

# LAB1:Packet\_Sniffing\_Spoofing

学号: 57118227 姓名: 孙浩 日期: 2021年7月9日

## Task 1.1 Sniffing Packets

### Task 1.1A

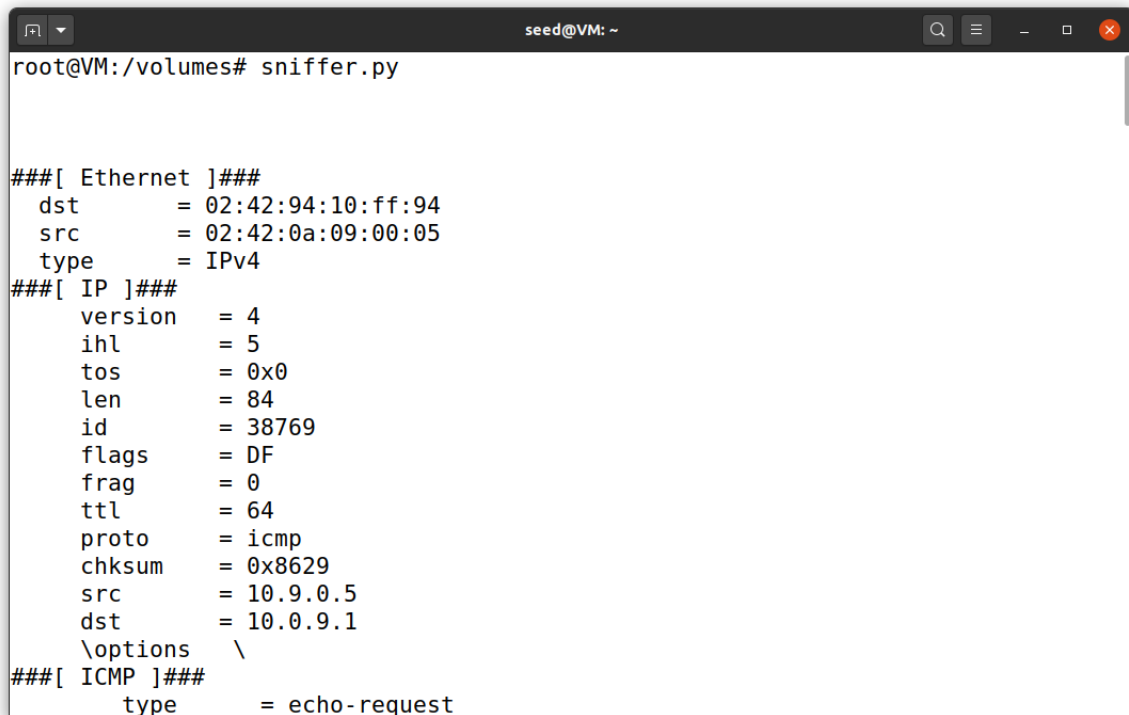
通过volumes文件夹将如下的代码送进docker容器seed-attacker内。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-b48035aa9cc3',filter='icmp',prn=print_pkt)
```

进入seed-attacker后以root权限运行以上代码，可以看到程序进入了监听状态，这时可以进入另一个容器host中向seed-attacker进行PING，发送ICMP报文，这时在seed-attacker上可以发现该程序已经监听到了相应的报文。



```
seed@VM: ~
root@VM:/volumes# sniffer.py

###[ Ethernet ]###
  dst      = 02:42:94:10:ff:94
  src      = 02:42:0a:09:00:05
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 38769
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x8629
  src      = 10.9.0.5
  dst      = 10.0.9.1
  \options \
###[ ICMP ]###
  type     = echo-request
```

切换为普通用户状态，再次运行程序，可以发现权限不够导致运行失败。

```
seed@VM:/volumes$ sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 7, in <module>
    pkt = sniff(iface='br-b48035aa9cc3',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, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.htons(type)) # 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
seed@VM:/volumes$
```

## Task 1.1B

- Capture only the ICMP packet

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-b48035aa9cc3',filter='icmp',prn=print_pkt)
```

运行结果见[上一部分](#)

- Capture any TCP packet that comes from a particular IP and with a destination port number 23

端口号为23，即telnet服务。可以IP选择为10.9.0.5，然后进入host容器对seed-attacker进行telnet测试。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-b48035aa9cc3',filter='tcp and src host 10.9.0.5 and dst port 23',prn=print_pkt)
```

在host10.9.0.1上进行telnet 10.9.0.5。

```
seed@VM: ~
root@ca6e3c1784e9:/# telnet 10.9.0.1
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: █
```

在seed-attacker的监听程序上可以看到相关报文。

```
seed@VM: ~
root@VM:/volumes# sniffer.py
###[ Ethernet ]###
  dst      = 02:42:94:10:ff:94
  src      = 02:42:0a:09:00:05
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 60
  id       = 60559
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  chksum   = 0x3a05
  src      = 10.9.0.5
  dst      = 10.9.0.1
  \options \
###[ TCP ]###
  sport    = 54984
  dport    = telnet
  seq      = 3352518757
  ack      = 0
  dataofs  = 10
  reserved = 0
  flags    = S
  window   = 64240
  chksum   = 0x1446
  urgptr   = 0
  options  = [('MSS', 1460), ('SAckOK', b''), ('Timestamp',
(259039149, 0)), ('NOP', None), ('WScale', 7)]
```

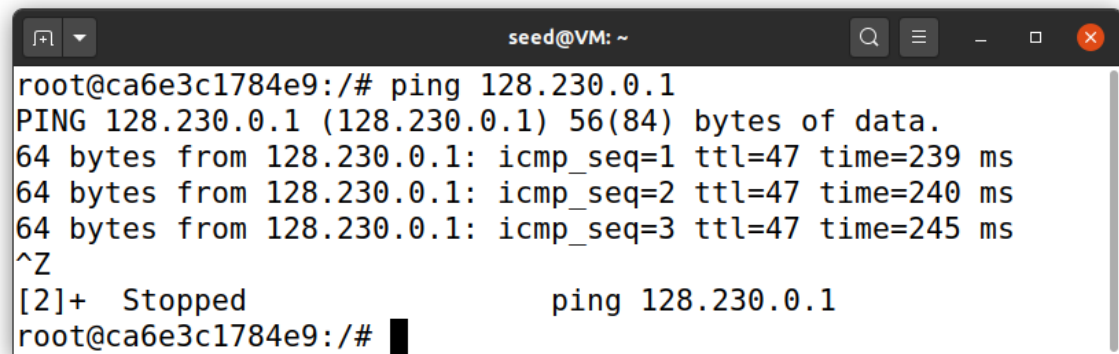
- Capture packets comes from or to go to a particular subnet. (128.230.0.0/16)

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-b48035aa9cc3', filter='net 128.230.0.0/16', prn=print_pkt)
```

在host上面进行ping128.230.0.1进行测试。

A terminal window titled 'seed@VM: ~' with standard window controls. The terminal shows a user at the root prompt running 'ping 128.230.0.1'. The output shows three successful ping requests with 64 bytes of data, TTL of 47, and response times around 240ms. The user then presses Ctrl-Z (^Z), which results in '[2]+ Stopped ping 128.230.0.1' and returns to the root prompt.

```
seed@VM: ~
root@ca6e3c1784e9:/# ping 128.230.0.1
PING 128.230.0.1 (128.230.0.1) 56(84) bytes of data.
64 bytes from 128.230.0.1: icmp_seq=1 ttl=47 time=239 ms
64 bytes from 128.230.0.1: icmp_seq=2 ttl=47 time=240 ms
64 bytes from 128.230.0.1: icmp_seq=3 ttl=47 time=245 ms
^Z
[2]+  Stopped                  ping 128.230.0.1
root@ca6e3c1784e9:/#
```

该程序可以监听到相应的报文。

```
seed@VM: ~
root@VM:/volumes# sniffer.py
###[ Ethernet ]###
  dst      = 02:42:94:10:ff:94
  src      = 02:42:0a:09:00:05
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 22842
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x567a
  src      = 10.9.0.5
  dst      = 128.230.0.1
  \options \
###[ ICMP ]###
  type     = echo-request
  code     = 0
  chksum   = 0x91ba
  id       = 0x27
  seq      = 0x6
```

## Task 1.2 Spoofing ICMP Packets

在seed-attacker上使用scapy进行spoof, 设定源IP地址为27.27.27.27, 代码如下。

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

a = IP()
a.src = '27.27.27.27'
a.dst = '10.9.0.5'
p = a/ICMP()
send(p)
```

```
seed@VM: ~
root@VM:/volumes# chmod a+x spoof.py
root@VM:/volumes# spoof.py
.
Sent 1 packets.
root@VM:/volumes#
```

利用Wireshark进行抓包，可以看见host向伪造的IP地址27.27.27.27发出了reply。

No.	Time	Source	Destination	Protocol	Length	Info
3	2021-0...	27.27.27.27	10.9.0.5	ICMP	42	Echo (ping) request
4	2021-0...	10.9.0.5	27.27.27.27	ICMP	42	Echo (ping) reply

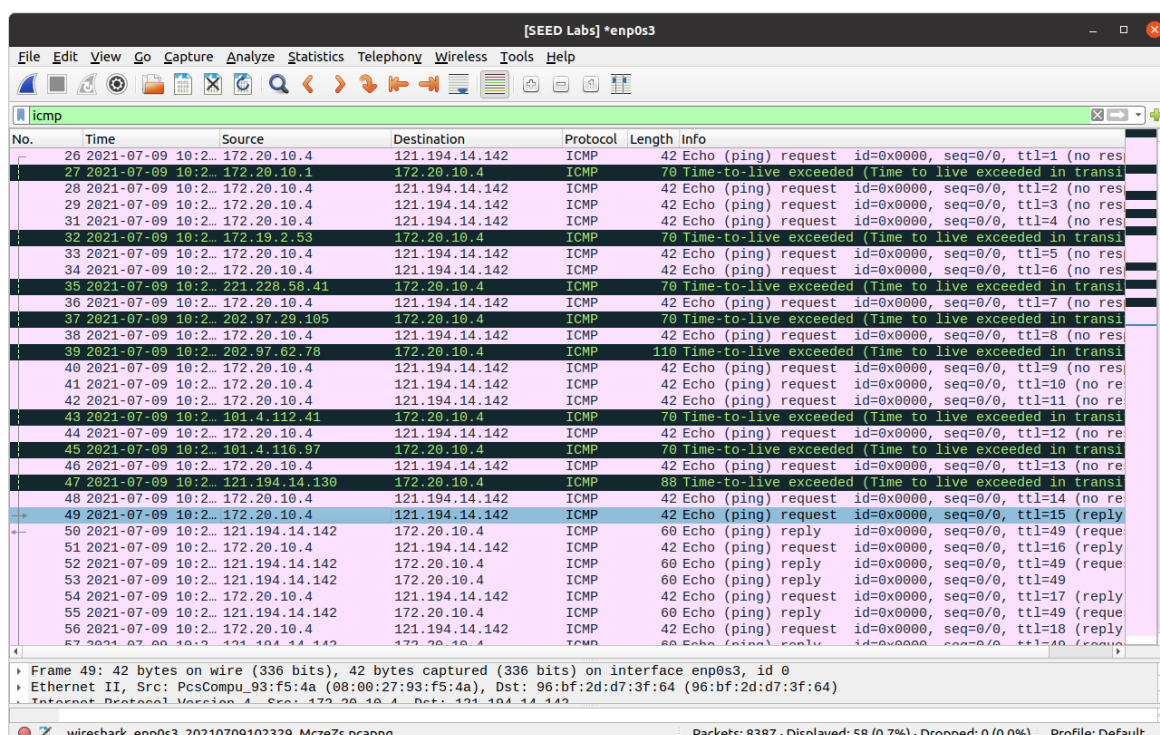
## Task 1.3 Traceroute

对东南大学校园主页[www.seu.edu.cn](http://www.seu.edu.cn)(121.194.14.142)进行测试。

```
from scapy.all import *

a = IP()
a.dst = '121.194.14.142'
for i in range(1,32):
    a.ttl=i
    b = ICMP()
    send(a/b)
```

使用wireshark进行观察，确定距离为15，因为在TTL大于等于15时才收到reply。



## Task 1.4 Sniffing and-then Spoofing

在VM上运行的代码如下。然后再在10.9.0.5上进行PING。

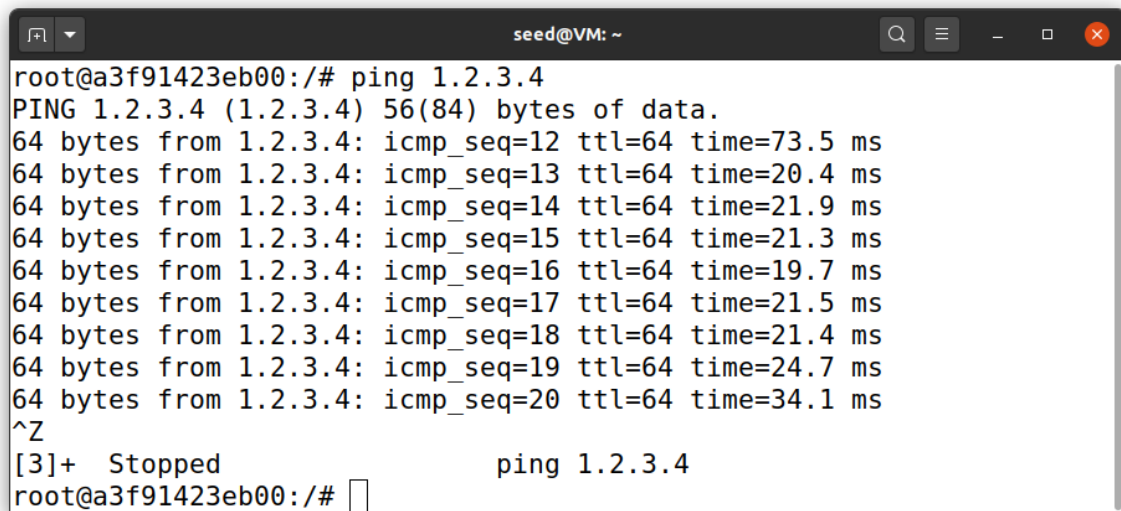
```
from scapy.all import *

def spoof(pkt):
    a = IP()
    a.src = pkt[IP].dst
    a.dst = '10.9.0.5'
    b = ICMP()
    b.type = 'echo-reply'
    b.code = 0
    b.id = pkt[ICMP].id
    b.seq = pkt[ICMP].seq
    c = pkt[Raw].load
    send(a/b/c)

pkt = sniff(iface='br-f3fd57bc6151', filter='icmp and src host
10.9.0.5', prn=spoof)
```

- ping 1.2.3.4

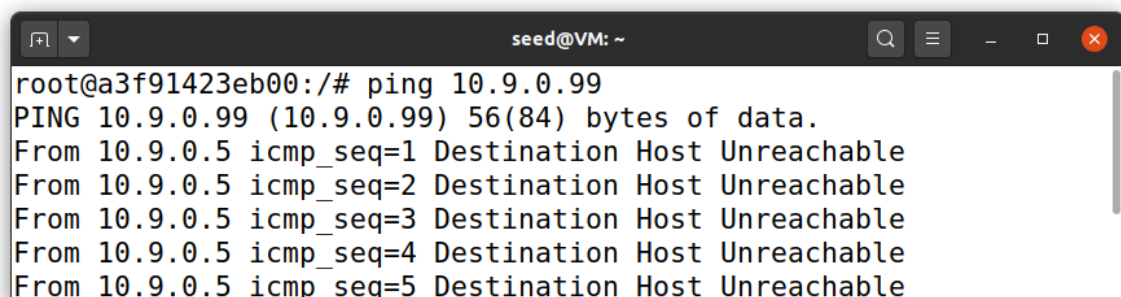
成功ping到一个不存在的地址



```
seed@VM: ~
root@a3f91423eb00:/# 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=12 ttl=64 time=73.5 ms
64 bytes from 1.2.3.4: icmp_seq=13 ttl=64 time=20.4 ms
64 bytes from 1.2.3.4: icmp_seq=14 ttl=64 time=21.9 ms
64 bytes from 1.2.3.4: icmp_seq=15 ttl=64 time=21.3 ms
64 bytes from 1.2.3.4: icmp_seq=16 ttl=64 time=19.7 ms
64 bytes from 1.2.3.4: icmp_seq=17 ttl=64 time=21.5 ms
64 bytes from 1.2.3.4: icmp_seq=18 ttl=64 time=21.4 ms
64 bytes from 1.2.3.4: icmp_seq=19 ttl=64 time=24.7 ms
64 bytes from 1.2.3.4: icmp_seq=20 ttl=64 time=34.1 ms
^Z
[3]+  Stopped                  ping 1.2.3.4
root@a3f91423eb00:/#
```

- ping 10.9.0.99

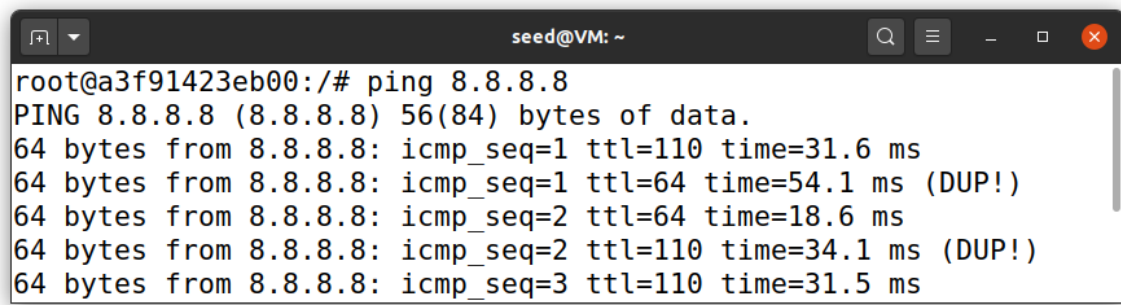
ping命令失败，因为在LAN，会首先进行ARP（who is 10.9.0.99），所以上述程序无法进行spoof。



```
seed@VM: ~
root@a3f91423eb00:/# 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
```

- ping 8.8.8.8

会收到伪造的回复和真正的回复，造成DUP!警告。

A terminal window titled 'seed@VM: ~' showing the output of a ping command to 8.8.8.8. The output shows five responses, with the second one marked as a duplicate (DUP!).

```
root@a3f91423eb00:/# 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=110 time=31.6 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=64 time=54.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=18.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=34.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=31.5 ms
```