Packet Sniffing and Spoofing Lab

姓名：宋雨帆 学号：57118225

Task 1.1

Task1.1A

（1）非 root 权限下运行

```
root@VM:/volumes# su seed
seed@VM:/volumes$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 4, in <module>
    pkt=sniff(iface='br-439134a5b817',filter='icmp',prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, i
n sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 906, in
 _run
    sniff_sockets[L2socket(type=ETH_P_ALL, iface=iface,
  File "/usr/local/lib/python3.8/dist-packages/scapy/arch/linux.py", line 398,
in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(ty
pe))  # noqa: E501
```

（2）root 权限下运行

```
root@VM:/volumes# chmod a+x sniffer.py
root@VM:/volumes# python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:ba:92:93:e2
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 4434
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x1540
     src       = 10.9.0.5
     dst       = 10.9.0.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x599e
        id        = 0x10
        seq       = 0x1
###[ Raw ]###
           load      = '\xfd5\xe3`\x00\x00\x00\x00\xf8\x
e6\x06\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x
17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234
567'
```

使用根权限运行该程序，证明 root 用户确实可以捕获数据包。再次运行程序，但不使用根权限，程序就会报错，证明 seed 用户不能捕获数据包。

Task1.1B

（1）只查看 ICMP 报文

和 1.1A 的程序一样，filter 等于'icmp'，因此上述结果就是只获取 icmp 报文的结果。

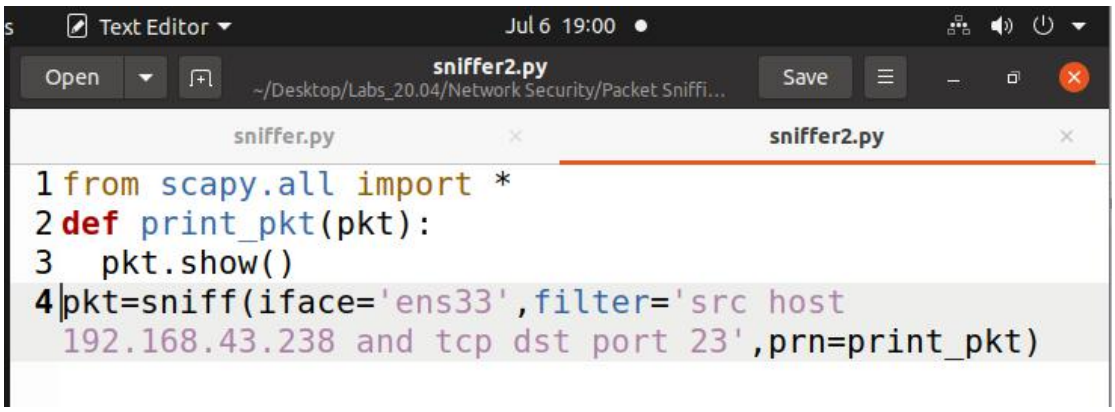
（2）查看特定 ip 发出，由 23 端口接受的报文

首先查看自己电脑的 ip 地址（192.168.43.238）：

再查看虚拟机 ip 地址（192.168.43.106），并记下端口名(ens33)：

修改 sniffer 程序:

程序代码如下：



程序运行结果：

```
###[ Ethernet ]###
  dst       = 00:0c:29:15:d6:e2
  src       = b8:9a:2a:74:8f:d8
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 38791
     flags     = DF
     frag      = 0
     ttl       = 128
     proto     = tcp
     chksum    = 0x8a9f
     src       = 192.168.43.238
     dst       = 192.168.43.106
     \options   \
###[ TCP ]###
        sport     = 58565
        dport     = telnet
        seq       = 1904674020
        ack       = 3160548148
        dataofs   = 5
        reserved  = 0
        flags     = A
        window    = 2052
        chksum    = 0x9448
        urgptr    = 0
        options   = []
###[ Padding ]###
           load      = '\x00\x00\x00\x00\x00\x00'
```

(3) 查看某个子网的报文

为了产生子网的报文，我们可以使用 Task1.2 中的伪装技术，在用户容器中产生源地址在 128.230.0.0/16 网络中的报文，并发送给攻击者，攻击者运行程序，从而模拟嗅探过程。代码如下（其中用 src net 来表示来源的网络，其值为 128.230，表示子网掩码是 255.255.0.0）：

```
>>> from scapy.all import *
>>> a=IP()
>>> a.src='128.230.2.1'
>>> a.dst='10.9.0.1'
>>> b=ICMP()
>>> p=a/b
>>> send(p)
.
Sent 1 packets.
>>> 
```

攻击者嗅探程序的运行结果：

```
root@VM:/volumes# python3 sniffer3.py
###[ Ethernet ]###
  dst       = 02:42:d8:e2:7e:b0
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xedef
     src       = 128.230.2.1
     dst       = 10.9.0.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xf7ff
        id        = 0x0
        seq       = 0x0
```

可以看到，成功收到了来自 128.230.2.1 的报文。

Task1.2

创建 IP 对象，将其源地址设为要伪造的 ip 地址，目标地址设为要发送的主机 ip。这里将伪造 ip 设为 1.1.2.3，目标 ip 地址为虚拟机地址。将其负载到 icmp 中对象中并发送。
程序如下：

```python
from scapy.all import *
a=IP()
a.src='1.1.2.3'
a.dst='192.168.43.106'
b=ICMP()
p=a/b
send(p)
```

Wireshark 检测 192.168.43.106 端口，发现了来自 1.1.2.3 的报文，说明伪造成功。

Task1.3

编写程序如下，不断增加 ttl 值，构造报文并发送：

```python
from scapy.all import *
ttl=1
for i in range(0,100):
  a=IP()
  a.dst='202.108.22.5'
  a.ttl=ttl
  b=ICMP()
  p=a/b
  send(p)
  ttl=ttl+1
```

百度 ip：

55	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response f…
56	2021-07-06 18:3…	192.168.43.1	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
57	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response f…
58	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response f…
59	2021-07-06 18:3…	10.136.121.138	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
60	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response f…
61	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response f…
62	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response f…
63	2021-07-06 18:3…	183.207.220.129	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
64	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response f…
65	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response f…
66	2021-07-06 18:3…	221.183.95.34	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
67	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response f…
68	2021-07-06 18:3…	219.158.44.29	192.168.43.106	ICMP	112	Time-to-live exceeded (Time to live exceeded in transit)
69	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response …
70	2021-07-06 18:3…	219.158.99.149	192.168.43.106	ICMP	112	Time-to-live exceeded (Time to live exceeded in transit)
71	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response …
72	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=12 (no response …
73	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=13 (no response …
74	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=14 (no response …
75	2021-07-06 18:3…	125.33.185.210	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
76	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=15 (no response …
77	2021-07-06 18:3…	220.206.193.82	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
78	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=16 (no response …
79	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=17 (no response …
80	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=18 (no response …
81	2021-07-06 18:3…	192.168.43.106	202.108.22.5	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=19 (reply in 83)
82	2021-07-06 18:3…	10.166.0.44	192.168.43.106	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)

可以看到在 ttl 为 19 的时候收到了 reply 报文，说明路径上的跳数大约为 19，其中有 9 个路由器发送了 icmp 错误报文。

Task1.4

该任务的目的是：对于正常用户发送的 ping，无论 ip 地址是否存在都返回一个回应的报文。实现方法：对于 IP 段，交换源地址和宿地址作为回应报文的 IP；对于 ICMP 类型，设为 reply 类型（0）；对于其它部分，复制用户报文的内容放到相应位置。

代码如下：

```
from scapy.all import *
def work_pkt(pkt):
    if pkt[ICMP].type !=8:
        return
    p1=IP()
    p1.src=pkt[IP].dst
    p1.dst=pkt[IP].src
    p1.ihl=pkt[IP].ihl
    p2=ICMP()
    p2.type=0;
    p2.id=pkt[ICMP].id
    p2.seq=pkt[ICMP].seq
    p3=pkt[Raw].load
    c=p1/p2/p3
    send(c)

pkt=sniff(iface='br-439134a5b817',filter='icmp',prn=work_pkt)
```

（1）尝试 ping 1.2.3.4（网络中不存在的 ip）：
攻击者未开启程序：

```
[07/05/21]seed@VM:~/Desktop$ dockps
ece141701323  host-10.9.0.5
05c976873a34  seed-attacker
[07/05/21]seed@VM:~/Desktop$ docksh ec
root@ece141701323:/# ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
```

攻击者开启程序：

```
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=114 ttl=64 time=65.4 ms
64 bytes from 1.2.3.4: icmp_seq=115 ttl=64 time=20.5 ms
64 bytes from 1.2.3.4: icmp_seq=116 ttl=64 time=23.4 ms
64 bytes from 1.2.3.4: icmp_seq=117 ttl=64 time=21.4 ms
64 bytes from 1.2.3.4: icmp_seq=118 ttl=64 time=17.8 ms
64 bytes from 1.2.3.4: icmp_seq=119 ttl=64 time=17.7 ms
64 bytes from 1.2.3.4: icmp_seq=120 ttl=64 time=17.9 ms
64 bytes from 1.2.3.4: icmp_seq=121 ttl=64 time=31.7 ms
```

开启伪造程序后，用户成功收到了相应报文，说明伪造成功！

（2）尝试 ping （局域网中不存在的 ip）

攻击者未开启：

```
root@ece141701323:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
From 10.9.0.5 icmp_seq=4 Destination Host Unreachable
From 10.9.0.5 icmp_seq=5 Destination Host Unreachable
```

开启程序：

```
root@ece141701323:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.5 icmp_seq=1 Destination Host Unreachable
From 10.9.0.5 icmp_seq=2 Destination Host Unreachable
From 10.9.0.5 icmp_seq=3 Destination Host Unreachable
From 10.9.0.5 icmp_seq=4 Destination Host Unreachable
```

造成这种现象的原因是局域网中没有配置该地址的路由。

（3）尝试 ping 8.8.8.8（网络中存在的 ip）

攻击者未开启程序：

```
root@ece141701323:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=49 time=87.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=49 time=144 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=49 time=90.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=49 time=89.1 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=49 time=87.1 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=49 time=83.7 ms
```

开启程序：

```
root@ece141701323:/# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=47.7 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=49 time=84.5 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=14.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=49 time=82.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=19.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=49 time=88.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=17.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=49 time=120 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=17.0 ms
```

出现 DUP 的原因是因为真实的 8.8.8.8 主机会发送 icmp 回应报文，而攻击者也会发送一份，这就导致用户会重复受到相同报文，从而提示 DUP 信息。