Lab 1

57118104 郭雅琪

1.1

1.1A

代码

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

def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-6b43a852a972',filter='icmp',prn=print_pkt)
```

在root用户下的运行结果：

```
root@VM:/home/seed# mycode.py
###[ Ethernet ]###
  dst       = 02:42:df:f0:f6:56
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 26643
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xb587
     src       = 10.9.0.5
     dst       = 10.0.9.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xe888
        id        = 0x25e
        seq       = 0x1
###[ Raw ]###
           load      = '\xacr\xe2`\x00\x0
x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x
01234567'

###[ Ethernet ]###
  dst       = 02:42:df:f0:f6:56
  src       = 02:42:0a:09:00:05
```

在root用户权限下，成功输出捕获的包信息。

没有root权限的情况下，因权限不够程序无法正常运行。

```
seed@VM:~$ mycode.py
Traceback (most recent call last):
  File "./mycode.py", line 7, in <module>
    pkt=sniff(iface='br-6b43a852a972',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
```

1.1B

只监听ICMP报文

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

def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-1d910c4cc163',filter='icmp',prn=print_pkt)
```

在host端分别构造ICMP报文和TCP报文发送给attacker。

```
>>> send(IP(dst="10.0.9.1")/ICMP())
.
Sent 1 packets.
>>> send(IP(dst="10.0.9.1")/TCP())
.
Sent 1 packets.
```

attacker端只收到了ICMP报文。

```
root@VM:/home/seed# mycode.py
###[ Ethernet ]###
  dst       = 02:42:28:31:3c:c9
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x5dd2
     src       = 10.9.0.5
     dst       = 10.0.9.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xf7ff
        id        = 0x0
        seq       = 0x0
```

捕获来自特定IP(1.1.1.1)且目的端口号为23的任何TCP数据包。

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

def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-1d910c4cc163',filter='tcp and src host 1.1.1.1
and dst port 23',prn=print_pkt)
```

在host端构造3个TCP数据包发送给attacker。

```
>>> send(IP(src="1.1.1.1",dst="10.0.9.1")/TCP(dport=23))
.
Sent 1 packets.
>>> send(IP(src="2.2.2.2",dst="10.0.9.1")/TCP(dport=23))
.
Sent 1 packets.
>>> send(IP(src="1.1.1.1",dst="10.0.9.1")/TCP(dport=22))
.
Sent 1 packets.
```

attacker只收到了第一个数据包。

```
root@VM:/home/seed# mycode.py
###[ Ethernet ]###
  dst       = 02:42:28:31:3c:c9
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x65cd
     src       = 1.1.1.1
     dst       = 10.0.9.1
     \options   \
###[ TCP ]###
        sport     = ftp_data
        dport     = telnet
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0x7ab5
        urgptr    = 0
        options   = []
```

捕获来自特定子网(1.1.1.0/24)的数据包。

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

def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-1d910c4cc163',filter='src net
1.1.1.0/24',prn=print_pkt)
```

在host端构造2个TCP数据包发送给attacker。

```
>>> send(IP(src="1.1.1.1",dst="10.0.9.1")/TCP())
.
Sent 1 packets.
>>> send(IP(src="1.1.2.1",dst="10.0.9.1")/TCP())
.
Sent 1 packets.
```

attacker只收到了第一个数据包。

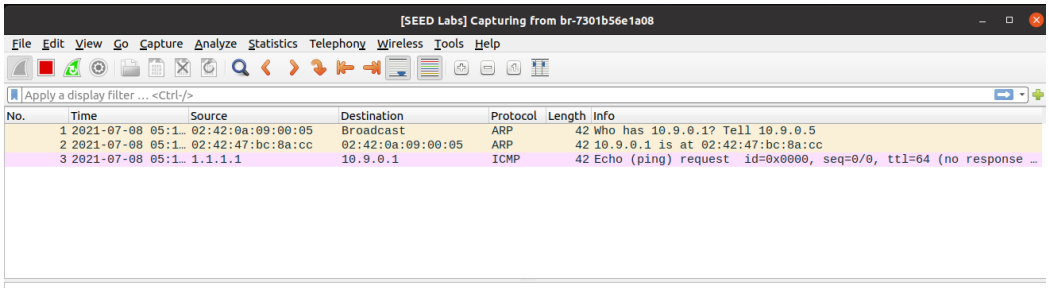
```
root@VM:/home/seed# mycode.py
###[ Ethernet ]###
  dst       = 02:42:28:31:3c:c9
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x65cd
     src       = 1.1.1.1
     dst       = 10.0.9.1
     \options   \
###[ TCP ]###
        sport     = ftp_data
        dport     = http
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0x7a7c
        urgptr    = 0
        options   = []
```

1.2

构造任意的ICMP报文，将宿地址设置为10.9.0.1，源地址设置成1.1.1.1。

```
>>> send(IP(src="1.1.1.1",dst="10.9.0.1")/ICMP())
.
Sent 1 packets.
```

通过wireshark软件，捕获到发送的ICMP报文。



1.3

根据手册中给出的traceroute原理编写代码。主机不断构造发往目的主机的ICMP报文，其中的ttl值从1开始不断增大，直到收到的ICMP应答报文的源地址和目的主机相同为止。

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

MAX_TTL=255
tempIP=input("Please input the IP address or domain name:")
dstIP=socket.gethostbyname(tempIP)

ip=IP(dst=dstIP,ttl=1)
icmp=ICMP()

while ip.ttl<=MAX_TTL:
    reply=sr1(ip/icmp,verbose=0,timeout=2)
    if(reply==None):
        print(str(ip.ttl)+"\t ***")
        ip.ttl+=1
        continue
    print(str(ip.ttl)+"\t"+reply.src)

    if(reply.src==dstIP):
        break
    ip.ttl+=1
```

输入8.8.8.8进行测试，结果如下：

```
root@VM:/home/seed# tracer.py
Please input the IP address or domain name:8.8.8.8
1       192.168.43.51
2        ***
3        ***
4        ***
5        ***
6       122.193.15.13
7        ***
8       58.240.53.166
9       58.240.48.98
10      8.8.8.8
```

输入www.baidu.com进行测试，结果如下：

```
root@VM:/home/seed# tracer.py
Please input the IP address or domain name:www.baidu.com
1       192.168.43.51
2        ***
3        ***
4        ***
5        ***
6       122.193.15.17
7       221.6.5.21
8       122.96.66.102
9       58.240.96.34
10      182.61.216.0
11       ***
12      112.80.248.76
```

部分中间节点没有显示，可能是因为对应主机被禁止回应ICMP报文。

1.4

ICMP请求应答报文的type值为8，因此，遇到type值不是8的报文可以直接返回。对于ICMP请求应答报文，将其源地址和宿地址交换；type值修改为0，其余部分不变即可构造ICMP应答报文。

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

def spoof(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)
    load=pkt[Raw].load
    newpkt=ip/icmp/load

    send(newpkt)
    print("spoof\n")

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

ping 1.2.3.4

```
[07/08/21]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=14.5 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=14.5 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=16.5 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=19.1 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=11.7 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=17.2 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=19.5 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=22.1 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=20.2 ms
```

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

.
Sent 1 packets.
spoof

.
Sent 1 packets.
spoof

.
Sent 1 packets.
spoof

.
Sent 1 packets.
spoof

^C.
Sent 1 packets.
spoof
```

此时伪造程序伪造了报文并发送给attacker, 令attacker误认为主机存在。

ping 10.9.0.99

```
[07/08/21]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
```

```
root@VM:/home/seed# spoof.py
```

伪造程序没有构造报文发送，attacker可以发现主机并不存在。

ping 8.8.8.8

```
[07/08/21]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=21.4 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=112 time=40.0 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=15.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=112 time=39.8 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=23.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=112 time=40.7 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=12.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=112 time=40.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=5 ttl=64 time=18.7 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=112 time=40.0 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=6 ttl=64 time=16.7 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=112 time=40.1 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=7 ttl=64 time=19.6 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=112 time=44.9 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=8 ttl=64 time=21.5 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=112 time=41.9 ms (DUP!)
^C
```

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

.
Sent 1 packets.
spoof

.
Sent 1 packets.
spoof

.
Sent 1 packets.
spoof

.
Sent 1 packets.
spoof

.
```

伪造程序伪造了响应报文。

发往外网的报文会经过attacker(网关)，因此伪造程序可以观察到报文并回应。而发送到内网的报文则不会，所以不会被伪造。