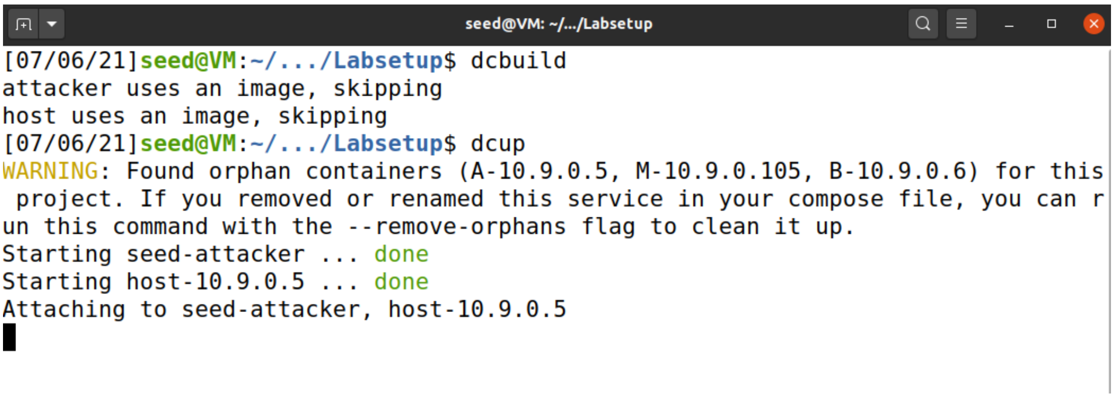
Lab1-report

57118239 张鹏

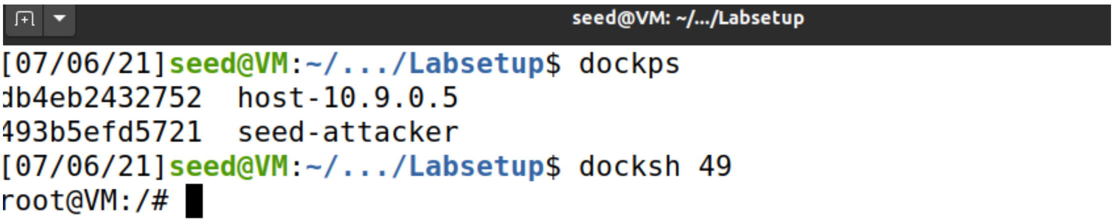
task1：

task1.1A：

本次实验通过 python 中的 scapy 模块来实现报文的监听和伪造。
首先通过 docker 登录 attacker 容器

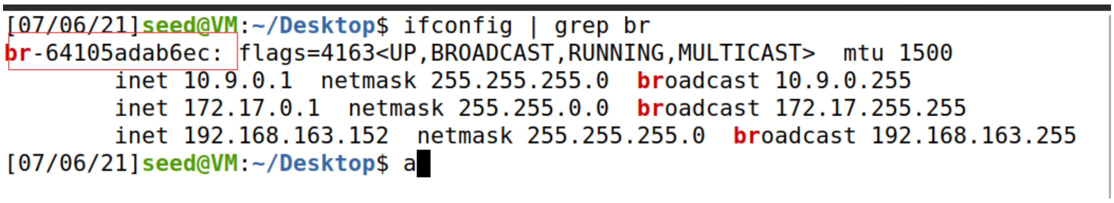


```
[07/06/21]seed@VM:~/.../Labsetup$ dcbuild
attacker uses an image, skipping
host uses an image, skipping
[07/06/21]seed@VM:~/.../Labsetup$ dcup
WARNING: Found orphan containers (A-10.9.0.5, M-10.9.0.105, B-10.9.0.6) for this
 project. If you removed or renamed this service in your compose file, you can r
un this command with the --remove-orphans flag to clean it up.
Starting seed-attacker ... done
Starting host-10.9.0.5 ... done
Attaching to seed-attacker, host-10.9.0.5
```



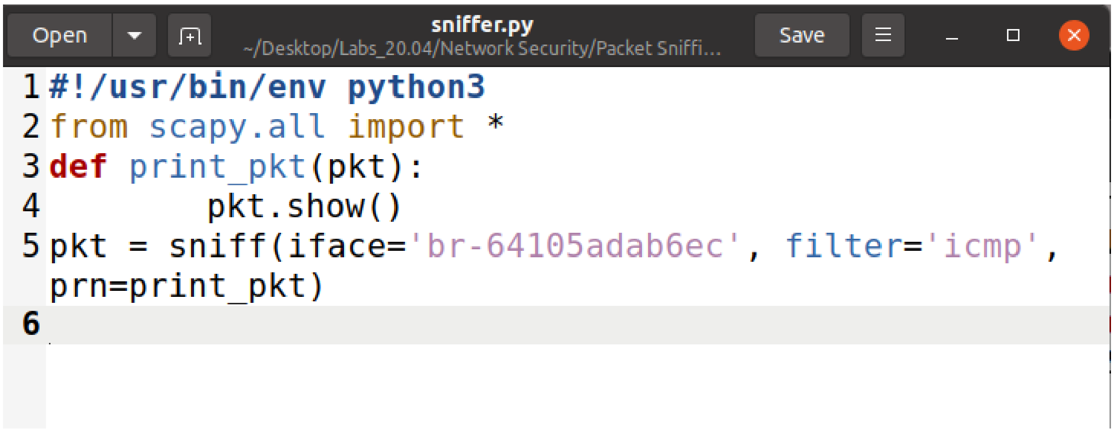
```
[07/06/21]seed@VM:~/.../Labsetup$ dockps
db4eb2432752  host-10.9.0.5
493b5efd5721  seed-attacker
[07/06/21]seed@VM:~/.../Labsetup$ docksh 49
root@VM:/# 
```

通过 ifconfig 确定需要监听的接口



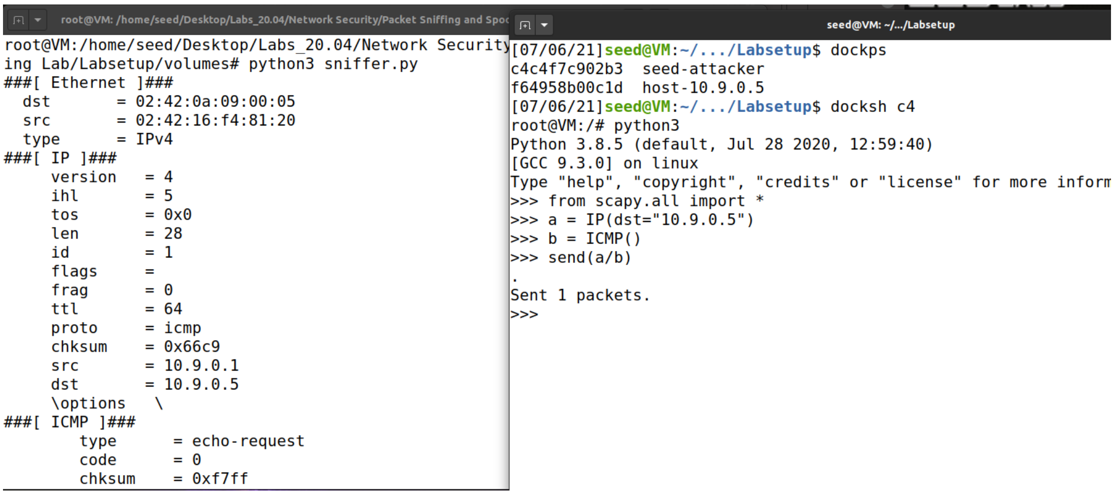
```
[07/06/21]seed@VM:~/Desktop$ ifconfig | grep br
br-64105adab6ec: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        inet 192.168.163.152  netmask 255.255.255.0  broadcast 192.168.163.255
[07/06/21]seed@VM:~/Desktop$ a
```

通过 sniff 来监听数据包，关键字参数 prn 的值是监听到符合 filter 规则的报文后所调用的函数，以下实验中定义了 print_pkt 函数用于显示报文的内容



```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
pkt = sniff(iface='br-64105adab6ec', filter='icmp',
prn=print_pkt)

```

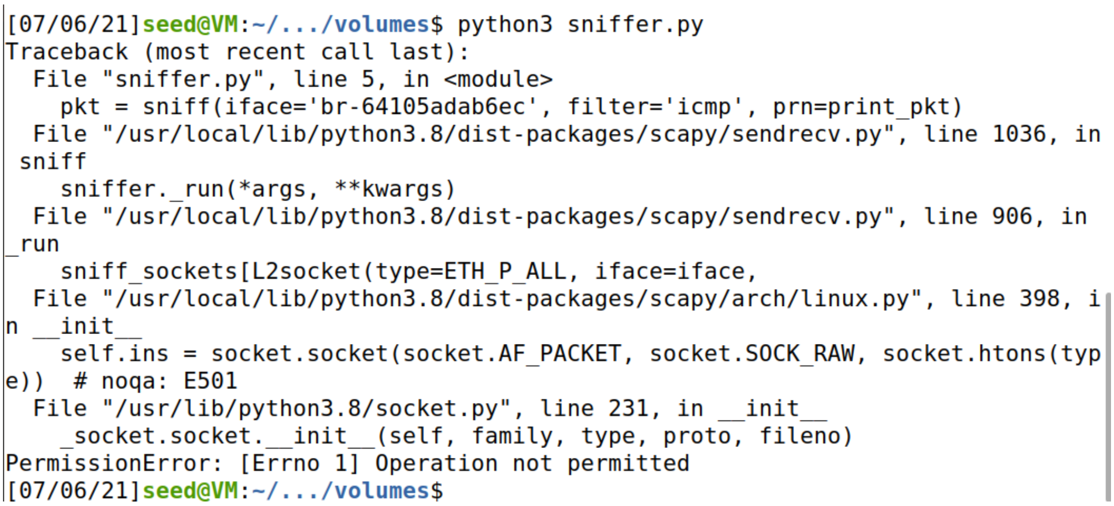
构造一个 ICMP 报文并发送，可以观察到被监听到了


root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoo
seed@VM: ~/.../Labsetup

```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security
ing Lab/Labsetup/volumes# python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:16:f4:81:20
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x66c9
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xf7ff
```

```
[07/06/21]seed@VM:~/.../Labsetup$ dockps
c4c4f7c902b3  seed-attacker
f64958b00c1d  host-10.9.0.5
[07/06/21]seed@VM:~/.../Labsetup$ docksh c4
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more inform
>>> from scapy.all import *
>>> a = IP(dst="10.9.0.5")
>>> b = ICMP()
>>> send(a/b)
.
Sent 1 packets.
>>>
```

当没有 root 权限要运行此程序的时候，可以发现程序会报错，因为其权限不够。

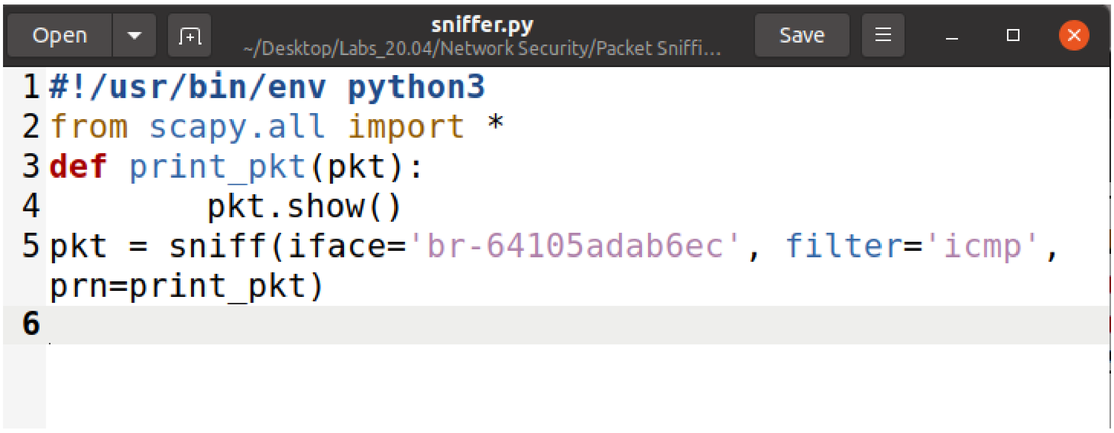


```
[07/06/21]seed@VM:~/.../volumes$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 5, in <module>
    pkt = sniff(iface='br-64105adab6ec', filter='icmp', prn=print_pkt)
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 1036, in
 sniff
    sniffer._run(*args, **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, i
n __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(typ
e))  # noqa: E501
  File "/usr/lib/python3.8/socket.py", line 231, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
[07/06/21]seed@VM:~/.../volumes$
```

task1.1B：

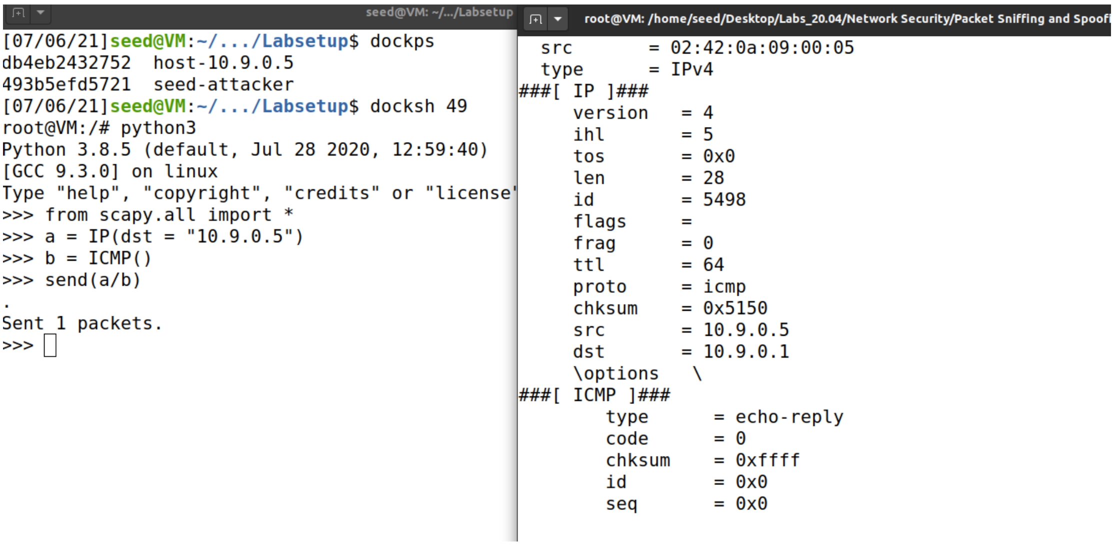
此实验的要求是通过更改 filter 规则，来实现对不同报文的监听。

下图为第一个规则，只监听 ICMP 报文。



```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
pkt = sniff(iface='br-64105adab6ec', filter='icmp', prn=print_pkt)
```

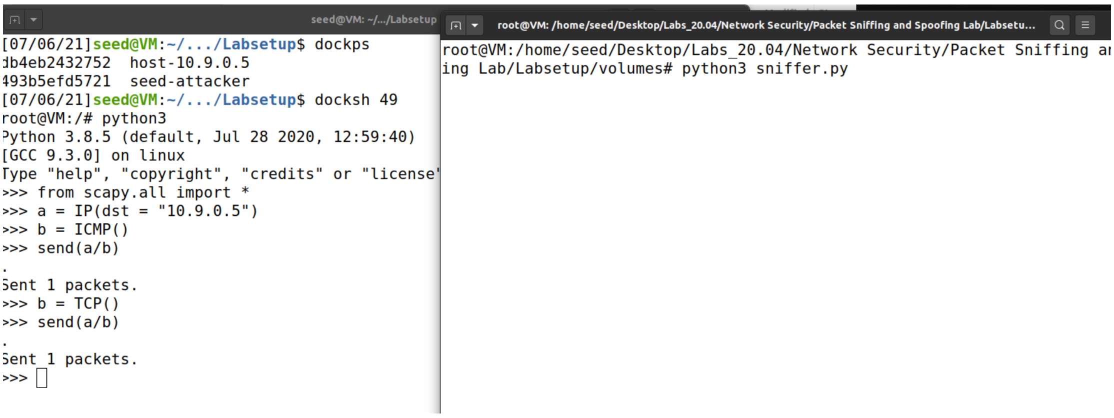
当发送 ICMP 报文时，可以发现监听到了。


seed@VM: ~/.../Labsetup
root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofi

```
[07/06/21]seed@VM:~/.../Labsetup$ dockps
db4eb2432752  host-10.9.0.5
493b5efd5721  seed-attacker
[07/06/21]seed@VM:~/.../Labsetup$ docksh 49
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license"
>>> from scapy.all import *
>>> a = IP(dst = "10.9.0.5")
>>> b = ICMP()
>>> send(a/b)
.
Sent 1 packets.
>>> 
```

```
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 5498
     flags     = 
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x5150
     src       = 10.9.0.5
     dst       = 10.9.0.1
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0xffff
        id        = 0x0
        seq       = 0x0
```

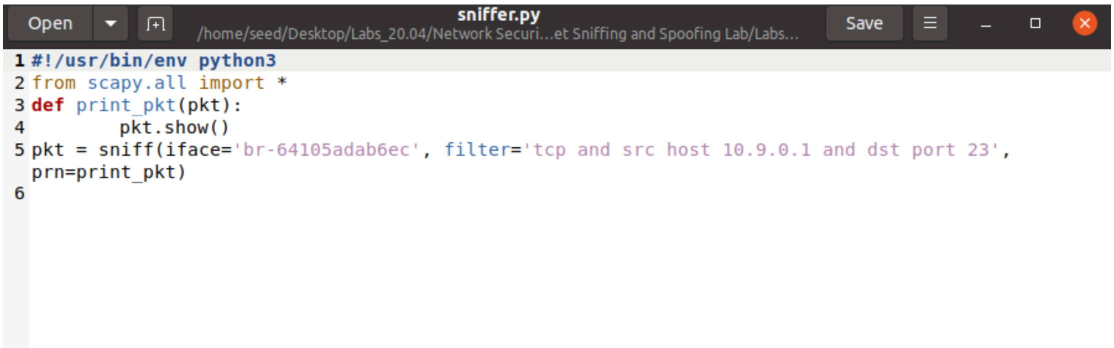
当发送的不是 ICMP 报文，可以发现监听程序并没有返回结果。


seed@VM: ~/.../Labsetup
root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab/Labsetu...

```
[07/06/21]seed@VM:~/.../Labsetup$ dockps
ib4eb2432752  host-10.9.0.5
493b5efd5721  seed-attacker
[07/06/21]seed@VM:~/.../Labsetup$ docksh 49
root@VM:/# python3
ython 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
ype "help", "copyright", "credits" or "license"
>>> from scapy.all import *
>>> a = IP(dst = "10.9.0.5")
>>> b = ICMP()
>>> send(a/b)
.
ent 1 packets.
>>> b = TCP()
>>> send(a/b)
.
ent 1 packets.
>>> 
```

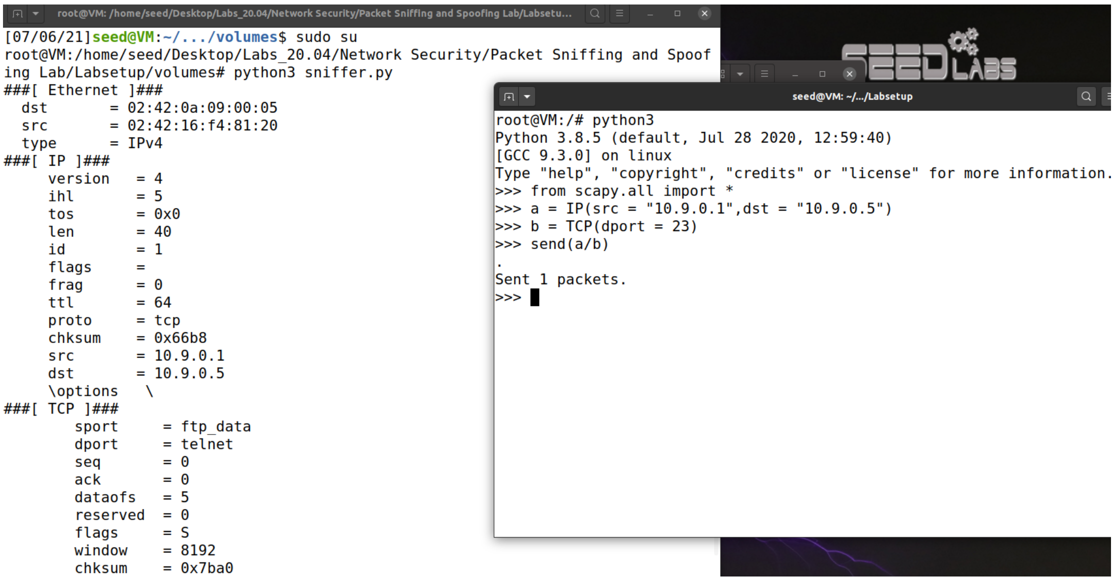
```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing ar
ing Lab/Labsetup/volumes# python3 sniffer.py
```

以下第二种是只监听 tcp 报文，且源地址是 10.9.0.1 和目的端口是 23。


Open
sniffer.py
/home/seed/Desktop/Labs_20.04/Network Securi...et Sniffing and Spoofing Lab/Labs...
Save

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
pkt = sniff(iface='br-64105adab6ec', filter='tcp and src host 10.9.0.1 and dst port 23',
prn=print_pkt)
```

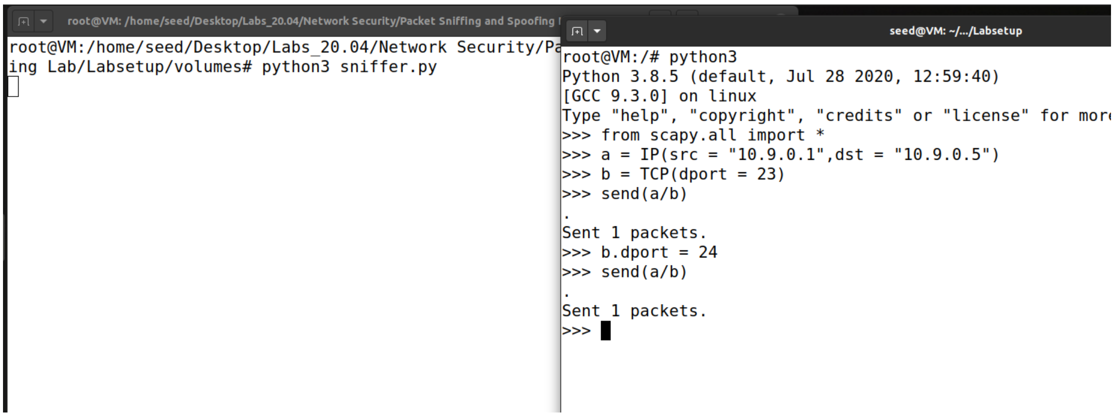
构造一个符合 filter 规则的报文，可以发现监听程序展示了其报文的详细信息。


root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab/Labsetu...
seed@VM: ~/.../Labsetup

```
[07/06/21]seed@VM:~/.../volumes$ sudo su
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoof
ing Lab/Labsetup/volumes# python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:16:f4:81:20
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x66b8
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ TCP ]###
        sport     = ftp_data
        dport     = telnet
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0x7ba0
```

```
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> a = IP(src = "10.9.0.1",dst = "10.9.0.5")
>>> b = TCP(dport = 23)
>>> send(a/b)
.
Sent 1 packets.
>>> 
```

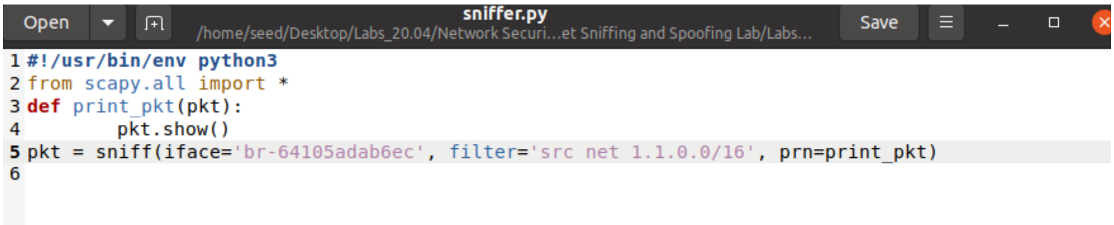
将刚刚构造的报文的目的端口号进行更改，再发送，可以发现监听程序没有输出。


root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing
seed@VM: ~/.../Labsetup

```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Pa
ing Lab/Labsetup/volumes# python3 sniffer.py
```

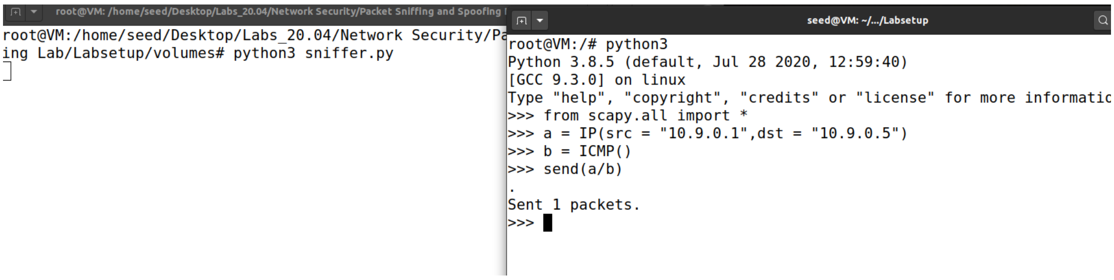
```
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more
>>> from scapy.all import *
>>> a = IP(src = "10.9.0.1",dst = "10.9.0.5")
>>> b = TCP(dport = 23)
>>> send(a/b)
.
Sent 1 packets.
>>> b.dport = 24
>>> send(a/b)
.
Sent 1 packets.
>>> 
```

以下第三种是监听源地址网段是 1.1.0.0/16 的报文。


Open
sniffer.py
/home/seed/Desktop/Labs_20.04/Network Securi...et Sniffing and Spoofing Lab/Labs...
Save

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
        pkt.show()
pkt = sniff(iface='br-64105adab6ec', filter='src net 1.1.0.0/16', prn=print_pkt)
```

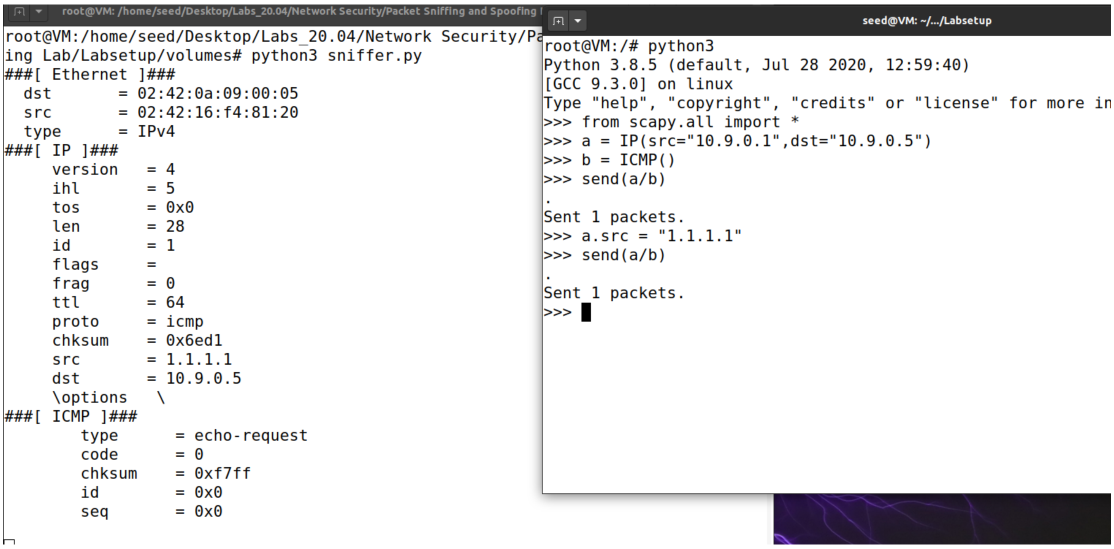
构造一个源地址不是此网段的报文可以发现监听程序没有输出。


root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing l
seed@VM: ~/.../Labsetup

```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Pa
ing Lab/Labsetup/volumes# python3 sniffer.py
```

```
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more informatic
>>> from scapy.all import *
>>> a = IP(src = "10.9.0.1",dst = "10.9.0.5")
>>> b = ICMP()
>>> send(a/b)
.
Sent 1 packets.
>>> 
```

构造一个源地址是此网段的报文可以发现监听程序展示了该报文的相应信息。


root@VM: /home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing l
seed@VM: ~/.../Labsetup

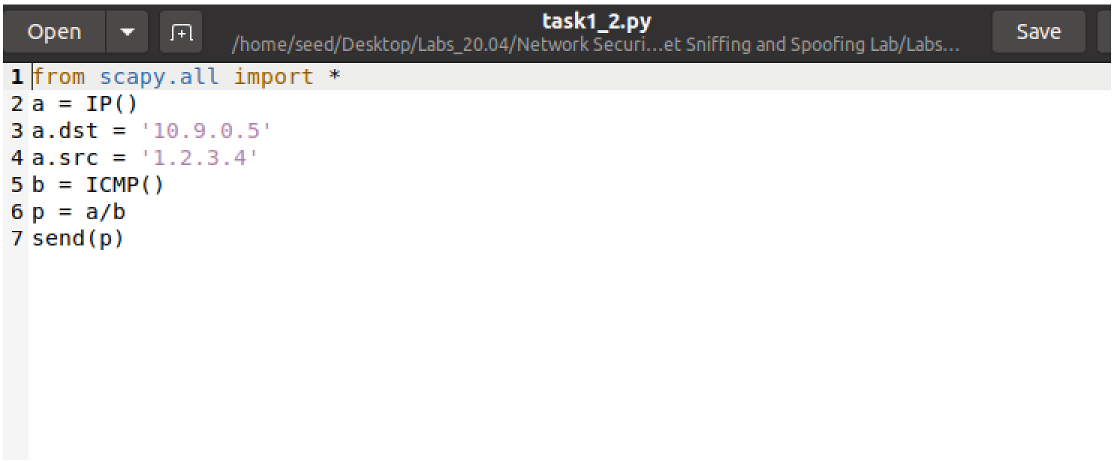
```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Pa
ing Lab/Labsetup/volumes# python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:16:f4:81:20
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     = 
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x6ed1
     src       = 1.1.1.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xf7ff
        id        = 0x0
        seq       = 0x0
```

```
root@VM:/# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more in
>>> from scapy.all import *
>>> a = IP(src="10.9.0.1",dst="10.9.0.5")
>>> b = ICMP()
>>> send(a/b)
.
Sent 1 packets.
>>> a.src = "1.1.1.1"
>>> send(a/b)
.
Sent 1 packets.
>>> 
```

task2:

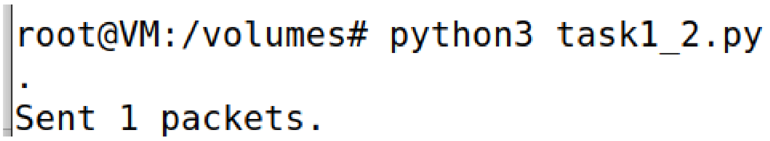
该实验的目的是根据其提供的代码进行一些必要的修改，可以构造任意源 IP 地址的 ICMP 报文。

下面的实验中将宿地址设置为 10.9.0.5， 源地址随意设置成了 1.2.3.4。


Open
task1_2.py
/home/seed/Desktop/Labs_20.04/Network Securi...et Sniffing and Spoofing Lab/Labs...
Save

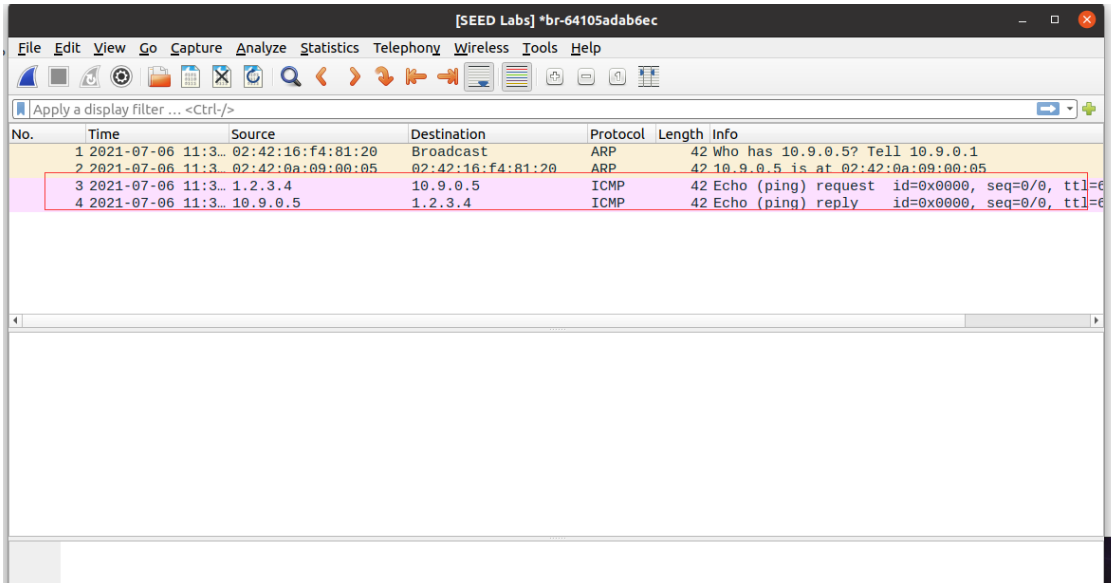
```python
from scapy.all import *
a = IP()
a.dst = '10.9.0.5'
a.src = '1.2.3.4'
b = ICMP()
p = a/b
send(p)
```

attacker 的 vm 中运行该段代码，则其伪造了一个源地址是 1.2.3.4，宿地址是 10.9.0.5 的 ICMP 报文。



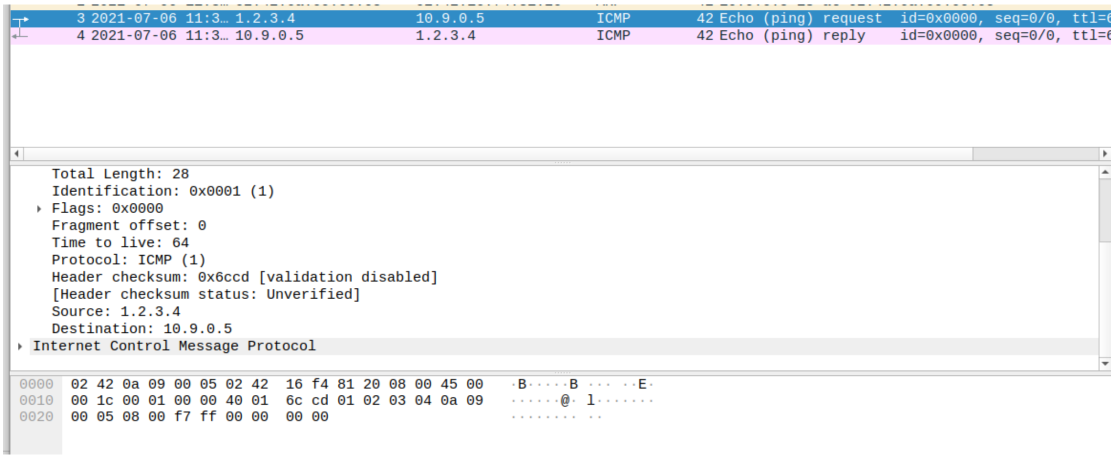
```
root@VM:/volumes# python3 task1_2.py
.
Sent 1 packets.
```

通过 wireshark 软件， 我们可以捕获到发送的 ICMP 报文及其内容。



[SEED Labs] *br-64105adab6ec
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.   Time               Source              Destination         Protocol Length Info
    1 2021-07-06 11:3…  02:42:16:f4:81:20   Broadcast           ARP        42 Who has 10.9.0.5? Tell 10.9.0.1
    2 2021-07-06 11:3…  02:42:0a:09:00:05   02:42:16:f4:81:20   ARP        42 10.9.0.5 is at 02:42:0a:09:00:05
    3 2021-07-06 11:3…  1.2.3.4             10.9.0.5            ICMP       42 Echo (ping) request  id=0x0000, seq=0/0, ttl=6
    4 2021-07-06 11:3…  10.9.0.5            1.2.3.4             ICMP       42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=6
```



```
    3 2021-07-06 11:3…  1.2.3.4             10.9.0.5            ICMP       42 Echo (ping) request  id=0x0000, seq=0/0, ttl=6
    4 2021-07-06 11:3…  10.9.0.5            1.2.3.4             ICMP       42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=6
```

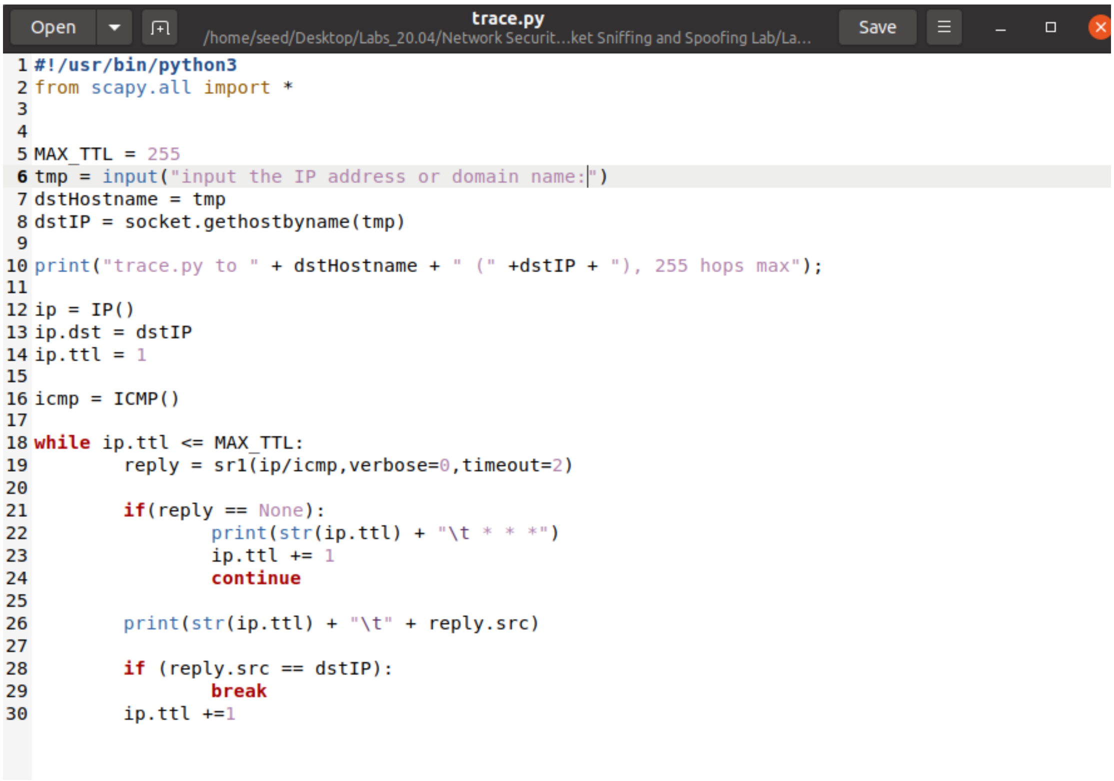
```
      Total Length: 28
      Identification: 0x0001 (1)
    ▸ Flags: 0x0000
      Fragment offset: 0
      Time to live: 64
      Protocol: ICMP (1)
      Header checksum: 0x6ccd [validation disabled]
      [Header checksum status: Unverified]
      Source: 1.2.3.4
      Destination: 10.9.0.5
  ▸ Internet Control Message Protocol
```

```
0000  02 42 0a 09 00 05 02 42  16 f4 81 20 08 00 45 00   ·B·····B ··· ··E·
0010  00 1c 00 01 00 00 40 01  6c cd 01 02 03 04 0a 09   ······@· l·······
0020  00 05 08 00 f7 ff 00 00  00 00                     ········ ··
```

task1.3：

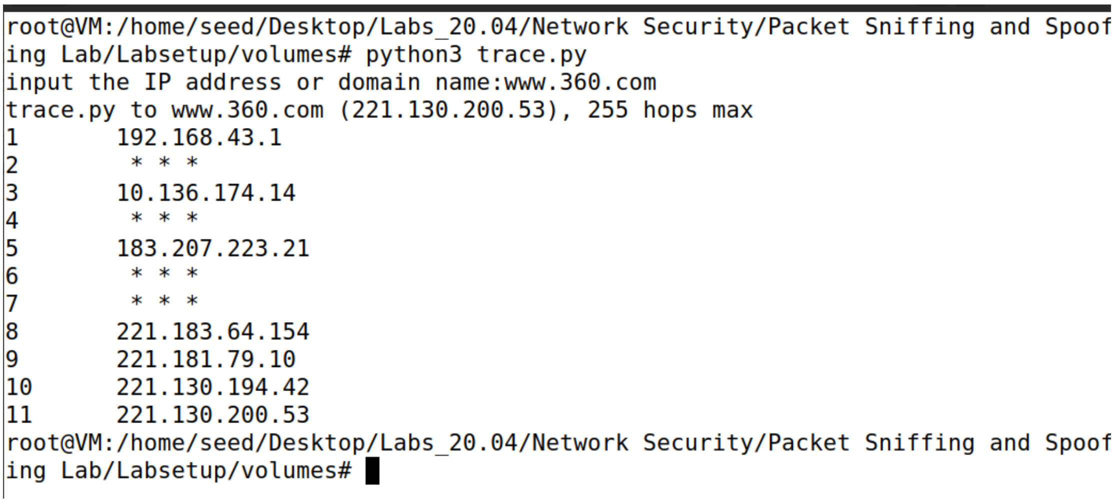
该实验的目的是让我们通过 scapy 实现 traceroute 的功能，原理是 ttl 在为 0 的时候会被路由器丢弃，同时返回一个 ICMP error，通过将 ttl 从 1 开始不断递增，可以得知从该地址到目标地址所跳转的地址。

代码的核心思想：运行程序后由用户主动输入需要 traceroute 的 IP 或者 domain name，一开始设置 ttl 为 1，构造一个 ICMP 报文，然后通过观察响应报文，如果是源地址是否为目标地址来判断是否到达，未到达则对 ttl 进行自增，进入下一次判断。


Open
trace.py
/home/seed/Desktop/Labs_20.04/Network Securit...ket Sniffing and Spoofing Lab/La...
Save

```python
#!/usr/bin/python3
from scapy.all import *


MAX_TTL = 255
tmp = input("input the IP address or domain name:")
dstHostname = tmp
dstIP = socket.gethostbyname(tmp)

print("trace.py to " + dstHostname + " (" +dstIP + "), 255 hops max");

ip = IP()
ip.dst = dstIP
ip.ttl = 1

icmp = ICMP()

while ip.ttl <= MAX_TTL:
        reply = sr1(ip/icmp,verbose=0,timeout=2)

        if(reply == None):
                print(str(ip.ttl) + "\t * * *")
                ip.ttl += 1
                continue

        print(str(ip.ttl) + "\t" + reply.src)

        if (reply.src == dstIP):
                break
        ip.ttl +=1
```

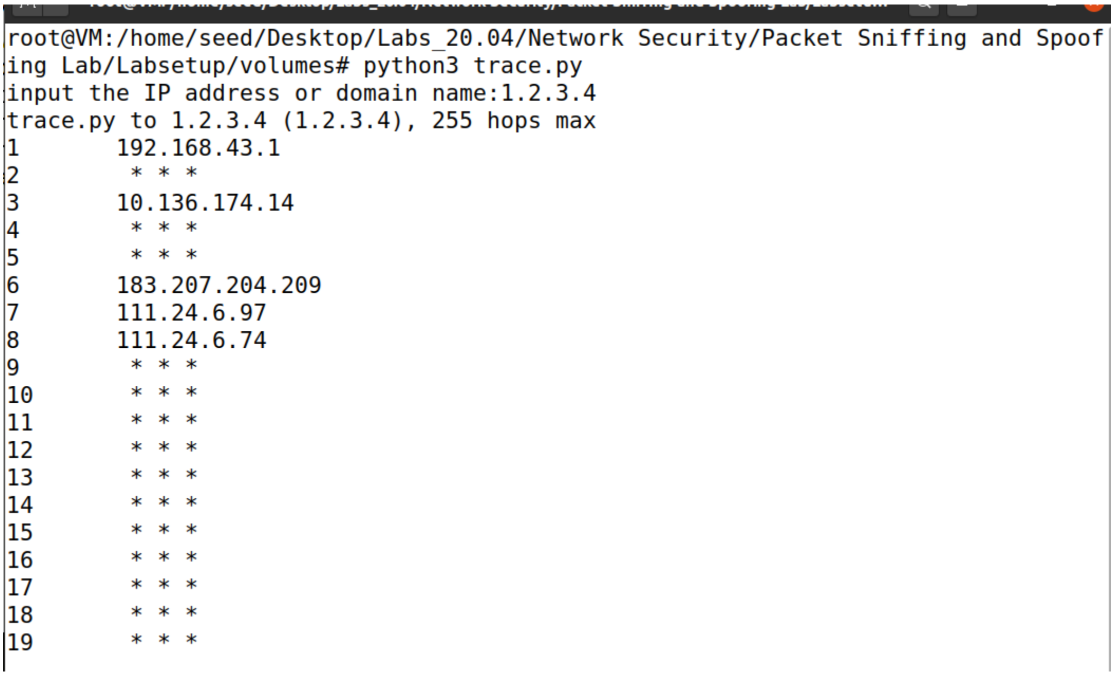
运行代码，对 www.360.com 进行 traceroute，结果如下



```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoof
ing Lab/Labsetup/volumes# python3 trace.py
input the IP address or domain name:www.360.com
trace.py to www.360.com (221.130.200.53), 255 hops max
1       192.168.43.1
2        * * *
3       10.136.174.14
4        * * *
5       183.207.223.21
6        * * *
7        * * *
8       221.183.64.154
9       221.181.79.10
10      221.130.194.42
11      221.130.200.53
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoof
ing Lab/Labsetup/volumes# █
```

可以得到结果，中间有几跳没有显示应该是对应主机的防火墙封掉了 ICMP 的信息。

对 1.2.3.4 进行 traceroute，结果如下：



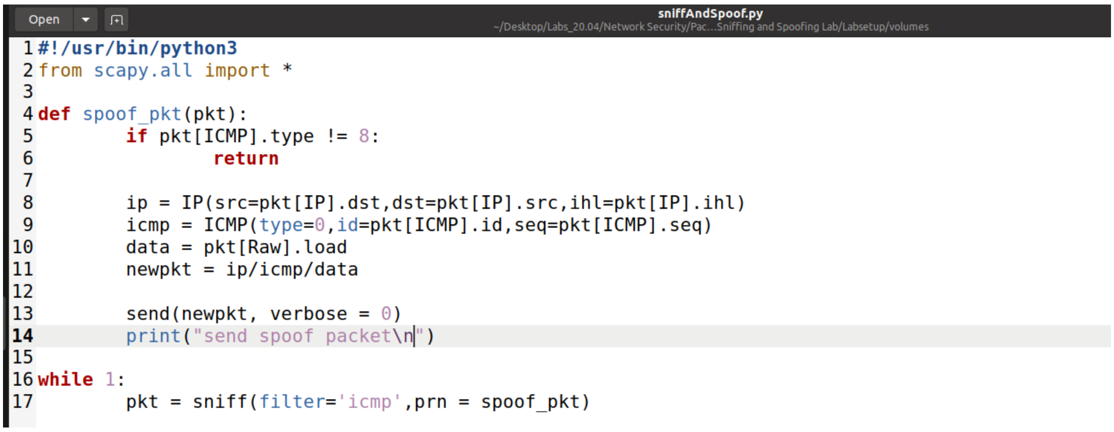
```
root@VM:/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoof
ing Lab/Labsetup/volumes# python3 trace.py
input the IP address or domain name:1.2.3.4
trace.py to 1.2.3.4 (1.2.3.4), 255 hops max
1        192.168.43.1
2         * * *
3        10.136.174.14
4         * * *
5         * * *
6        183.207.204.209
7        111.24.6.97
8        111.24.6.74
9         * * *
10        * * *
11        * * *
12        * * *
13        * * *
14        * * *
15        * * *
16        * * *
17        * * *
18        * * *
19        * * *
```

后续仍然没有结果，猜测可能该地址并不存在或者不可达。

task1.4：

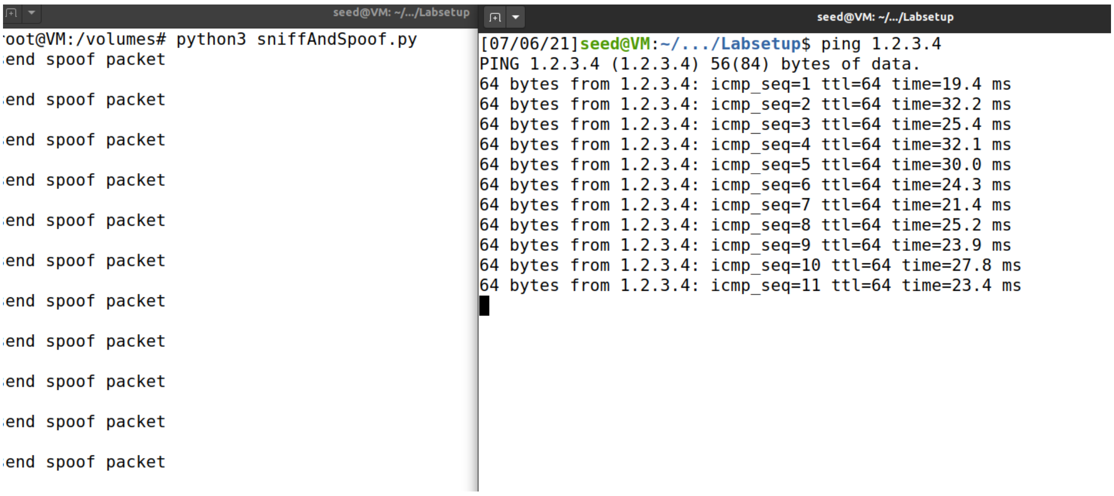
该实验的目的让我们通过监听 ICMP 请求报文，然后伪造 ICMP 的响应报文。

通过查阅 ICMP 报文的类型，可以知道实验中要求的 echo request 报文的 type 值是 8，所以当监听的 icmp 报文的 type 不是 8 的时候可以直接结束。构造的 ICMP 报文为 ICMP 响应报文，type 为 0，id、seq、负载内容、长度与请求报文相同，源宿地址互换。这样就伪造好请求报文，发送即可。


Open
sniffAndSpoof.py
~/Desktop/Labs_20.04/Network Security/Pac...Sniffing and Spoofing Lab/Labsetup/volumes

```python
#!/usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
        if pkt[ICMP].type != 8:
                return

        ip = IP(src=pkt[IP].dst,dst=pkt[IP].src,ihl=pkt[IP].ihl)
        icmp = ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        newpkt = ip/icmp/data

        send(newpkt, verbose = 0)
        print("send spoof packet\n")

while 1:
        pkt = sniff(filter='icmp',prn = spoof_pkt)
```

下图是 ping 1.2.3.4（一个在互联上不存在的主机）

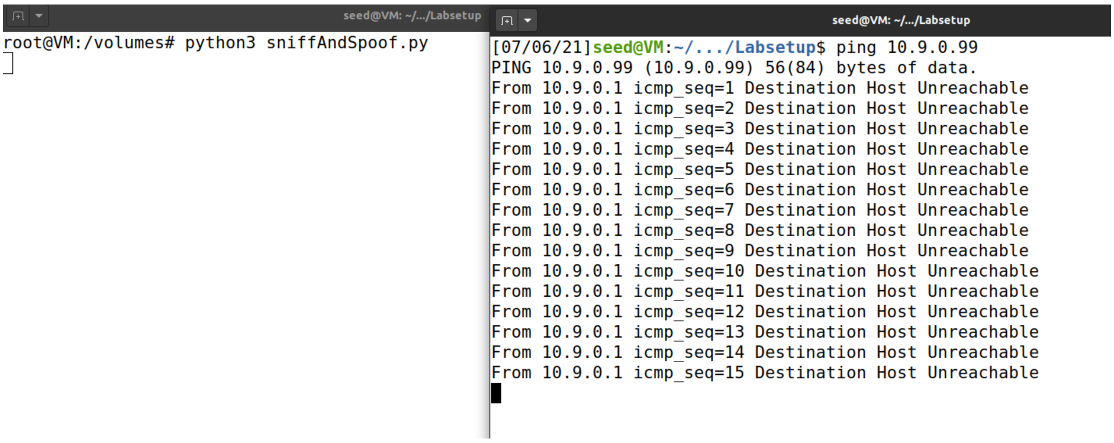


```
oot@VM:/volumes# python3 sniffAndSpoof.py
end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet

end spoof packet
```

```
[07/06/21]seed@VM:~/.../Labsetup$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=19.4 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=32.2 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=25.4 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=32.1 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=30.0 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=24.3 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=21.4 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=25.2 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=23.9 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=27.8 ms
64 bytes from 1.2.3.4: icmp_seq=11 ttl=64 time=23.4 ms
```

可以观察到伪造程序伪造了 ICMP 响应报文并发送给了 ping 该地址的主机，尽管该地址的主机并不存在。

下图是 ping 10.9.0.99（一个在局域网内不存在的主机）

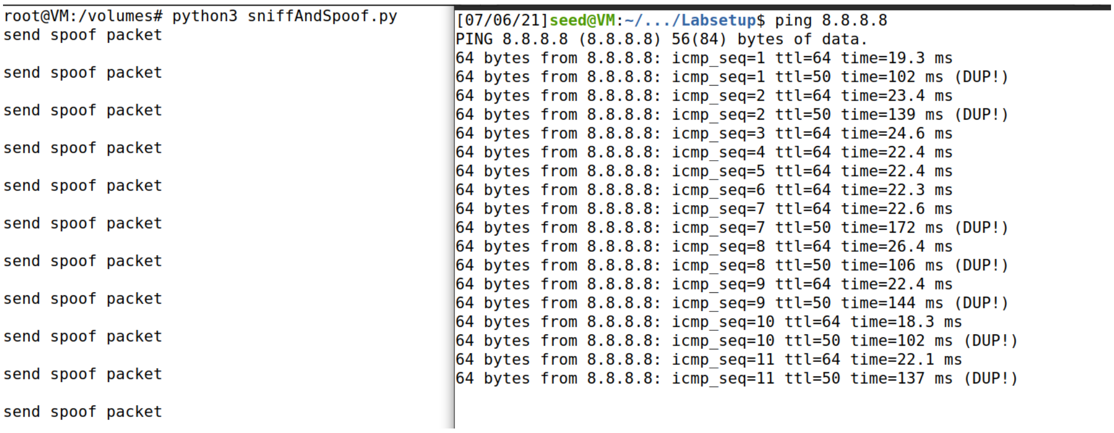


```
root@VM:/volumes# python3 sniffAndSpoof.py
```

```
[07/06/21]seed@VM:~/.../Labsetup$ ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Host Unreachable
From 10.9.0.1 icmp_seq=2 Destination Host Unreachable
From 10.9.0.1 icmp_seq=3 Destination Host Unreachable
From 10.9.0.1 icmp_seq=4 Destination Host Unreachable
From 10.9.0.1 icmp_seq=5 Destination Host Unreachable
From 10.9.0.1 icmp_seq=6 Destination Host Unreachable
From 10.9.0.1 icmp_seq=7 Destination Host Unreachable
From 10.9.0.1 icmp_seq=8 Destination Host Unreachable
From 10.9.0.1 icmp_seq=9 Destination Host Unreachable
From 10.9.0.1 icmp_seq=10 Destination Host Unreachable
From 10.9.0.1 icmp_seq=11 Destination Host Unreachable
From 10.9.0.1 icmp_seq=12 Destination Host Unreachable
From 10.9.0.1 icmp_seq=13 Destination Host Unreachable
From 10.9.0.1 icmp_seq=14 Destination Host Unreachable
From 10.9.0.1 icmp_seq=15 Destination Host Unreachable
```

可以观察到因为所在的局域网内并不存在该主机，所以 ping 之后返回了目的主机不可达的信息。

下图是 ping 8.8.8.8（一个在互联上存在的主机）



```
root@VM:/volumes# python3 sniffAndSpoof.py
send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet

send spoof packet
```

```
[07/06/21]seed@VM:~/.../Labsetup$ 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=19.3 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=50 time=102 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=23.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=50 time=139 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=24.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=22.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=22.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=22.3 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=22.6 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=50 time=172 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=26.4 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=50 time=106 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=9 ttl=64 time=22.4 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=50 time=144 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=10 ttl=64 time=18.3 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=50 time=102 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=11 ttl=64 time=22.1 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=50 time=137 ms (DUP!)
```

可以观察到 ping 该地址后伪造程序伪造了响应报文。

ping 的报文应该需要经过 attacker 的代表的网关，所以到达互联网的数据报文会被程序监听，然后伪造响应报文，而发往内网的报文不需要发往网关，所以不会被监听并伪造。

总结:

本次实验介绍了使用 python 及 scapy 进行数据报文的监听和伪造, 数据报文的监听利用 BPF 进行过滤是一个非常重要且常用的技术，而伪造数据报文则是攻击者经常做的事情，通过对这次的实验，对数据报文的监听和伪造有了更深的理解，对于从事网络安全方面以及防止被攻击者进行欺骗，这些概念尤其重要。