**LAB5 Part3**

**Task1: Get Familiar with the Shellcode**

**Turn off address space randomization**

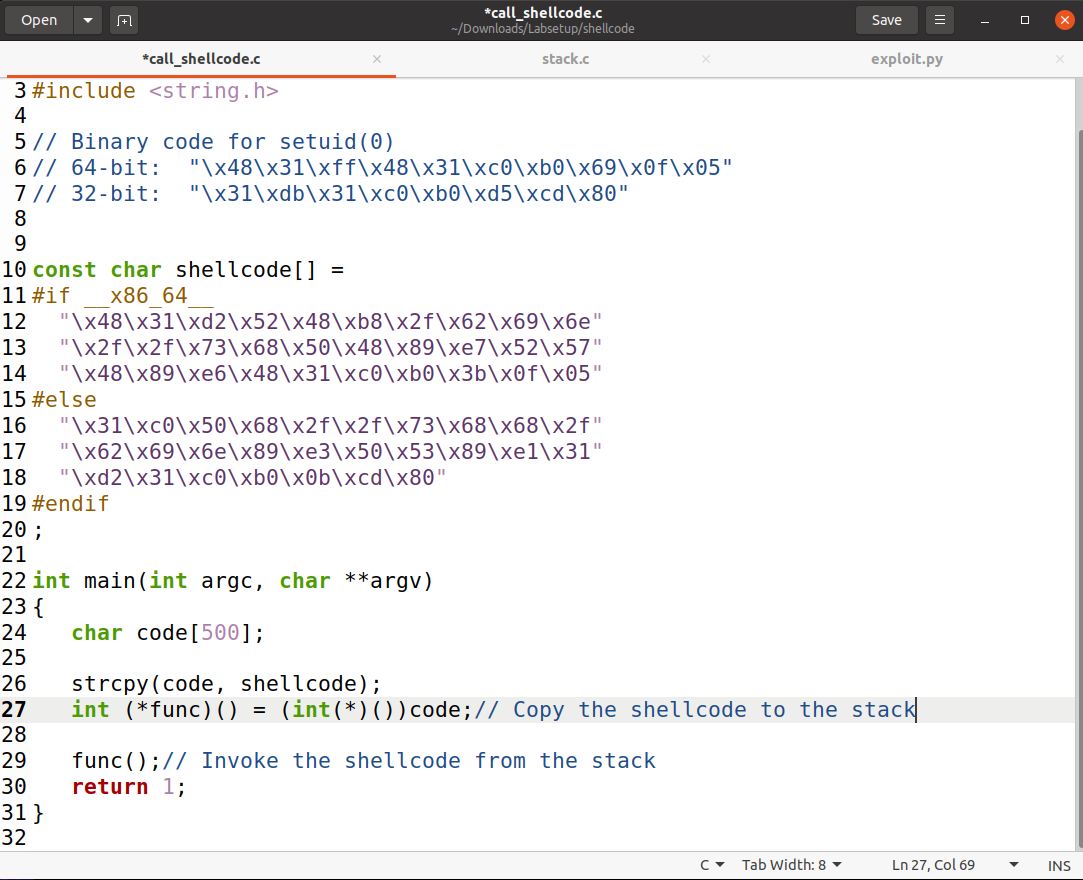
The system randomizes the start addresses of heaps and stacks. Guessing the address is one of the key steps in buffering an overflow attack. You can drop this feature with the following command:

sudo sysctl -w kernel.randomize\_va\_space

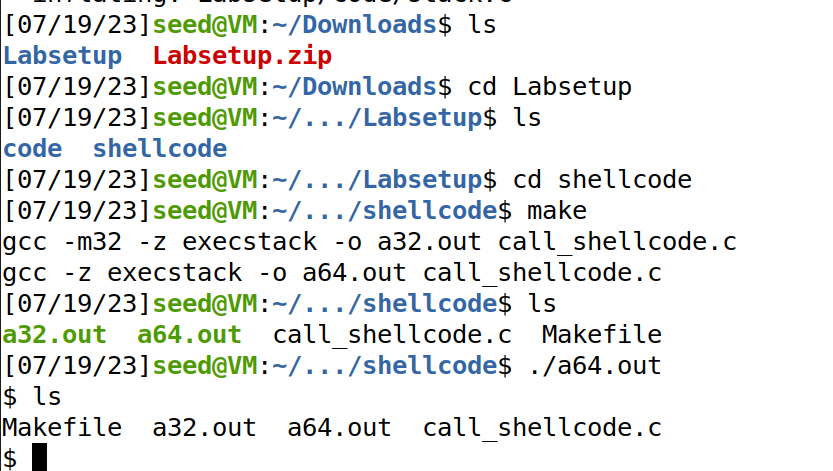
Because the /bin/sh symbolic link now points to /bin/dash, and in the first experiment, we learned that dash has a security policy for set-UID programs, relinquishing privileges. So we need to change the symlink to /bin/zsh and use the following command:

|  |  |
| --- | --- |
| 1 | sudo ln -sf /bin/zsh /bin/sh |

Task: Invoking the Shellcode source code is as follows:

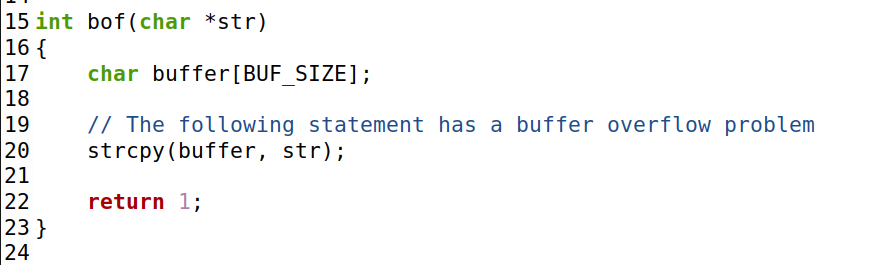
****

Because a Makefile file is already provided, it can be implemented very easily

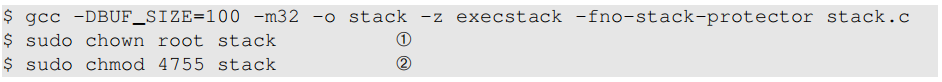
****

**Task 2: Understanding the Vulnerable Program**

From the source code audit, we know that the vulnerability lies in the strcpy (buffer, str) function in the bof function.

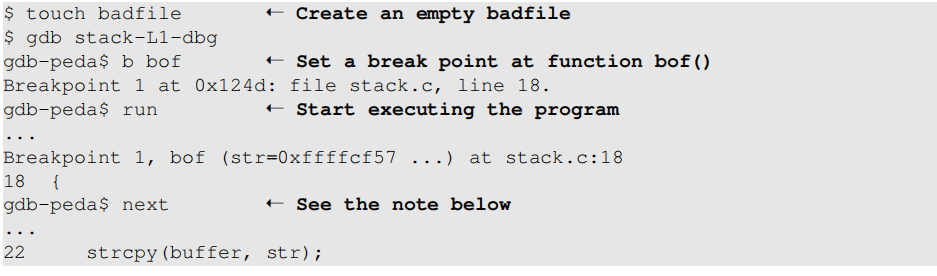


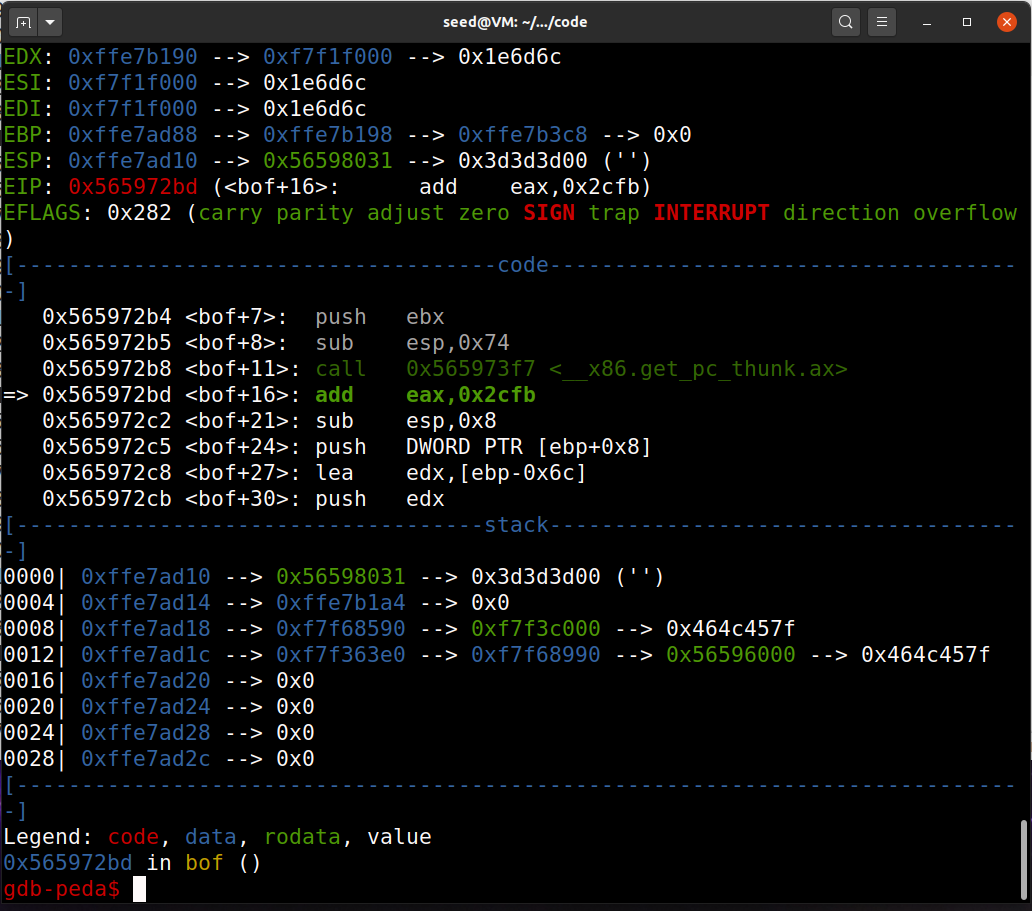
We need to fill the badfile file with a payload to cause a buffer overflow vulnerability, and the specific command line is given in the official experiment pdf file

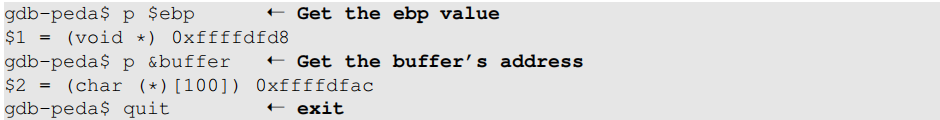


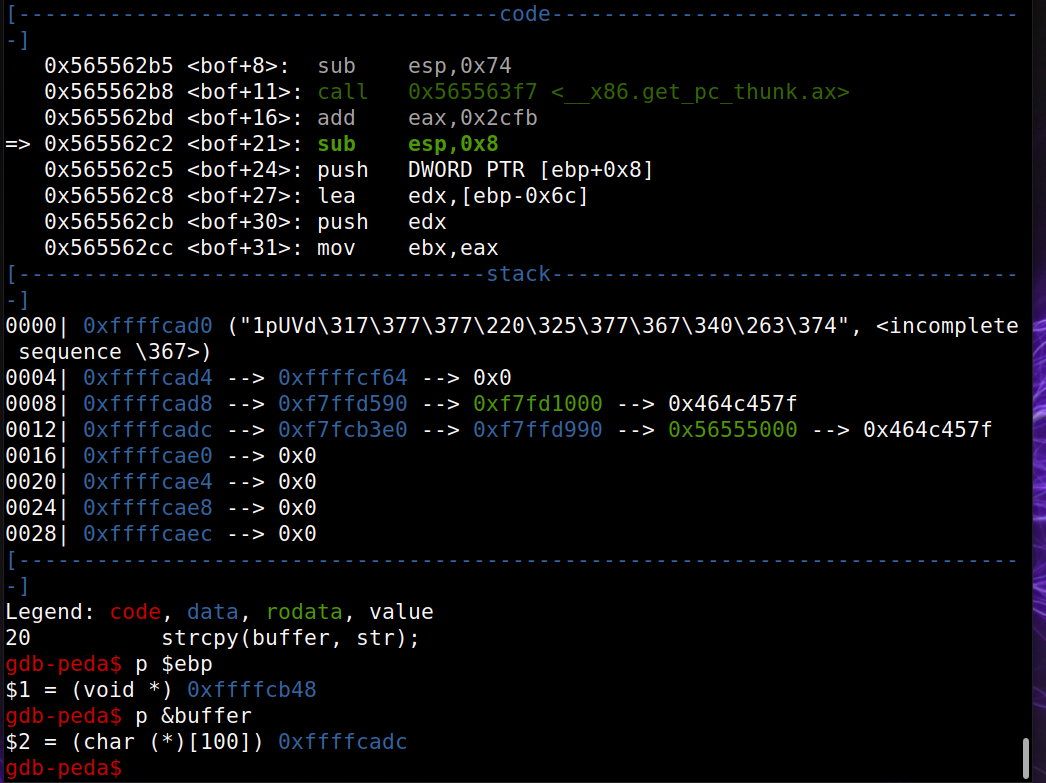
**Task 3: Launching Attack on 32-bit Program**

To launch a buffer overflow attack, perform the following steps:

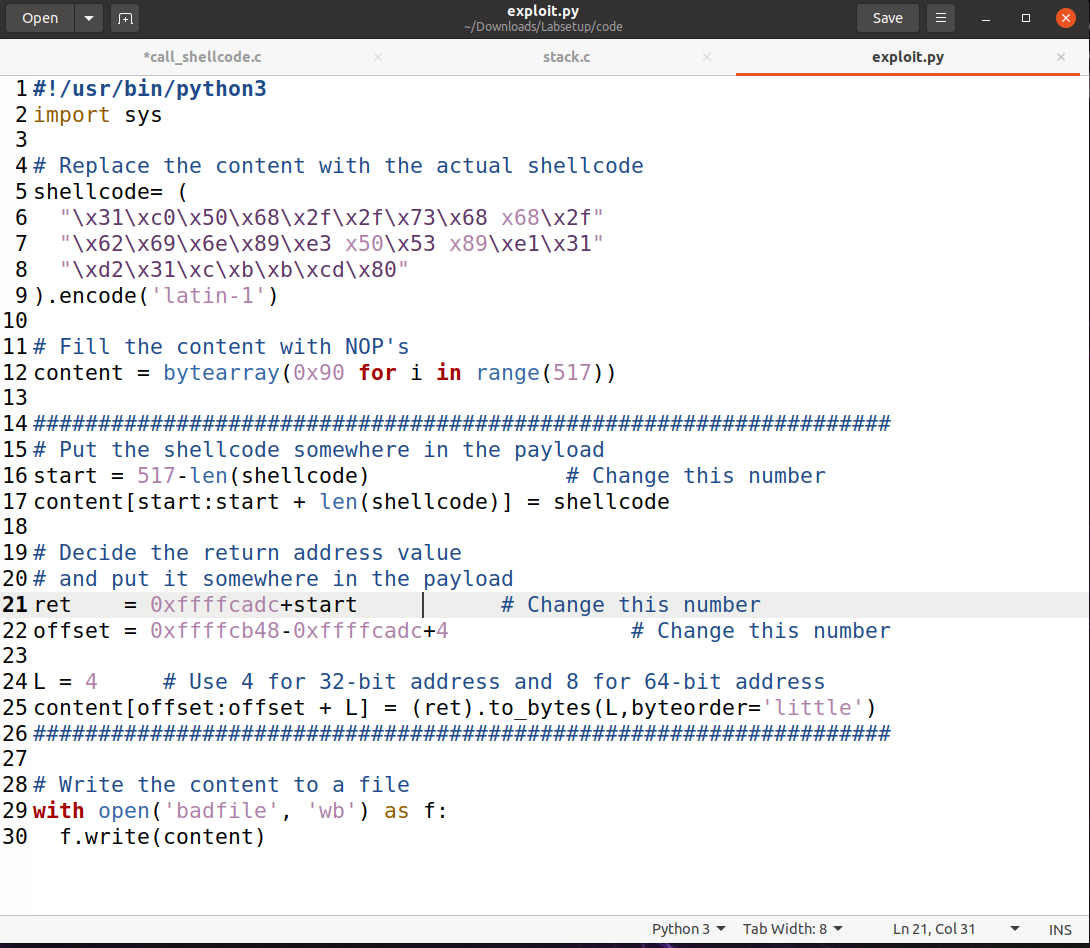


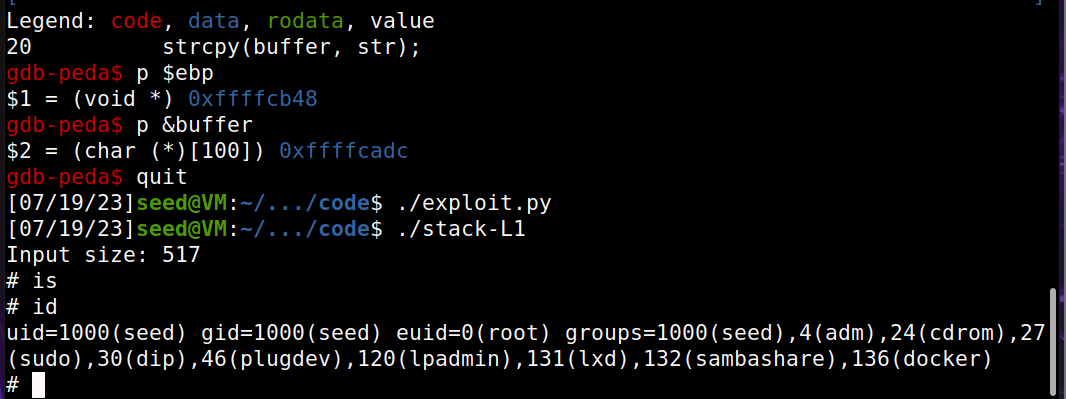






Modify the exploit.py file



Generate payloads and attack  


You can see that we successfully got the root access