

# Lab1

57118233 周厚霖

## Task 1.1: Sniffing Packets

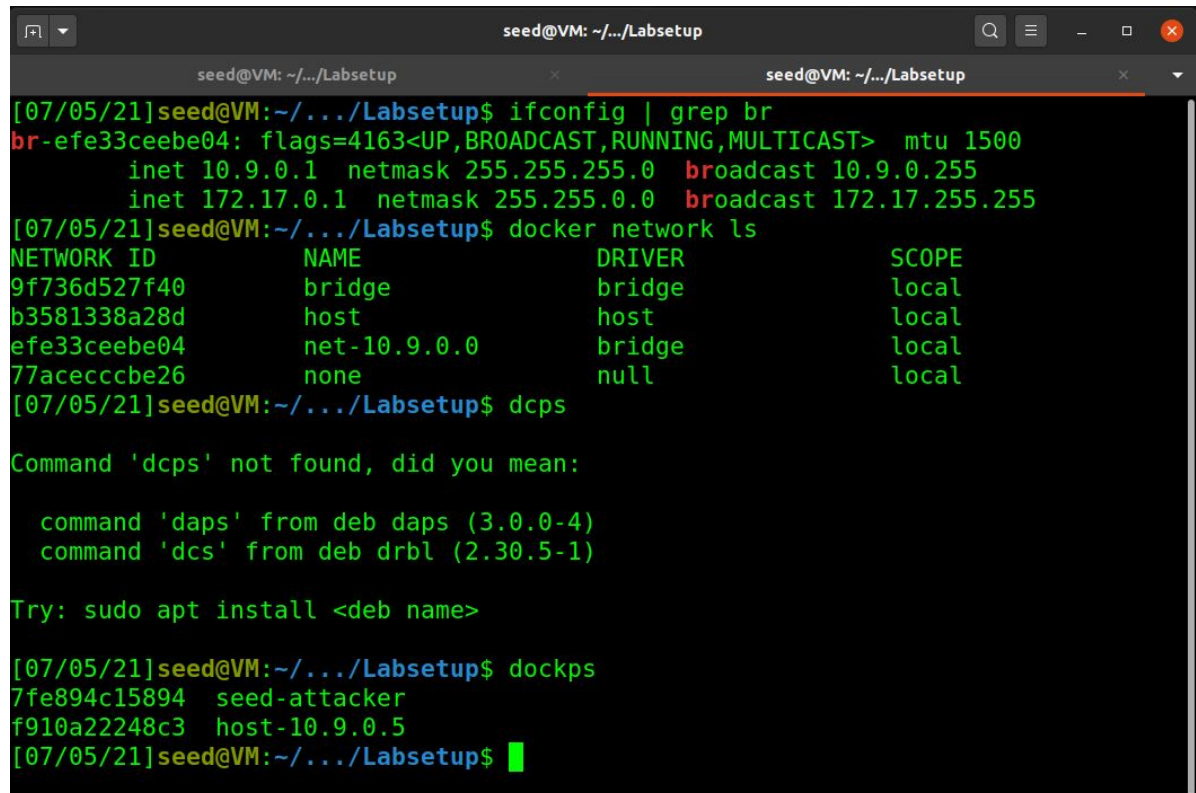
### Task 1.1A

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

def print_pkt(pkt):
    pkt.show()

pkt = sinff(iface='br-efe33ceebe04', filter='icmp', prn=print_pkt)
```

启动docker后，查看网络ID。



```
seed@VM: ~/.../Labsetup
[07/05/21]seed@VM:~/.../Labsetup$ ifconfig | grep br
br-efe33ceebe04: 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
[07/05/21]seed@VM:~/.../Labsetup$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9f736d527f40        bridge             bridge             local
b3581338a28d        host               host               local
efe33ceebe04        net-10.9.0.0       bridge             local
77acecccbe26        none              null               local
[07/05/21]seed@VM:~/.../Labsetup$ dcps

Command 'dcps' not found, did you mean:

  command 'daps' from deb daps (3.0.0-4)
  command 'dcs' from deb drbl (2.30.5-1)

Try: sudo apt install <deb name>

[07/05/21]seed@VM:~/.../Labsetup$ dockps
7fe894c15894  seed-attacker
f910a22248c3  host-10.9.0.5
[07/05/21]seed@VM:~/.../Labsetup$
```

以root权限运行sniffer.py，新开一个命令行对主机IP进行ping命令。

```
seed@VM: ~/Desktop
[07/05/21]seed@VM:~/Desktop$ chmod a+x sniffer.py
[07/05/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
#### [ Ethernet ] ####
    dst      = 02:42:0a:09:00:05
    src      = 02:42:fd:5f:39:e5
    type     = IPv4
#### [ IP ] ####
    version  = 4
    ihl      = 5
    tos      = 0x0
    len      = 84
    id       = 18209
    flags    = DF
    frag     = 0
    ttl      = 64
    proto    = icmp
    chksum   = 0xdf70
    src      = 10.9.0.1
    dst      = 10.9.0.5
    \options \
#### [ ICMP ] ####
    type     = echo-request
    code     = 0
    chksum   = 0xf497
```

```
seed@VM: ~/Desktop
[07/05/21]seed@VM:~/Desktop$ ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.129 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.235 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.067 ms
64 bytes from 10.9.0.5: icmp_seq=4 ttl=64 time=0.065 ms
64 bytes from 10.9.0.5: icmp_seq=5 ttl=64 time=0.085 ms
64 bytes from 10.9.0.5: icmp_seq=6 ttl=64 time=0.071 ms
64 bytes from 10.9.0.5: icmp_seq=7 ttl=64 time=0.094 ms
64 bytes from 10.9.0.5: icmp_seq=8 ttl=64 time=0.056 ms
64 bytes from 10.9.0.5: icmp_seq=9 ttl=64 time=0.050 ms
64 bytes from 10.9.0.5: icmp_seq=10 ttl=64 time=0.066 ms
64 bytes from 10.9.0.5: icmp_seq=11 ttl=64 time=0.048 ms
64 bytes from 10.9.0.5: icmp_seq=12 ttl=64 time=0.098 ms
64 bytes from 10.9.0.5: icmp_seq=13 ttl=64 time=0.069 ms
64 bytes from 10.9.0.5: icmp_seq=14 ttl=64 time=0.086 ms
64 bytes from 10.9.0.5: icmp_seq=15 ttl=64 time=0.092 ms
64 bytes from 10.9.0.5: icmp_seq=16 ttl=64 time=0.093 ms
64 bytes from 10.9.0.5: icmp_seq=17 ttl=64 time=0.114 ms
64 bytes from 10.9.0.5: icmp_seq=18 ttl=64 time=0.066 ms
64 bytes from 10.9.0.5: icmp_seq=19 ttl=64 time=0.053 ms
64 bytes from 10.9.0.5: icmp_seq=20 ttl=64 time=0.155 ms
64 bytes from 10.9.0.5: icmp_seq=21 ttl=64 time=0.153 ms
64 bytes from 10.9.0.5: icmp_seq=22 ttl=64 time=0.161 ms
```

以seed用户运行sniffer.py时，系统会报错。

```
seed@VM: ~/Desktop
00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

^C[07/05/21]seed@VM:~/Desktop$
[07/05/21]seed@VM:~/Desktop$ su seed
Password:
[07/05/21]seed@VM:~/Desktop$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 7, in <module>
    pkt = sniff(iface='br-efe33ceebe04', 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
[07/05/21]seed@VM:~/Desktop$
```

## Task 1.1B

只抓取ICMP报文，见Task 1.1A所示。

捕获任何来自特定IP的TCP数据包，目的端口为23。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-efe33ceebe04', filter='tcp port 23 and host 10.9.0.5',
prn=print_pkt)
```

利用docksh获取host的shell，telnet任意一个IP地址建立连接。

```
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
br-efe33ceebe04: 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
[07/05/21]seed@VM:~/.../Labsetup$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
9f736d527f40        bridge             bridge              local
b3581338a28d        host               host                local
efe33ceebe04        net-10.9.0.0       bridge              local
77acecccbe26        none              null                local
[07/05/21]seed@VM:~/.../Labsetup$ dcps

Command 'dcps' not found, did you mean:

  command 'daps' from deb daps (3.0.0-4)
  command 'dcs' from deb drbl (2.30.5-1)

Try: sudo apt install <deb name>

[07/05/21]seed@VM:~/.../Labsetup$ dockps
7fe894c15894 seed-attacker
f910a22248c3 host-10.9.0.5
[07/05/21]seed@VM:~/.../Labsetup$ docksh f9
root@f910a22248c3:/# telnet 1.1.1.1
Trying 1.1.1.1...
telnet: Unable to connect to remote host: Network is unreachable
root@f910a22248c3:/#
```

在另一处可看到tcp数据包。

```
seed@VM: ~/Desktop
[07/05/21]seed@VM:~/Desktop$ vi sniffer.py
[07/05/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst      = 02:42:fd:5f:39:e5
  src      = 02:42:0a:09:00:05
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x10
  len      = 60
  id       = 64685
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = tcp
  chksum   = 0x31ef
  src      = 10.9.0.5
  dst      = 1.1.1.1
  \options \
###[ TCP ]###
  sport    = 46984
  dport    = telnet
  seq      = 927616116
  ack      = 0
  dataofs  = 10
  reserved = 0
  flags    = S
  window   = 64240
  chksum   = 0xc3e
  urgptr   = 0
  options  = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (1328902202, 0)), ('NOP', None), ('WScale', 7)]
###[ Ethernet 1 ]###
```

捕获来自或去特定子网的数据包。可以选择任何子网，如128.230.0.0/16；不应该选择VM所绑定的子网。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(iface='br-13a5b79724e2', filter='host 10.9.0.8', prn=print_pkt)
```



直接ping 10.9.0.8，可得到捕获的数据包。

```
seed@VM: ~/.../Labsetup
[07/05/21]seed@VM:~/.../Labsetup$ ping 10.9.0.8
PING 10.9.0.8 (10.9.0.8) 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
From 10.9.0.1 icmp_seq=16 Destination Host Unreachable
From 10.9.0.1 icmp_seq=17 Destination Host Unreachable
From 10.9.0.1 icmp_seq=18 Destination Host Unreachable
From 10.9.0.1 icmp_seq=19 Destination Host Unreachable
From 10.9.0.1 icmp_seq=20 Destination Host Unreachable
From 10.9.0.1 icmp_seq=21 Destination Host Unreachable
From 10.9.0.1 icmp_seq=22 Destination Host Unreachable
From 10.9.0.1 icmp_seq=23 Destination Host Unreachable
From 10.9.0.1 icmp_seq=24 Destination Host Unreachable
From 10.9.0.1 icmp_seq=25 Destination Host Unreachable
From 10.9.0.1 icmp_seq=26 Destination Host Unreachable
From 10.9.0.1 icmp_seq=27 Destination Host Unreachable
From 10.9.0.1 icmp_seq=28 Destination Host Unreachable
From 10.9.0.1 icmp_seq=29 Destination Host Unreachable
From 10.9.0.1 icmp_seq=30 Destination Host Unreachable

seed@VM: ~/Desktop
[07/05/21]seed@VM:~/Desktop$ vi sniffer.py
[07/05/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 02:42:38:ae:21:39
  type     = ARP
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = 6
  plen     = 4
  op       = who-has
  hwsrsrc  = 02:42:38:ae:21:39
  psrsrc   = 10.9.0.1
  hwdst    = 00:00:00:00:00:00
  pdst     = 10.9.0.8
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
  src      = 02:42:38:ae:21:39
  type     = ARP
###[ ARP ]###
  hwtype   = 0x1
  ptype    = IPv4
  hwlen    = 6
  plen     = 4
  op       = who-has
  hwsrsrc  = 02:42:38:ae:21:39
  psrsrc   = 10.9.0.1
  hwdst    = 00:00:00:00:00:00
  pdst     = 10.9.0.8
###[ Ethernet ]###
  dst      = ff:ff:ff:ff:ff:ff
```

## Task 1.2: Spoofing ICMP Packets

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
a.dst = '10.9.0.3'
b = ICMP()
p = a/b
send(p)
ls(a)
```

第一行创建了一个ICMP对象，默认类型为echo request。在第六行中，我们将a和b堆叠在一起形成了一个新对象，“/”操作符被重载，不在表示除法，而是将b添加为a的有效负载字段，并相应地修改a的字段。最终我们得到一个表示ICMP数据包的新对象，报文重组后，向子网内的一个IP发送数据包，打开Wireshark可观测发送数据包和响应数据包。

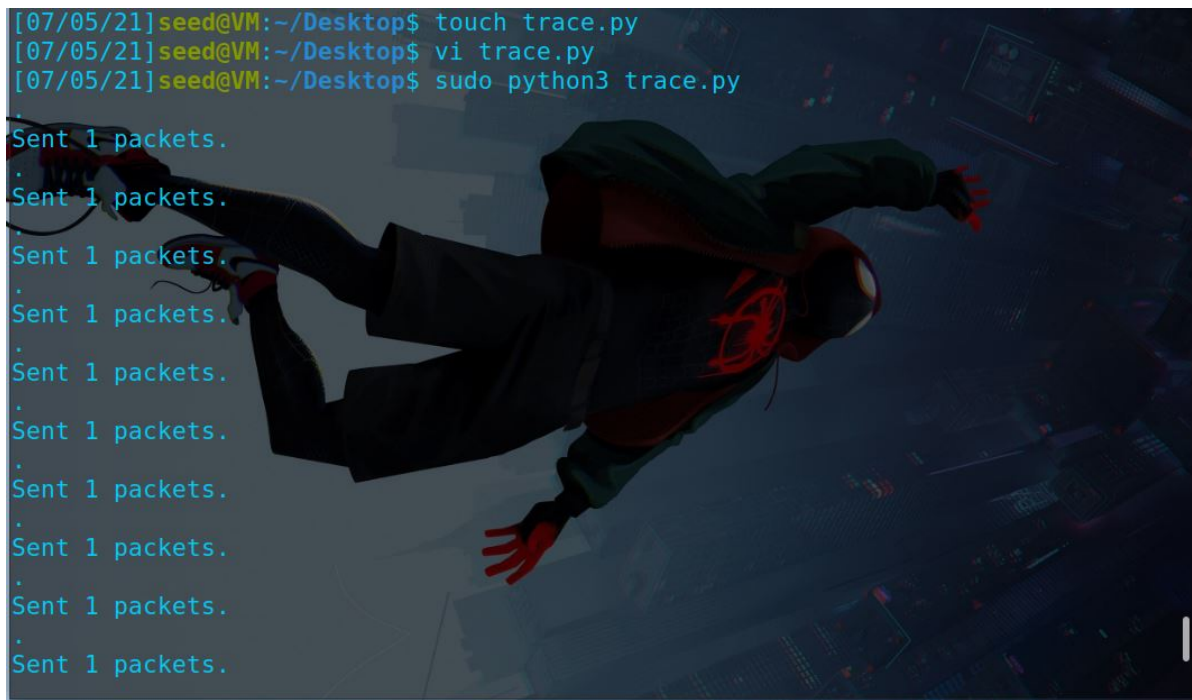
```
[07/05/21]seed@VM:~/Desktop$ sudo python3 spoofer.py
Sent 1 packets.
version      : BitField  (4 bits)      = 4      (4)
ihl          : BitField  (4 bits)      = None   (None)
tos          : XByteField = 0      (0)
len          : ShortField = None   (None)
id           : ShortField = 1      (1)
flags        : FlagsField (3 bits)     = <Flag 0 (>) (<Flag 0 (>))
frag         : BitField  (13 bits)     = 0      (0)
ttl          : ByteField  = 64      (64)
proto        : ByteEnumField = 0      (0)
chksum       : XShortField = None   (None)
src          : SourceIPField = '10.9.0.1' (None)
dst          : DestIPField = '10.9.0.5' (None)
options      : PacketListField = []     ([])
[07/05/21]seed@VM:~/Desktop$
```

[SEED Labs] *any									
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help									
icmp									
No.	Time	Source	Destination	Protocol	Length	Info			
30	2021-07-05 19:0...	10.9.0.1	10.9.0.5	ICMP	44	Echo (ping) request	id=0x0000, seq=0/0, ttl=64	(no response ...	
31	2021-07-05 19:0...	10.9.0.1	10.9.0.5	ICMP	44	Echo (ping) request	id=0x0000, seq=0/0, ttl=64	(reply in 32)	
32	2021-07-05 19:0...	10.9.0.5	10.9.0.1	ICMP	44	Echo (ping) reply	id=0x0000, seq=0/0, ttl=64	(request in 3...	
33	2021-07-05 19:0...	10.9.0.5	10.9.0.1	ICMP	44	Echo (ping) reply	id=0x0000, seq=0/0, ttl=64		

## Task 1.3: Traceroute

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
b = ICMP()
a.dst = '1.2.3.4'
for i in range(30):
    a.ttl = i + 1
    p = a / b
    send(p)
```

创建一个文件trace.py，向目标IP发送 ICMP 数据包，一开始设置 TTL (Time-To-Live)值为 1，那么发出的 ICMP 数据包在经历一个路由结点后，就会失活被抛弃，我们利用循环，不断增加TTL的值，最终使得数据包到达目的地。



从Wireshark中我们可以观察到，途径的IP有172.20.10.1，172.20.73.13，172.18.0.5，最后到达1.2.3.4，即目的地。

[SEED Labs] *any						
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help						
icmp						
No.	Time	Source	Destination	Protocol	Length	Info
34	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response f...
35	2021-07-05 10:0...	172.20.10.1	172.20.10.8	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
36	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response f...
37	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response f...
38	2021-07-05 10:0...	172.20.73.13	172.20.10.8	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
39	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response f...
40	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response f...
41	2021-07-05 10:0...	172.20.10.5	172.20.10.8	ICMP	72	Time-to-live exceeded (Time to live exceeded in transit)
42	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response f...
43	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response f...
44	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response f...
45	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response f...
46	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response f...
47	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response ...
48	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=12 (no response ...
49	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=13 (no response ...
50	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=14 (no response ...
51	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=15 (no response ...
52	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=16 (no response ...
53	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=17 (no response ...
54	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=18 (no response ...
55	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=19 (no response ...
56	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=20 (no response ...
57	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=21 (no response ...
58	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=22 (no response ...
59	2021-07-05 10:0...	172.20.10.8	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=23 (no response ...

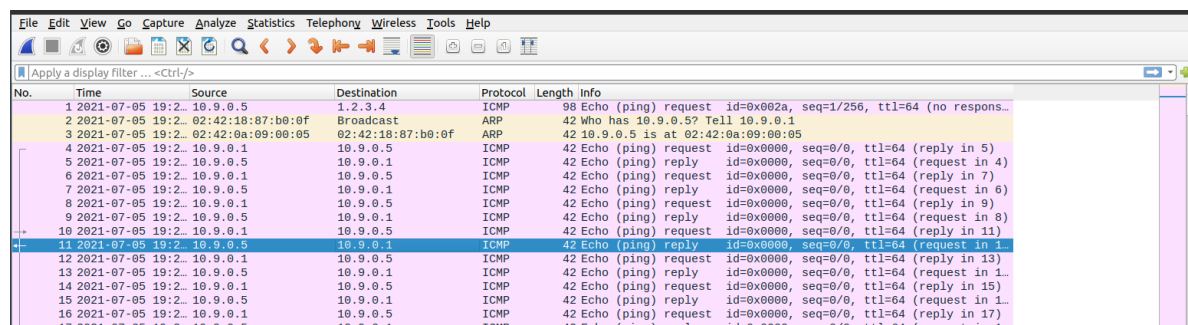
## Task 1.4: Sniffing and-then Spoofing

```
#!/usr/bin/evn python3
from scapy.all import *
def spoof_pkt(pkt):
    a = IP()
    b = ICMP()
    a.dst = '10.9.0.5'
    p = a/b
    send(p)

pkt = sniff(iface='br-6dfc36db5662', filter='icmp and host 10.9.0.5',
prn=spoof_pkt)
```



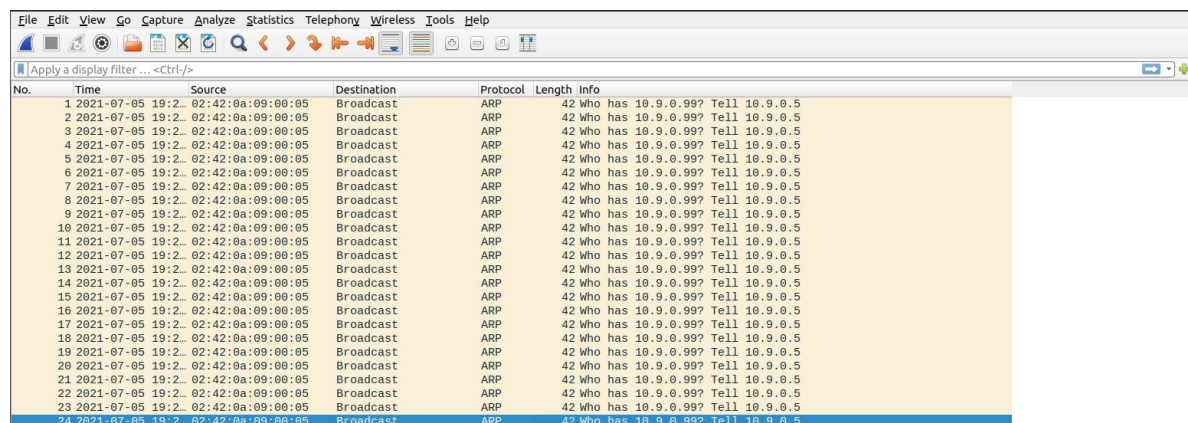
## ping 1.2.3.4 # a non-existing host on the Internet



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-05 19:22:02.421887	10.9.0.5	1.2.3.4	ICMP	98	Echo (ping) request id=0x002a, seq=1/256, ttl=64 (no response yet)
2	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
3	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
4	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5)
5	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 4)
6	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 7)
7	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 6)
8	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 9)
9	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 8)
10	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 11)
11	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 10)
12	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 13)
13	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 12)
14	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 15)
15	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 14)
16	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 17)
17	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 16)

通过Wireshark可以看到，对于互联网上不存的主机，先利用MAC地址进行广播，告知目标和本地MAC地址相对应的IP地址，然后出现了报文欺骗。

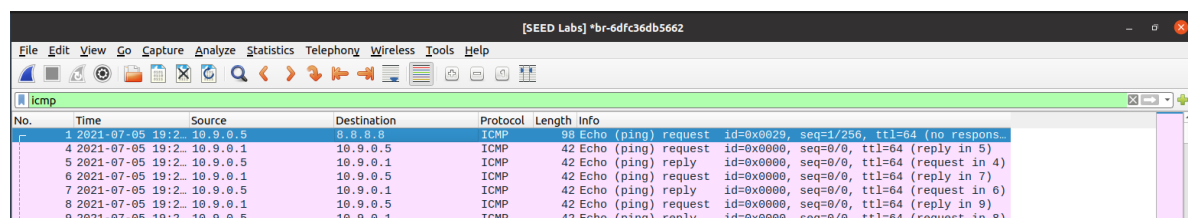
## ping 10.9.0.99 # a non-existing host on the LAN



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
2	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
3	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
4	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
5	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
6	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
7	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
8	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
9	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
10	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
11	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
12	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
13	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
14	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
15	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
16	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
17	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
18	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
19	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
20	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
21	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
22	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
23	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5
24	2021-07-05 19:22:02.421887	10.9.0.5	Broadcast	ARP	42	Who has 10.9.0.99? Tell 10.9.0.5

对于局域网内不存在的主机，一直在利用MAC地址进行广播，但是得不到响应。

## ping 8.8.8.8 # an existing host on the Internet



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-05 19:22:02.421887	10.9.0.5	8.8.8.8	ICMP	98	Echo (ping) request id=0x0029, seq=1/256, ttl=64 (no response yet)
4	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 5)
5	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 4)
6	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 7)
7	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 6)
8	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 9)
9	2021-07-05 19:22:02.421887	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 8)

IP为10.9.0.1的VM将监听LAN中的通信，嗅探到10.9.0.5的ICMP请求包的时候，就会自动发送一个伪造的ICMP响应包进行欺骗，通过Wireshark前三行可以很清晰地看到报文欺骗。