lab1 Packet Sniffing and Spoofing Lab

57118138 李嘉怡

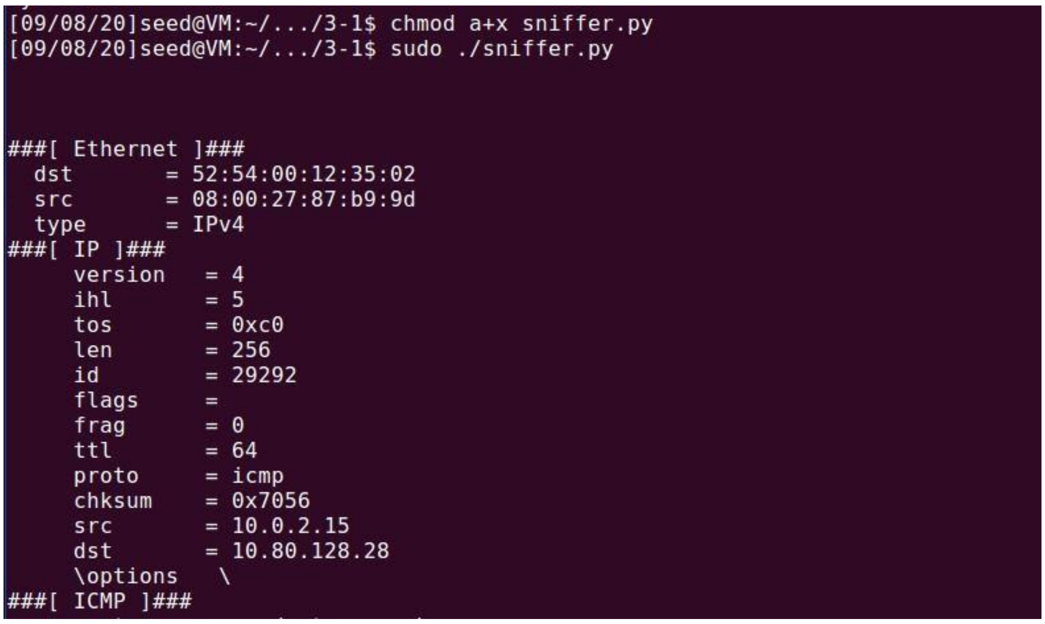
Task 1.1: Sniffing Packets

Task 1.1A

将手册中的代码复制到sniffer.py中，执行以下命令：

```bash
chmod a+x sniffer.py
sudo ./sniffer.py
```

输出的结果如下图所示（部分）：

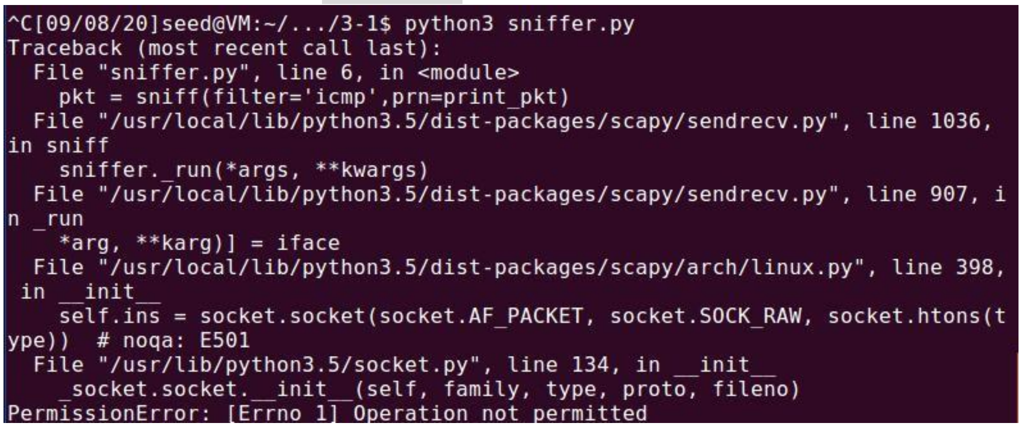


```
[09/08/20]seed@VM:~/.../3-1$ chmod a+x sniffer.py
[09/08/20]seed@VM:~/.../3-1$ sudo ./sniffer.py



###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:87:b9:9d
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0xc0
     len       = 256
     id        = 29292
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x7056
     src       = 10.0.2.15
     dst       = 10.80.128.28
     \options   \
###[ ICMP ]###
```

以普通用户权限执行sniffer.py时，报错：



```
^C[09/08/20]seed@VM:~/.../3-1$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 6, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 1036,
in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py", line 907, i
n _run
    *arg, **karg)] = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py", line 398,
 in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(t
ype))  # noqa: E501
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

Task1.1B

- 仅捕获ICMP报文：

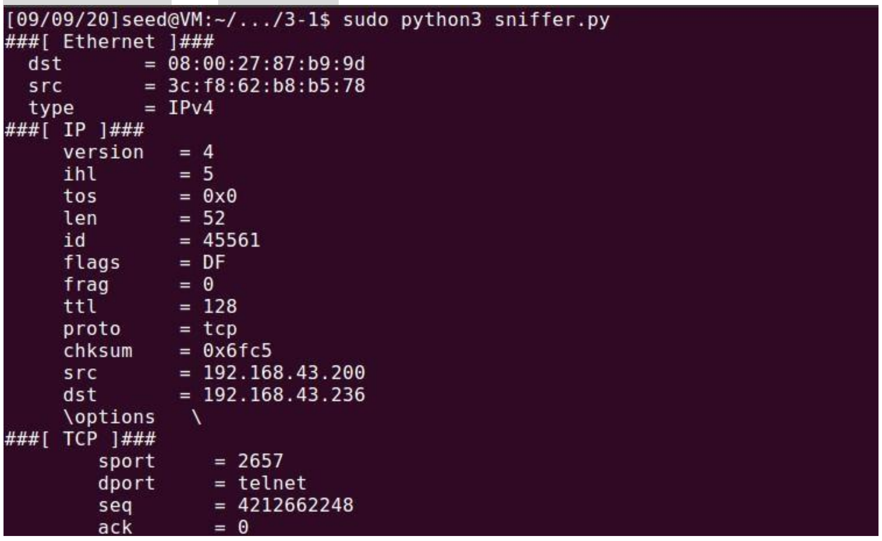
```
pkt=sniff(filter='icmp',prn=print_pkt)
```

filter与原代码一致，直接为 "icmp" 即可，输出也与上面一样。捕获从特定IP发出的，目的端口为23的TCP包 ，宿主机地址为： 192.168.43.200 ，虚拟机地址为：192.168.43.236 。

将程序sniffer.py 中的filter的代码改为：

```
src host 192.168.43.200 and tcp dst port23
```

在虚拟机中运行程序sniffer.py，然后在宿主机中运行telnet 的192.168.43.236 ， sniffer.py 输出结果图下图所示（部分）：



```
[09/09/20]seed@VM:~/.../3-1$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 08:00:27:87:b9:9d
  src       = 3c:f8:62:b8:b5:78
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 52
     id        = 45561
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x6fc5
     src       = 192.168.43.200
     dst       = 192.168.43.236
     \options   \
###[ TCP ]###
        sport     = 2657
        dport     = telnet
        seq       = 4212662248
        ack       = 0
```

- 捕获从特定子网中发起的报文： 将filter的代码改为：

```
src net 192.168.43.0/24 and dst net 192.168.43.0/24
```

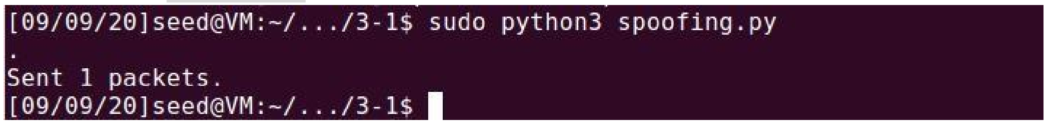
Task 1.2: Spoofing ICMP Packets

将手册中代码中的IP地址更改为自己的IP，如下所示：

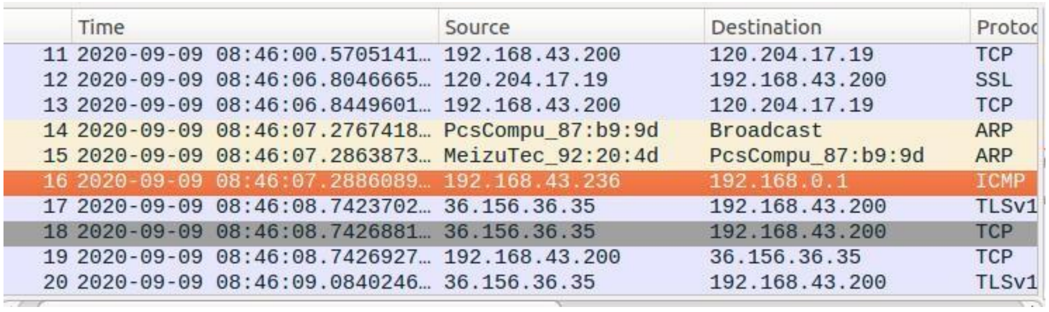
```
from scapy.all import *
a = IP()
a.src = '192.168.43.236'
a.dst = '192.168.0.1'
b = ICMP()
p = a/b
send(p)
```

其中， 192.168.43.236是虚拟机源地址， 192.168.0.1是目的地址。

运行程序spoofing.py，结果如下图所示：


```
[09/09/20]seed@VM:~/.../3-1$ sudo python3 spoofing.py
.
Sent 1 packets.
[09/09/20]seed@VM:~/.../3-1$ 
```

同时，wireshark抓到了来自192.168.43.236 发往192.168.0.1 的ICMP包，如下图所示：



	Time	Source	Destination	Protoc
11	2020-09-09 08:46:00.5705141…	192.168.43.200	120.204.17.19	TCP
12	2020-09-09 08:46:06.8046665…	120.204.17.19	192.168.43.200	SSL
13	2020-09-09 08:46:06.8449601…	192.168.43.200	120.204.17.19	TCP
14	2020-09-09 08:46:07.2767418…	PcsCompu_87:b9:9d	Broadcast	ARP
15	2020-09-09 08:46:07.2863873…	MeizuTec_92:20:4d	PcsCompu_87:b9:9d	ARP
16	2020-09-09 08:46:07.2886089…	192.168.43.236	192.168.0.1	ICMP
17	2020-09-09 08:46:08.7423702…	36.156.36.35	192.168.43.200	TLSv1
18	2020-09-09 08:46:08.7426881…	36.156.36.35	192.168.43.200	TCP
19	2020-09-09 08:46:08.7426927…	192.168.43.200	36.156.36.35	TCP
20	2020-09-09 08:46:09.0840246…	36.156.36.35	192.168.43.200	TLSv1

Task 1.3: Traceroute

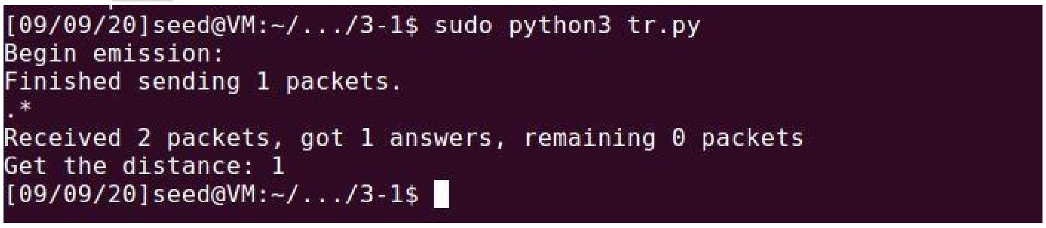
根据手册中的代码，稍加更改，保存为tr.py ，如下所示：

```
#!/usr/bin/python3

from scapy.all import *
import sys

a=IP()
a.dst = '192.168.43.200'         # 宿主机的IP
b = ICMP()
is_get_dis = 0
m_ttl = 1
i = 1
while is_get_dis == False:
    a.ttl = m_ttl
    ans, un_ans = sr(a/b)
    if ans.res[0][1].type ==0:
        is_get_dis = True
    else:
        i += 1
        m_ttl += 1
print('Get the distance:',i)
```

运行tr.py ，结果显示到宿主机IP的跳数为1，如下图所示：


```
[09/09/20]seed@VM:~/.../3-1$ sudo python3 tr.py
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
Get the distance: 1
[09/09/20]seed@VM:~/.../3-1$ 
```

Task 1.4: Sniffing and-then Spoofing

编写sniff-spoof.py，将ICMP报文的源地址和宿地址互换，然后发送。如下所示：

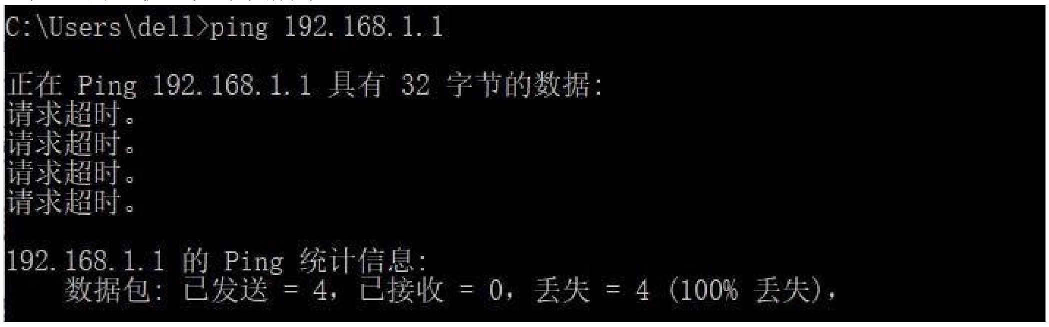
```python
#!/usr/bin/python3

from scapy.all import *

def send_back_pkt(pkt):
    head = IP()
    head.src  = pkt[IP].dst
    head.dst  = pkt[IP].src
    icmp = ICMP()
    icmp.type = 'echo-reply'
    icmp.code = 0
    icmp.id = pkt[ICMP].id
    icmp.seq =pkt[ICMP].seq
    new_pkt = head/icmp
    send(new_pkt)

pkt = sniff(filter='icmp[icmptype] ==icmp-echo',prn=send_back_pkt)
```

首先在宿主机上直接运行ping 192.168.1.1 ，此时显示请求超时，因为并没有这个IP 的主机，如下图所示：



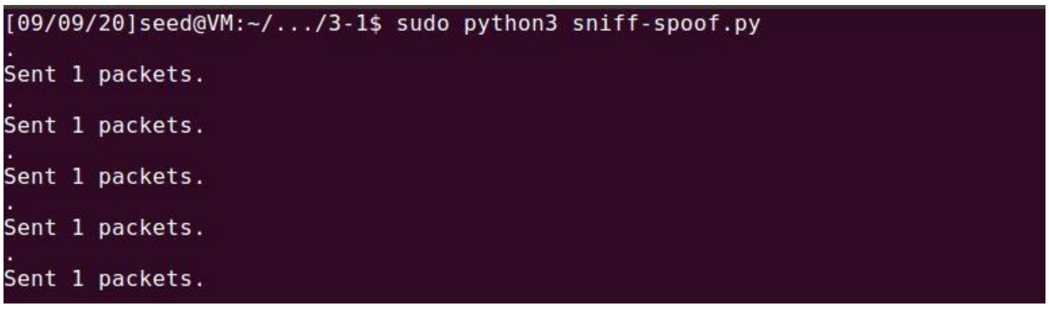
```
C:\Users\dell>ping 192.168.1.1

正在 Ping 192.168.1.1 具有 32 字节的数据:
请求超时。
请求超时。
请求超时。
请求超时。

192.168.1.1 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 0，丢失 = 4 (100% 丢失)，
```

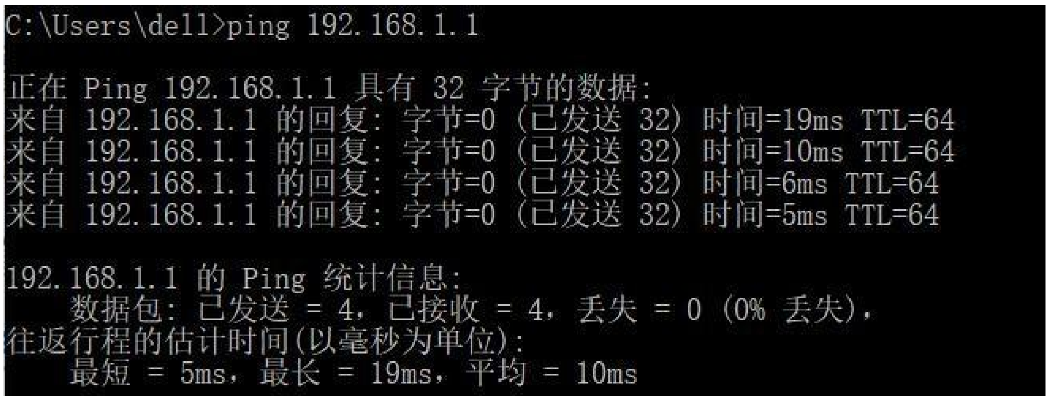
然后在虚拟机上运行程序sniff-spoof.py,再次在宿主机上运行ping 192.168.1.1，这样不管ping的对端IP是否存活，都可以收到回复。

虚拟机上sniff-spoof.py的输出结果如下图所示：



```
[09/09/20]seed@VM:~/.../3-1$ sudo python3 sniff-spoof.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

宿主机上的输出结果如下图所示：



```
C:\Users\dell>ping 192.168.1.1

正在 Ping 192.168.1.1 具有 32 字节的数据:
来自 192.168.1.1 的回复: 字节=0 (已发送 32) 时间=19ms TTL=64
来自 192.168.1.1 的回复: 字节=0 (已发送 32) 时间=10ms TTL=64
来自 192.168.1.1 的回复: 字节=0 (已发送 32) 时间=6ms TTL=64
来自 192.168.1.1 的回复: 字节=0 (已发送 32) 时间=5ms TTL=64

192.168.1.1 的 Ping 统计信息:
    数据包: 已发送 = 4，已接收 = 4，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 5ms，最长 = 19ms，平均 = 10ms
```

这说明成功对IP 192.168.1.1进行了伪造。