Chapter1 实验报告

57118238 刘欣宇

Task 1.1A：

sniffer.py 代码如图 1-1 所示，使用 chmod 命令设置其可执行，以 root 权限运行 sniffer.py，可以捕获数据包，结果如图 1-2。在 seed 用户下运行程序，结果如图 1-3 所示，在调用 socket 时被拒绝，程序无法运行。

```
1#!/usr/bin/env python3
2from scapy.all import *
3
4def print_pkt(pkt):
5        pkt.show()
6
7
8pkt = sniff(iface='br-9088c209d191', filter='icmp',
 prn=print_pkt)
9
```

图 1-1 代码

```
root@VM:/home/seed/Desktop# ./sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:b5:2d:b2:98
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 34018
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xa1af
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x4ae6
        id        = 0x4
        seq       = 0x1
###[ Raw ]###
           load      = '\x87\xb4\xe2`\x00\x00\x00\x00w,\r\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x1
f !"#$%&\'()*+,-./01234567'
```

图 1-2 root 模式下

```
^Croot@VM:/home/seed/Desktop# su seed
[07/05/21]seed@VM:~/Desktop$ ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 8, in <module>
    pkt = sniff(iface='br-9088c209d191', 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
```

图 1-3 seed 模式下

Task 1.1B

(1) 仅捕获 ICMP 数据包

过滤规则同 Task1.1 sniffer.py，具体实现效果也同上图。

(2) 捕获来自特定 IP 且目标端口号为 23 的任何 TCP 数据包。

设特定 ip 为 10.9.0.1，代码如图 1-4：

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

def print_pkt(pkt):
        pkt.show()


pkt = sniff(iface='br-71bbea4a146f', filter='tcp and src host 10.9.0.1 and dst port 23', prn=print_pkt)

```

图 1-4 过滤规则

由于 23 端口为 telnet 协议使用的端口，使用命令：

```
[07/05/21]seed@VM:~/Desktop$ telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection refused
[07/05/21]seed@VM:~/Desktop$
```

图 1-5 telnet 命令

同时观察运行的监听程序：

```
root@VM:/home/seed/Desktop# ./1_1B2.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:7d:9a:6a:c4
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 45299
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x75a1
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ TCP ]###
        sport     = 38856
        dport     = telnet
        seq       = 1544713842
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0x1446
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (1963098344, 0)), ('NOP', None), ('WScale', 7)]
```

图 1-6 此时捕获到的 tcp 报文

可以看到报文地址与端口满足要求。

(3) 捕获来自或前往特定子网的数据包。

设特定网段为 10.203.0.0/16，代码如图 1-7：

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

def print_pkt(pkt):
        pkt.show()


#10.203.0.0/16
pkt = sniff( filter='net 10.203.0.0/16', prn=print_pkt)

```

图 1-7 过滤代码

分别对处在此网段和未处在此网段的 IP 进行 ping，可以捕获到设定网段内的报文：

```
[07/05/21]seed@VM:~/Desktop$ ping 10.203.0.207
PING 10.203.0.207 (10.203.0.207) 56(84) bytes of data.
^C
--- 10.203.0.207 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3077ms

[07/05/21]seed@VM:~/Desktop$ ping 114.114.114.114
PING 114.114.114.114 (114.114.114.114) 56(84) bytes of data.
64 bytes from 114.114.114.114: icmp_seq=1 ttl=128 time=6.52 ms
64 bytes from 114.114.114.114: icmp_seq=2 ttl=128 time=4.75 ms
64 bytes from 114.114.114.114: icmp_seq=3 ttl=128 time=23.8 ms
^C
--- 114.114.114.114 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 4.754/11.686/23.790/8.588 ms
[07/05/21]seed@VM:~/Desktop$
```

图 1-8 命令

```
###[ Ethernet ]###
  dst       = 00:50:56:e3:41:fd
  src       = 00:0c:29:8d:01:01
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 33281
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x93e5
     src       = 192.168.88.128
     dst       = 10.203.0.207
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x5fe1
        id        = 0x5
        seq       = 0x4
###[ Raw ]###
           load      = '\xf6g\xe3`\x00\x00\x00\x00\xf3y\x0c\x00\x00\x00\x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&'
```

图 1-9 捕获到的报文

Task 1.2 构造并发送欺骗报文

```
>>> from scapy.all import *
>>> a=IP()
>>> a.dst='10.9.0.5'
>>> ls(a)
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                     = []              ([])
>>>
```

图 1-10 构造 IP 报文所包含的字段并显示

```
>>> b=ICMP()
>>> p=a/b
>>> ls(p)
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                     = 1               (0)
chksum     : XShortField                       = None            (None)
src        : SourceIPField                     = '10.9.0.1'      (None)
dst        : DestIPField                       = '10.9.0.5'      (None)
options    : PacketListField                   = []              ([])
--
type       : ByteEnumField                     = 8               (8)
code       : MultiEnumField (Depends on type)  = 0               (0)
chksum     : XShortField                       = None            (None)
id         : XShortField (Cond)                = 0               (0)
seq        : XShortField (Cond)                = 0               (0)
ts_ori     : ICMPTimeStampField (Cond)         = 28336071        (28336071)
ts_rx      : ICMPTimeStampField (Cond)         = 28336071        (28336071)
ts_tx      : ICMPTimeStampField (Cond)         = 28336071        (28336071)
gw         : IPField (Cond)                    = '0.0.0.0'       ('0.0.0.0')
ptr        : ByteField (Cond)                  = 0               (0)
reserved   : ByteField (Cond)                  = 0               (0)
length     : ByteField (Cond)                  = 0               (0)
addr_mask  : IPField (Cond)                    = '0.0.0.0'       ('0.0.0.0')
nexthopmtu : ShortField (Cond)                 = 0               (0)
```

图 1-10 构造 ICMP 报文所包含的字段并显示

```
>>> send(p)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/local/lib/python3.8/dist-packages/scapy/sendrecv.py", line 345, in send
    socket = socket or conf.L3socket(*args, **kargs)
  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
```

图 1-11 在 seed 用户下运行时不允许发送

```
root@VM:/home/seed/Desktop# python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> a=IP()
>>> a.dst='10.9.0.5'
>>> b=ICMP()
>>> p=a/b
>>> send(p)
.
Sent 1 packets.
>>>
```

图 1-12 在 root 用户下运行时发送成功

同时监听，能够捕获发送的报文：

```
root@VM:/home/seed/Desktop# ./sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:b5:2d:b2:98
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x66c9
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xf7ff
        id        = 0x0
        seq       = 0x0

###[ Ethernet ]###
  dst       = 02:42:b5:2d:b2:98
  src       = 02:42:0a:09:00:05
  type      = IPv4
```

图 1-13 所捕获的报文

Task 1.3: 利用 ICMP 原理进行 Traceroute

在 Linux 环境下实验时出现的问题：只有网关的 ICMP 数据包能返回，查阅资料发现 NAT 模式下 Linux 下用高端口的 UDP 协议，发回 Host 的数据包中，vmnat 将整个 IP 头替换，没有解析到虚拟机。

```
root@VM:/home/seed/Desktop# ./1_3.py
ttl: 1 192.168.16.2
^Z
[3]+  Stopped                 ./1_3.py
```

图 1-14 只能在 ttl 为 1 时接收到网关的响应包

```
root@VM:/home/seed/Desktop# traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  _gateway (192.168.16.2)  0.398 ms  0.246 ms  0.600 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
```

图 1-15 直接使用 traceroute 尝试的结果相同

由于尝试过桥接或其他方式，也没能解决该问题，将相同代码运行于 Windows 主机，能够达到预期的结果，得到如图结果：

```
from scapy.all import *

i = 1
a = IP()
a.dst = '114.114.114.114'
b = ICMP()
while i < 255:
    a.ttl = i
    reply = sr1(a / b, verbose=0, timeout=2)
    if reply is None:
        i += 1
        continue
    print("ttl:", i, reply.src)
    if reply.src == a.dst:
        print("to the dst!")
        break
    i += 1

print("out of range!")
```

```
seed
"D:\Py program\2020srtp\Scripts\python.exe" "D:/Py program/2020srtp/seed.py"
ttl: 1 10.203.128.1
ttl: 2 10.255.254.1
ttl: 3 10.80.3.10
ttl: 4 180.101.230.145
ttl: 5 222.190.40.65
out of range!
```

图 1-16 代码及运行结果

Task 1.4：监听 ICMP 报文并返回欺骗的响应包。

首先监听所有本机 ICMP 的请求，ping 命令发出的请求为 ICMP 请求类型，type 为 8，针对符合类型的请求进行响应构造。此时欺骗报文的宿地址为捕获报文的源地址，源地址为捕获报文的宿地址，将 ICMP 报文类型设置为响应报文，其他数据段也进行与捕获报文相同的赋值。发送欺骗报文并显示提示信息。代码如图 1-17：

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


def spoofing(pkt):
        if pkt[ICMP].type != 8:
                return
        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
        send(ip/icmp/data,verbose=0)
        print("send spoofed packet from %s to %s!" %(pkt[IP].dst,pkt[IP].src))


while(1):
        pkt=sniff(filter='icmp',prn=spoofing)
```

图 1-17 代码

运行代码，并分别 ping 三个 IP 地址并得到结果：

```
[07/05/21]seed@VM:~/Desktop$ 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=23.2 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=20.0 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=26.9 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=23.3 ms
^C
--- 1.2.3.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 20.042/23.351/26.864/2.414 ms
[07/05/21]seed@VM:~/Desktop$ 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=24.2 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=52.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=18.0 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=54.3 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=17.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=55.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=16.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=128 time=58.1 ms (DUP!)
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, +4 duplicates, 0% packet loss, time 3008ms
rtt min/avg/max/mdev = 16.690/37.080/58.058/18.174 ms
[07/05/21]seed@VM:~/Desktop$ 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
^C
```

图 1-18 ping 命令及结果

```
root@VM:/home/seed/Desktop# ./try1_4.py
send spoofed packet from 1.2.3.4 to 192.168.88.128!
send spoofed packet from 1.2.3.4 to 192.168.88.128!
send spoofed packet from 1.2.3.4 to 192.168.88.128!
send spoofed packet from 1.2.3.4 to 192.168.88.128!
send spoofed packet from 8.8.8.8 to 192.168.88.128!
send spoofed packet from 8.8.8.8 to 192.168.88.128!
send spoofed packet from 8.8.8.8 to 192.168.88.128!
send spoofed packet from 8.8.8.8 to 192.168.88.128!
```

图 1-19 程序运行结果

观察到 ping 一个互联网上不存在的 IP 地址（1.2.3.4）时，能够收到欺骗报文而没有“DUP!”。而 ping 一个互联网上存在的 IP 地址（8.8.8.8）会出现“DUP!”提示，这是因为 ping 8.8.8.8 时，除了收到构造的 ICMP 响应，还会收到真实存在的来自 8.8.8.8 的响应，从而出现重复提示。而在 ping 本网段内一个不存在的

主机时，先发送 ARP 广播报文进行查找，但是对于不存在的主机无法进行响应，MAC 无法正确解析，此时 ping 直接会提示目标不可达。

其他：由于在实验过程中网络环境、docker 环境变化，所以代码中 iface 对应的网卡名在变化。