

# Lab 1 Packet Sniffing and Spoofing Lab

57118115 陈烨

## Task 1.1 Sniffing Packets

### Task 1.1A

```
[07/05/21]seed@VM:~/.../Labsetup$ pwd
/home/seed/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab/Labsetup
[07/05/21]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  sniffer.py  volumes
```

sniffer.py

```
1  #!/usr/bin/env python3
2  from scapy.all import *
3
4  def print_pkt(pkt):
5      pkt.show()
6
7  #pkt = sniff(iface='br-68c809615df2',filter='icmp',prn=print_pkt)
8  #pkt = sniff(filter='icmp',prn=print_pkt)
9  #ens33
10 pkt = sniff(iface='ens33',filter='icmp',prn=print_pkt)
11
```

其中虚拟机采用NAT，网卡名称使用ifconfig查看

运行sniffer.py seed@VM:~/.../Labsetup\$ sudo ./sniffer.py

使用另一terminal执行ping命令，sniffer.py获取报文

```
[07/05/21]seed@VM:~/.../Labsetup$ sudo ./sniffer.py
###[ Ethernet ]###
  dst      = 00:50:56:ff:a5:ad
  src      = 00:0c:29:6e:0e:ec
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 84
  id       = 7103
  flags    = DF
  frag     = 0
  ttl      = 64
  proto    = icmp
  chksum   = 0x6b61
  src      = 192.168.239.150
  dst      = 220.181.38.148
  \options \
###[ ICMP ]###

p_seq=19 ttl=128 time=25.7 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=20 ttl=128 time=26.3 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=21 ttl=128 time=26.3 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=22 ttl=128 time=26.1 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=23 ttl=128 time=26.6 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=24 ttl=128 time=25.7 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=25 ttl=128 time=26.1 ms
64 bytes from 220.181.38.148 (220.181.38.148): icmp
p_seq=26 ttl=128 time=26.6 ms
```

观察到root权限下可以正常抓包，普通用户权限下显示权限不够

```
[07/05/21]seed@VM:~/.../Labsetup$ ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 10, in <module>
    pkt = sniff(iface='ens33',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: E
501
  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 BPF

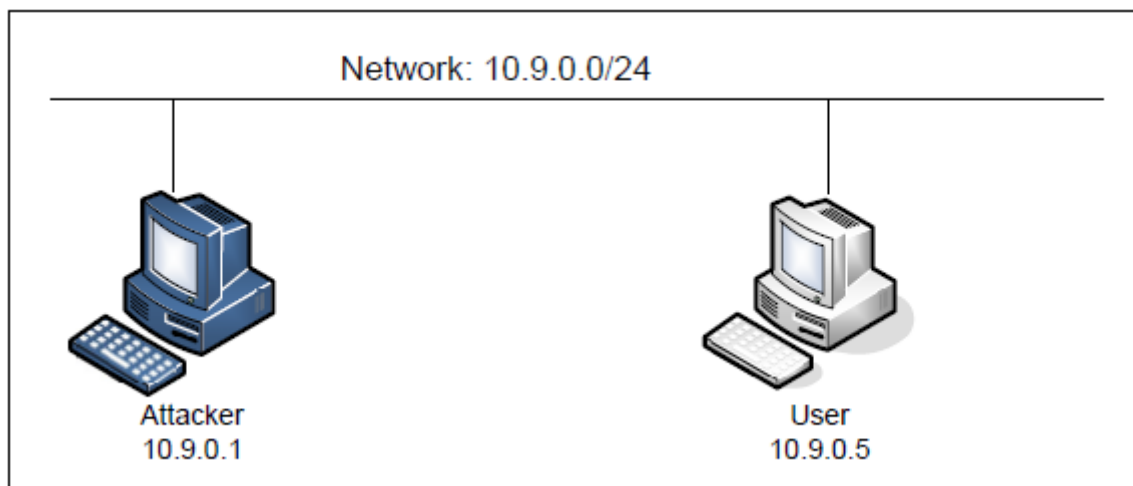
应用BPF语法来对scapy抓包进行过滤，分别满足

- Capture only the ICMP packet

filter与原代码一致，在这里使用docker

```
[07/05/21]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  sniffer.py  volumes
[07/05/21]seed@VM:~/.../Labsetup$ dockps
[07/05/21]seed@VM:~/.../Labsetup$ dcbuild
attacker uses an image, skipping
host uses an image, skipping
[07/05/21]seed@VM:~/.../Labsetup$ dcup
Starting seed-attacker ... done
Starting host-10.9.0.5 ... done
Attaching to seed-attacker, host-10.9.0.5
[07/05/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ cd Labs
etup/
[07/05/21]seed@VM:~/.../Labsetup$ ls
docker-compose.yml  sniffer.py  volumes
[07/05/21]seed@VM:~/.../Labsetup$ dockps
362310dfaef7  host-10.9.0.5
4ae6a2c8d91b  seed-attacker
[07/05/21]seed@VM:~/.../Labsetup$ docksh 4a
root@VM:/# ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
^C
--- 10.9.0.5 ping statistics ---
12 packets transmitted, 0 received, 100% packet loss, time 11273m
s
root@VM:/#
```

开启host和attacker并在attacker上ping主机host的IP



sniffer.py获取报文如下：

```

1  ###[ Ethernet ]###
2      dst      = 00:50:56:ff:a5:ad
3      src      = 00:0c:29:6e:0e:ec
4      type     = IPv4
5  ###[ IP ]###
6      version  = 4
7      ihl      = 5
8      tos      = 0x0
9      len      = 84
10     id       = 12878
11     flags    = DF
12     frag     = 0
13     ttl      = 64
14     proto    = icmp
15     checksum = 0x4e0e
16     src      = 192.168.239.150
17     dst      = 10.9.0.5
18     \options \
19  ###[ ICMP ]###
20     type     = echo-request
21     code     = 0
22     checksum = 0xd05c
23     id       = 0x5
24     seq      = 0x6
25  ###[ Raw ]###
26     load     =
27     'x\xba\xe2`\x00\x00\x00\x00#\xaa\n\x00\x00\x00\x00\x00\x10\x11\x12\x13\x14\x
    15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#${%&\'()*+,-./01234567'

```

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

将filter写为

```

1  pkt = sniff(iface='ens33',filter='src host 192.168.10.2 and tcp dst port
    23',prn=print_pkt)

```

## 无线局域网适配器 WLAN:

```
连接特定的 DNS 后缀 . . . . . : wifi
本地链接 IPv6 地址. . . . . : fe80::38d0:ba3d:5ab7:4ce3%19
IPv4 地址 . . . . . : 192.168.10.2
子网掩码 . . . . . : 255.255.255.0
默认网关. . . . . : 192.168.10.1
```

虚拟机改为桥接，IP为

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.31 netmask 255.255.255.0 broadcast 192.
168.10.255
    inet6 fe80::22ff:218e:cee2:425d prefixlen 64 scopeid 0x
20<link>
    ether 00:0c:29:6e:0e:ec txqueuelen 1000 (Ethernet)
```

尝试telnet连接，sniff.py抓到流量

```
load = "\xff\xfa'\x00\xff\xf0"
```

```
###[ Ethernet ]###
  dst      = 00:0c:29:6e:0e:ec
  src      = e8:6f:38:1a:09:67
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 50
  id       = 12685
  flags    = DF
  frag     = 0
  ttl      = 128
  proto    = tcp
  chksum   = 0x33c7
  src      = 192.168.10.2
  dst      = 192.168.10.31
  \options \
###[ TCP ]###
  sport    = 65058
  dport    = telnet
  seq      = 3015402091
```

Telnet 192.168.10.31

Ubuntu 20.04.1 LTS  
VM login:

```
1  ###[ Ethernet ]###
2  dst      = 00:0c:29:6e:0e:ec
3  src      = e8:6f:38:1a:09:67
4  type     = IPv4
5  ###[ IP ]###
6  version  = 4
7  ihl      = 5
8  tos      = 0x0
9  len      = 50
10 id       = 12685
11 flags    = DF
12 frag     = 0
13 ttl      = 128
14 proto    = tcp
15 chksum   = 0x33c7
```


```

16     src      = 192.168.10.2
17     dst      = 192.168.10.31
18     \options  \
19     ###[ TCP ]###
20         sport    = 65058
21         dport    = telnet
22         seq      = 3015402091
23         ack      = 887869522
24         dataofs  = 5
25         reserved = 0
26         flags    = PA
27         window   = 4106
28         chksum   = 0x4424
29         urgptr   = 0
30         options  = []
31     ###[ Raw ]###
32         load      = '\xff\xfa\x18\x00ANSI\xff\xf0'
33

```

- Capture packets comes from or to go to a particular subnet. You can pick any subnet, such as 128.230.0.0/16; you should not pick the subnet that your VM is attached to.

当抓取SEED所在子网时，能够抓取到包，filter为"net 192.168.10"，



```

seed@VM: ~/.../Labsetup

^C[07/07/21]seed@VM:~/.../Labsetup$ gedit sniffer.py
[07/07/21]seed@VM:~/.../Labsetup$ sudo ./sniffer.py
###[ Ethernet ]###
    dst      = ff:ff:ff:ff:ff:ff
    src      = 08:31:a4:6a:3f:ad
    type     = ARP
###[ ARP ]###
    hwtype   = 0x1
    ptype    = IPv4
    hwlen    = 6
    plen     = 4
    op       = who-has
    hwsrc    = 08:31:a4:6a:3f:ad
    psrc     = 192.168.10.1
    hwdst    = 00:00:00:00:00:00
    pdst     = 192.168.10.28
###[ Padding ]###
    load     = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'

###[ Ethernet ]###

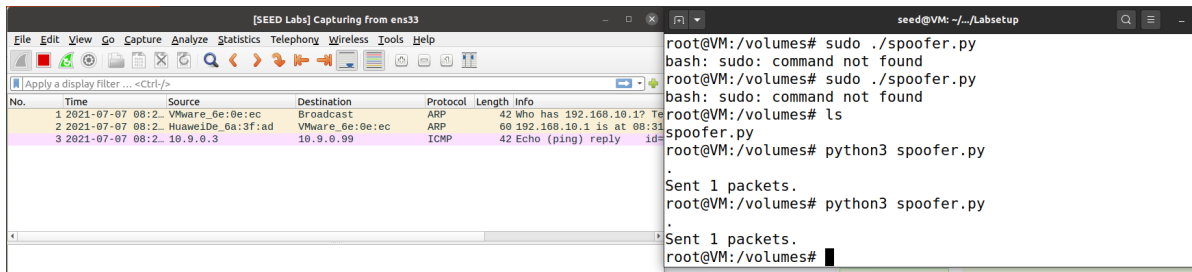
```

## Task 1.2: Spoofing ICMP Packets

```

1  #!/usr/bin/python3
2
3  from scapy.all import *
4
5  a = IP()
6  a.src = '10.9.0.3'
7  a.dst = '10.9.0.99'
8  b = ICMP(type=0)
9  p = a/b
10 send(p)
11

```



## Task 1.3: Traceroute

```
1 #!/usr/bin/python3
2 from scapy.all import *
3
4 ttl = 1
5 while True:
6     a = IP()
7     a.dst = '220.181.38.148'
8     a.ttl = ttl
9     b = ICMP()
10    send(a/b)
11    ttl += 1
```

这里选取的是baidu.com的IP地址

```
正在 Ping baidu.com [220.181.38.148] 具有 32 字节的数据:
来自 220.181.38.148 的回复: 字节=32 时间=28ms TTL=49
来自 220.181.38.148 的回复: 字节=32 时间=27ms TTL=49
来自 220.181.38.148 的回复: 字节=32 时间=27ms TTL=49
来自 220.181.38.148 的回复: 字节=32 时间=27ms TTL=49
```

```
220.181.38.148 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 27ms, 最长 = 28ms, 平均 = 27ms
```

No.	Time	Source	Destination	Protocol	Length	Info
12	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	76	Time-to-live exceeded (Time to live exceeded in transit)
13	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response f...
14	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response f...
15	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response f...
16	2021-07-07 08:4	222.190.59.173	192.168.10.31	ICMP	76	Time-to-live exceeded (Time to live exceeded in transit)
17	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response f...
18	2021-07-07 08:4	202.97.30.197	192.168.10.31	ICMP	76	Time-to-live exceeded (Time to live exceeded in transit)
19	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response ...
20	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response ...
21	2021-07-07 08:4	36.110.249.66	192.168.10.31	ICMP	76	Time-to-live exceeded (Time to live exceeded in transit)
22	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=12 (no response ...
23	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=13 (no response ...
24	2021-07-07 08:4	220.181.17.34	192.168.10.31	ICMP	76	Time-to-live exceeded (Time to live exceeded in transit)
25	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=14 (no response ...
26	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=15 (no response ...
27	2021-07-07 08:4	192.168.10.31	220.181.38.148	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=16 (no response ...

Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface ens33, id 0  
Ethernet II, Src: VMware\_0e:0e:ec (00:0c:29:0e:0e:ec), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
Address Resolution Protocol (request)

在wireshark中查看得到虚拟机到目标IP地址的路由跳数为13

## Task 1.4: Sniffing and-then Spoofing

通过捕获ICMP报文，并将其源宿地址对调，并设置ICMP类型为Reply，再发出后，就可以伪造ICMP的reply。

```

1  from scapy.all import *
2
3  def spoof_pkt(pkt):
4      if ICMP in pkt and pkt[ICMP].type == 8:
5          ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
6          icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
7          data = pkt[Raw].load
8          newpkt = ip/icmp/data
9          send(newpkt)
10
11  pkt = sniff(filter='icmp', prn=spoof_pkt)
12

```

启动程序之前执行ping

```

root@VM:/volumes# [07/07/21]seed@VM:~/.../Labsetup$ cd
[07/07/21]seed@VM:~$ ping 1.2.3.4
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
^C
--- 1.2.3.4 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7164ms

[07/07/21]seed@VM:~$ ping 10.9.0.99
PING 10.9.0.99 (10.9.0.99) 56(84) bytes of data.
^C
--- 10.9.0.99 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8180ms

[07/07/21]seed@VM:~$ 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=112 time=37.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=37.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 time=38.0 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=112 time=37.6 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 37.577/37.722/38.039/0.185 ms

```

启动程序之后:

```

[07/07/21]seed@VM:~$ 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=23.3 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=21.4 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=17.1 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=28.7 ms
■

```



```
.
root@362310dfaef7:/# 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
From 10.9.0.5 icmp_seq=6 Destination Host Unreachable
From 10.9.0.5 icmp_seq=7 Destination Host Unreachable
From 10.9.0.5 icmp_seq=8 Destination Host Unreachable
From 10.9.0.5 icmp_seq=9 Destination Host Unreachable
■

[07/07/21]seed@VM:~$ 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=15.8 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=112 time=59.5 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=16.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=146 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=26.3 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=26.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=27.6 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=28.7 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=20.5 ms
```

观察到程序在发ICMP报文:

```
[07/07/21]seed@VM:~/.../volumes$ sudo python3 sniff_spoof_icmp.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

对于1.2.3.4, 在互联网中不存在, 该程序能够欺骗容器10.9.0.5, 伪造返回的报文;

对于10.9.0.99, 该程序无法完成欺骗, 10.9.0.5不会通过网关来寻址, 而是使用ARP广播

对于Internet上真实存在的主机8.8.8.8, 该程序能够产生作用, 同时接受到真正的返回报文和虚假的返回报文, 且两者时延具有较大的区别。