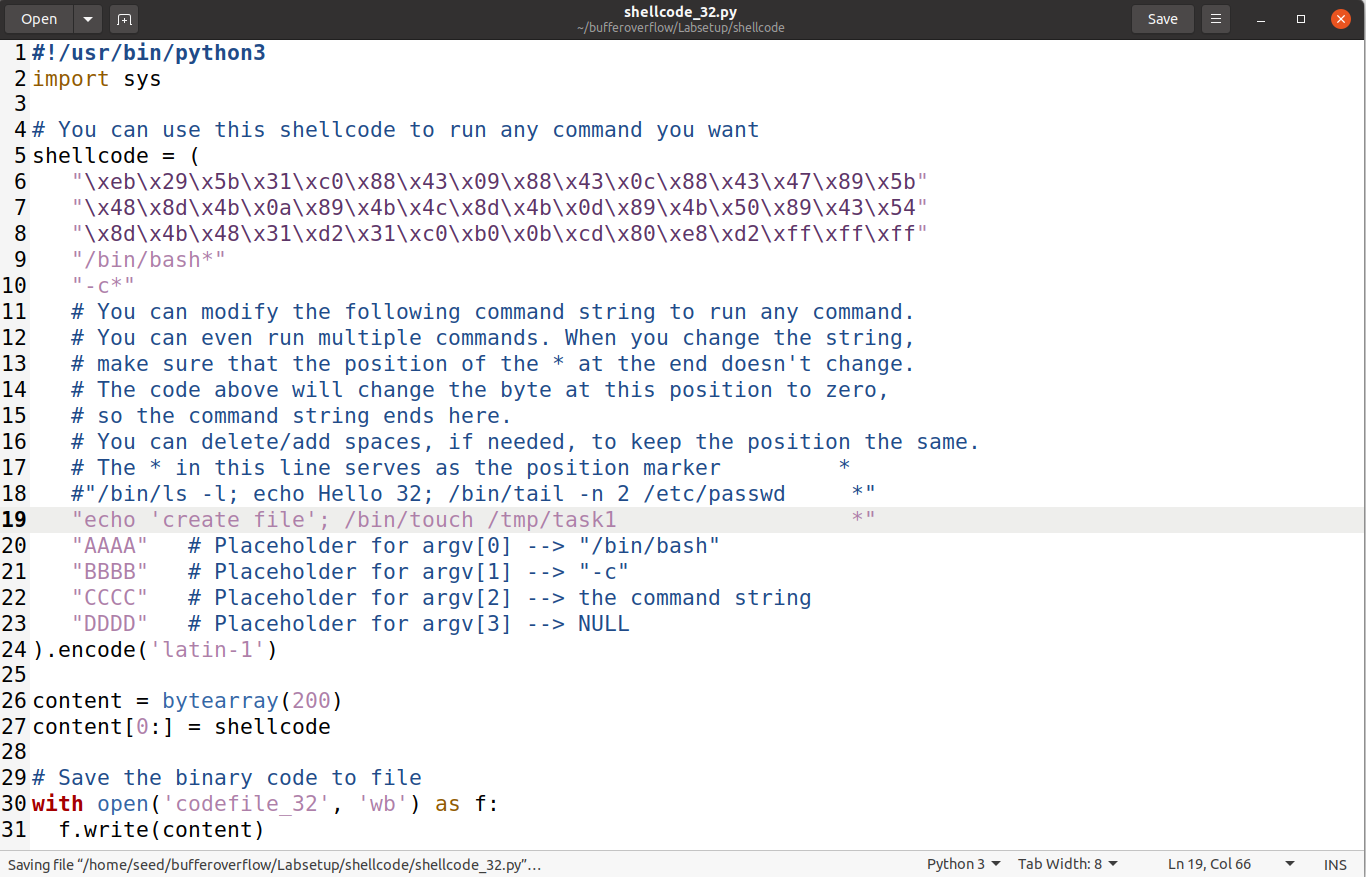
Name:Donghao Li

SU email: [dli106@syr.edu](mailto:dli106@syr.edu)

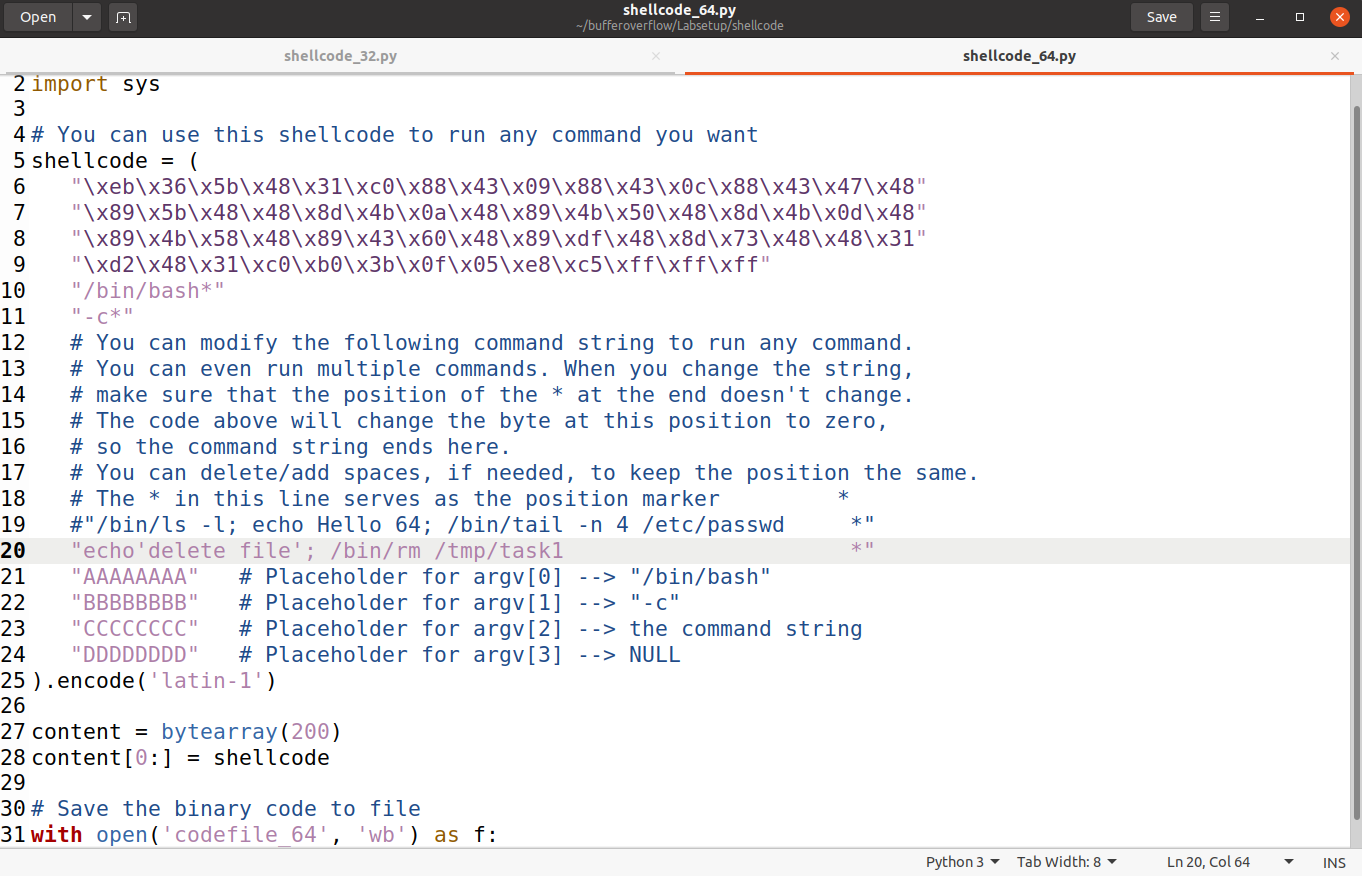
**Task 1: Get Familiar with the Shellcode**

****









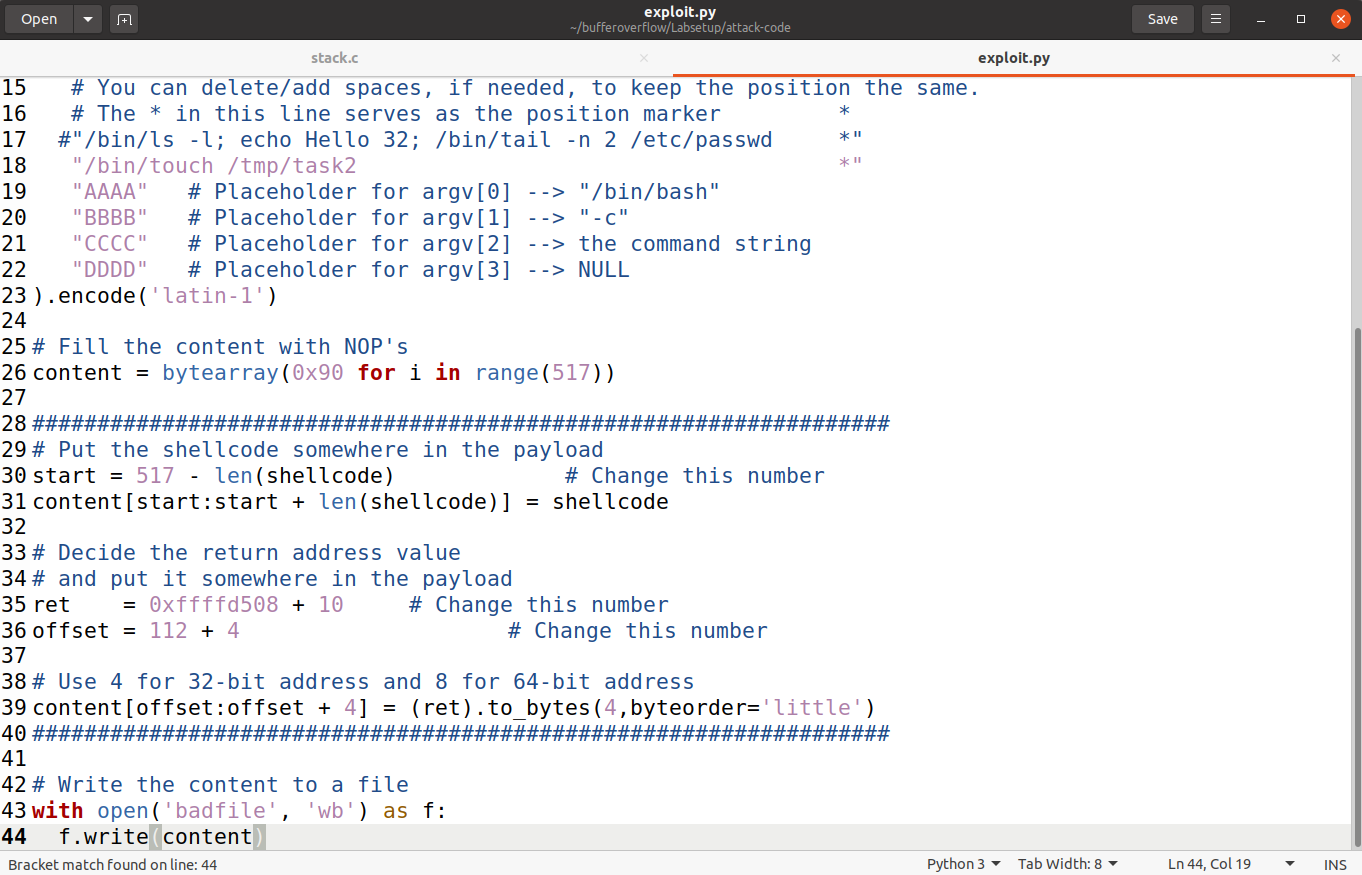


as you can see, i use 32bits python code to create a file task1 in tmp folder and then use 64bit python code to delete it.

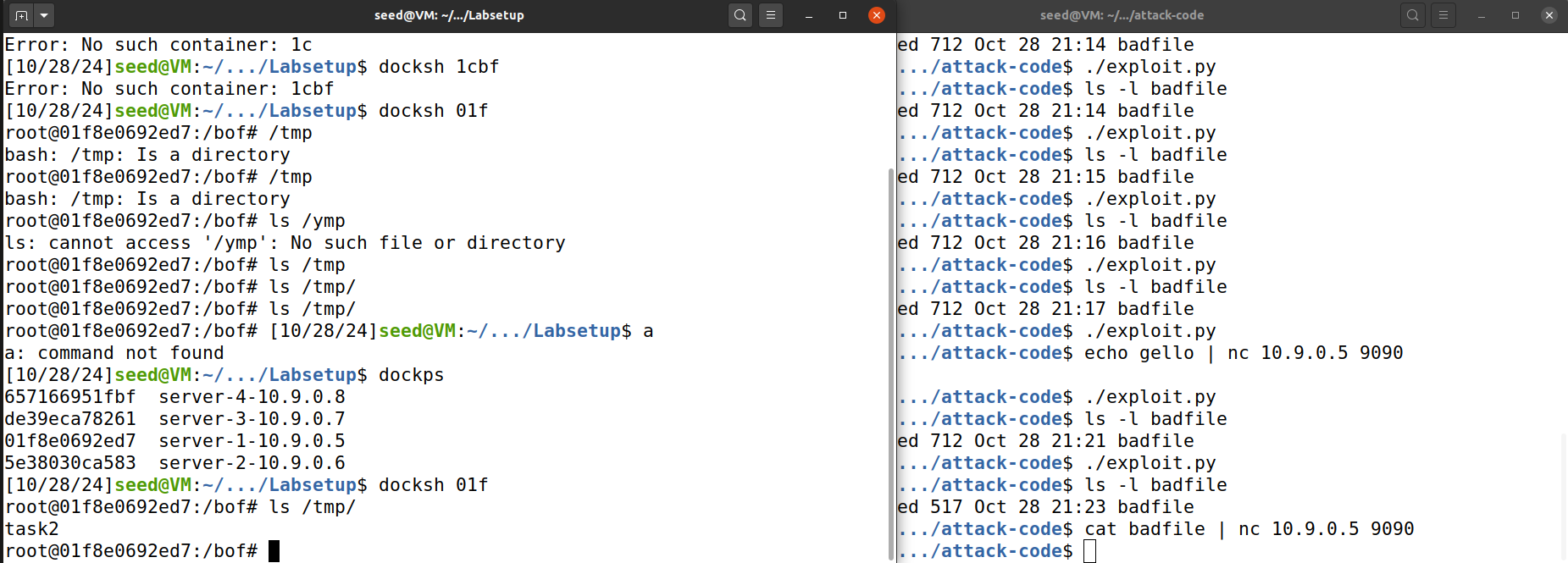
**Task 2: Level-1 Attack**



I echo a message to the server 1 and i can see its reflections.

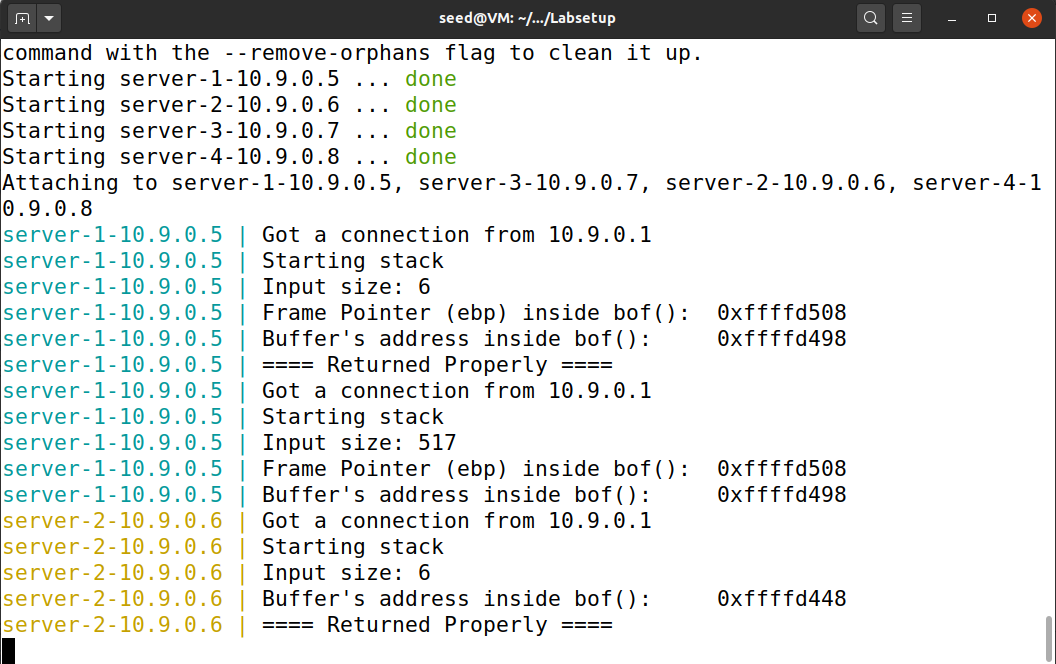


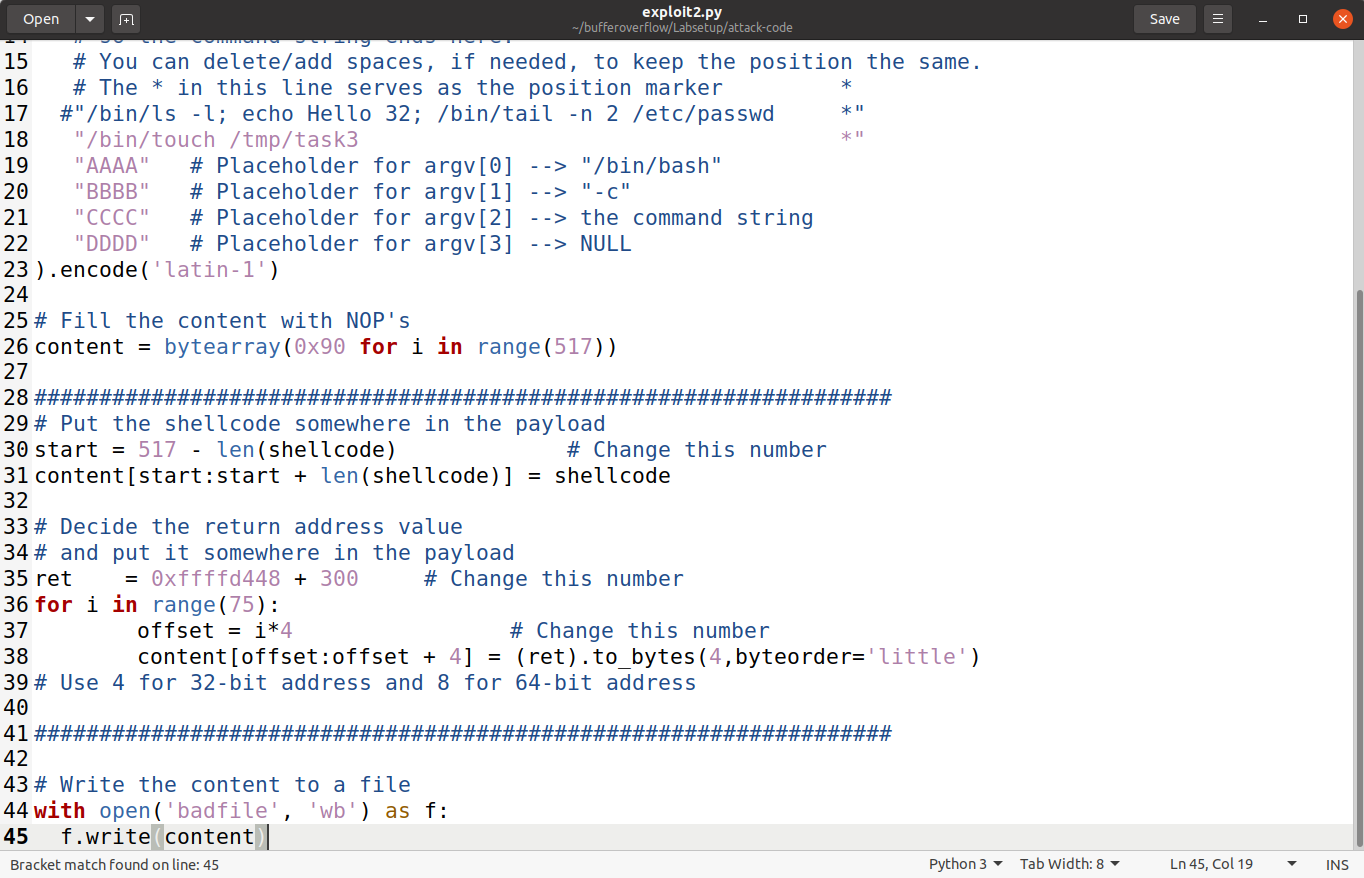
how i edit the exploit file is that i add a create file command to test if the attack is works. Then i set the start position be amount of the content minus the length of shellcode. Then I set the return be the ebp i get from the server page add 20. The offset is ebp minus buffer +4. Because 508 is the ebp and minus buffer 498 would be the esp which is the start of the ret then add four bit to make sure in actually in the range.

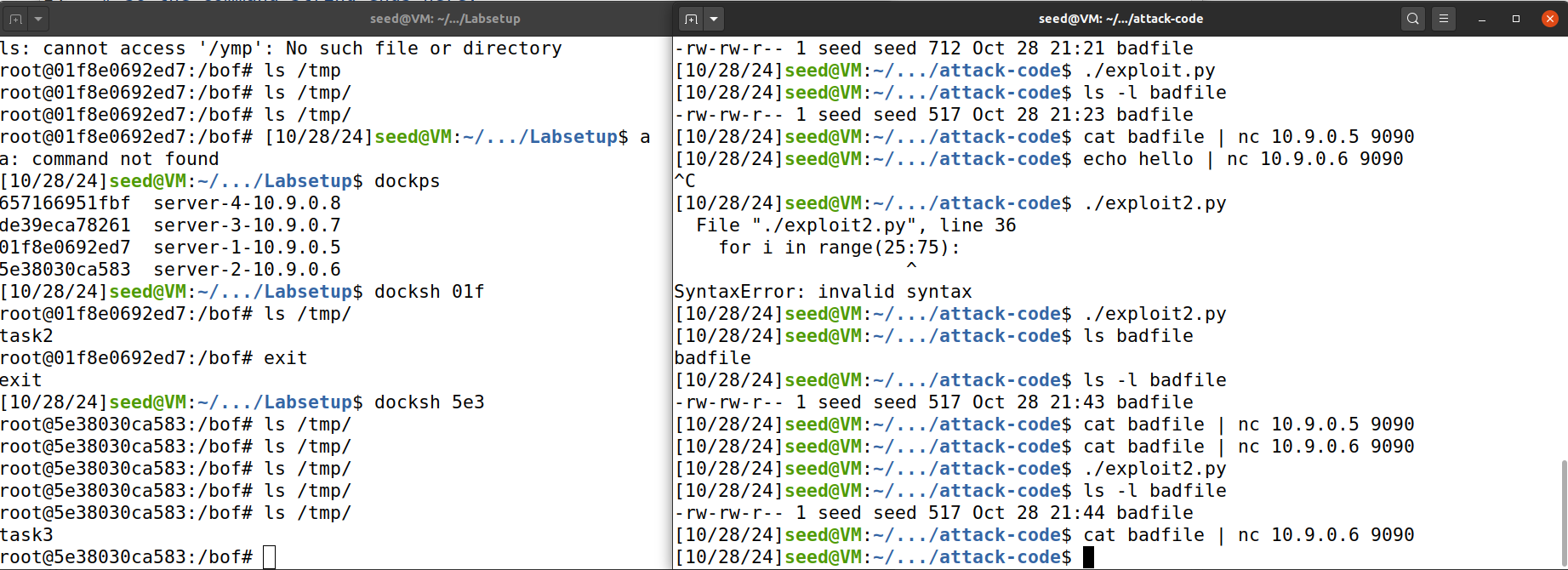


here as I cat the badfile to server1 i could find /tmp/task2 in my container.

**Task 3: Level-2 Attack**

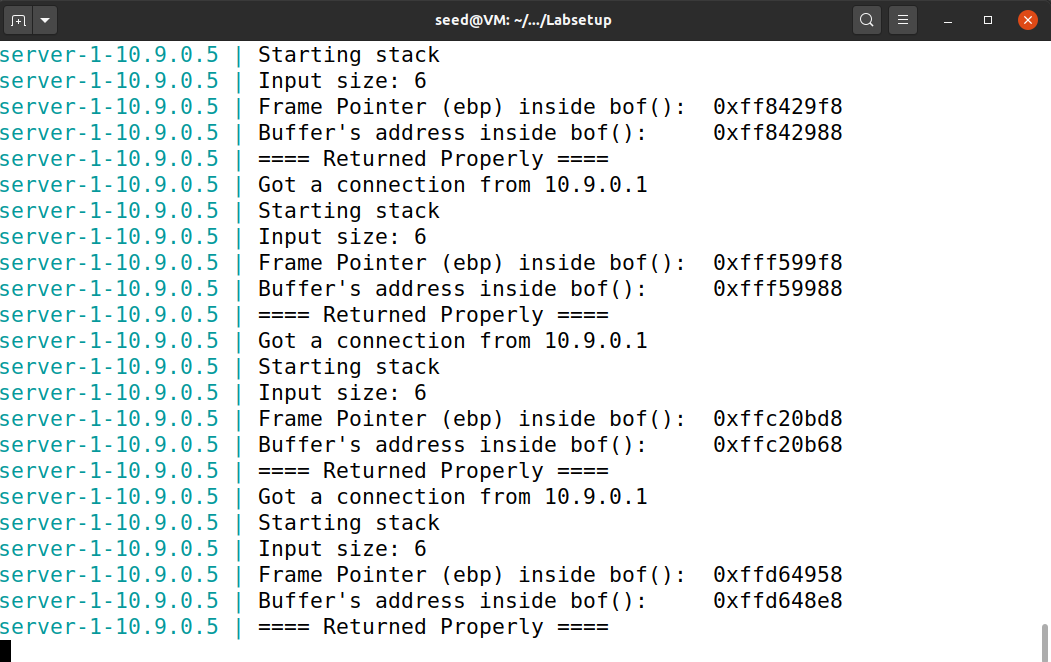


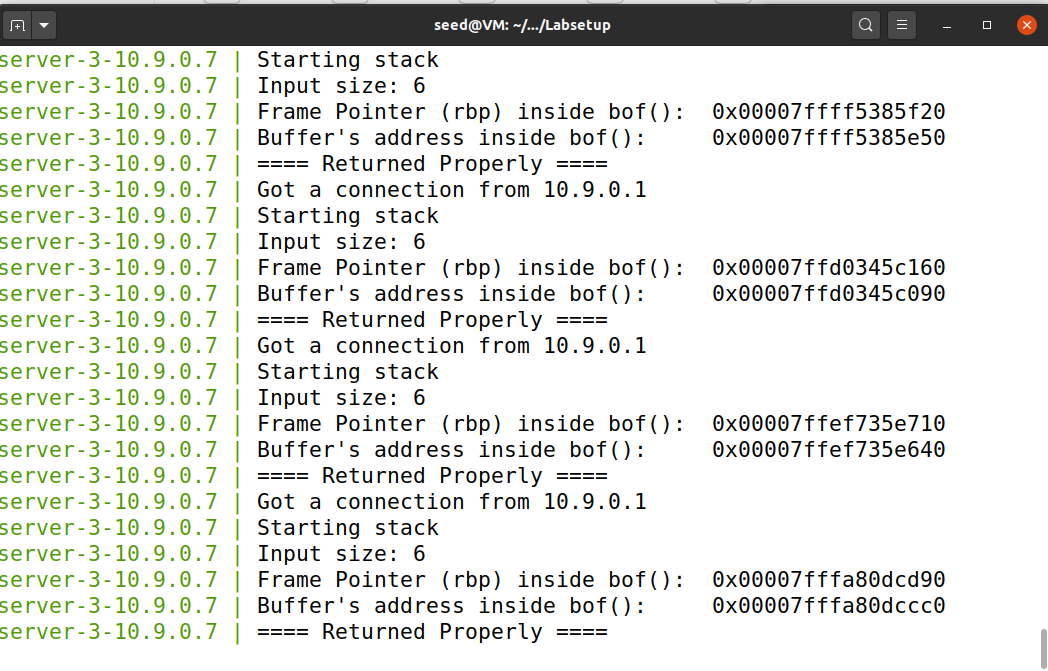




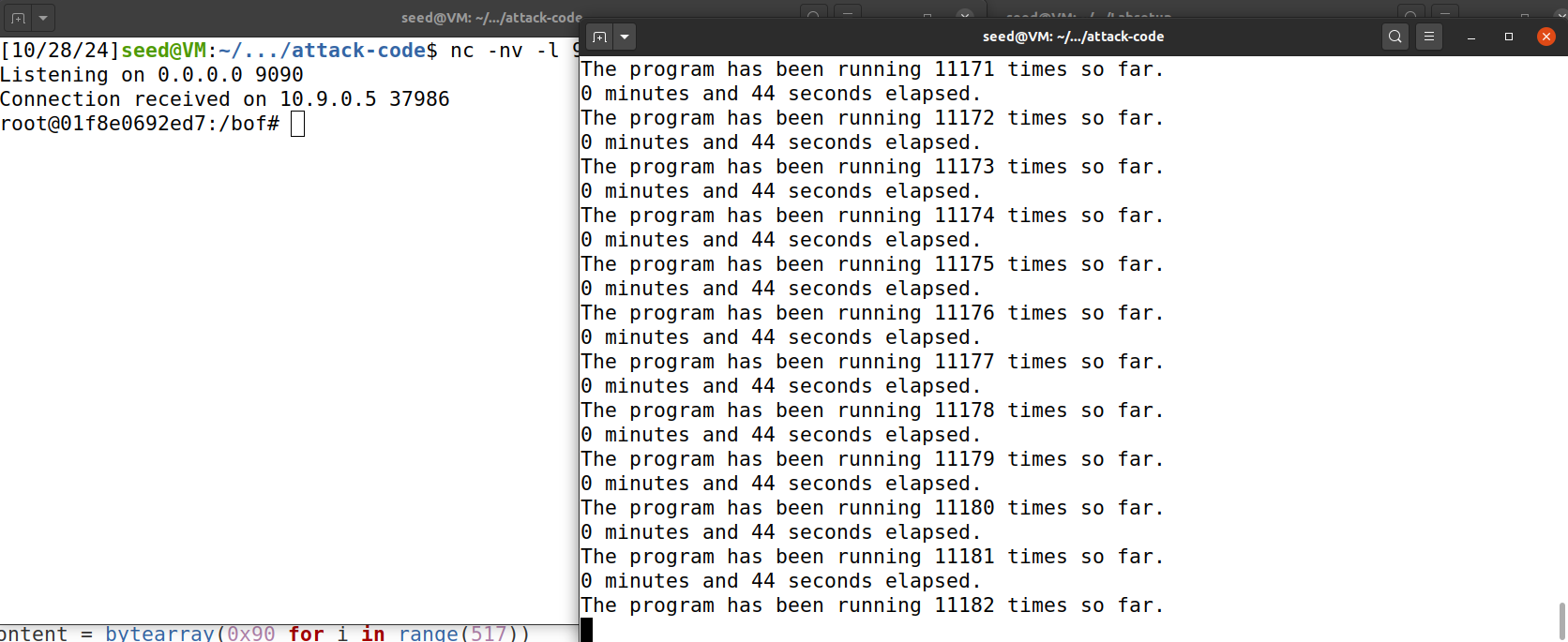
i stays the start the same, and then set up the return value the the buffer size plus 300 to make it closer to the shellcode, and i use a for loop to iteratively set the offset from 0 to 75 which make sure every s[ace from 100 to 300 being point to our malicious code.

**Task 6: Experimenting with the Address Randomization**



****

for both server one and three the ebp and buffer changes after a connection if finished, this may cause the exploit attack takes too much time to update the new ebp and buffer for attacking.



As you can see I add up the reverse shell command in my exploit.py and run it make sure i use the address of my local host, then open a local host wait for the result, then run brute\_force.sh to start the attack. After a 44 second elapsed, the program stop and i successfully reverse the shell.