Packet Sniffing and Spoofing Lab

57118224 邱龙

Task 1.1A: Sniffing Packets

sniffing.py 代码为：

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-91d0ed79147a', filter='icmp', prn=print  pkt)
```

（1）使用 root 权限运行 sniffing.py 时显示：

```
###[ Ethernet ]###
   dst            = 02:42:8e:4c:9f:e5
   src            = 02:42:0a:09:00:05
   type           = IPv4
###[ IP ]###
        version     = 4
        ihl         = 5
        tos         = 0x0
        len         = 84
        id          = 32604
        flags       = DF
        frag        = 0
        ttl         = 64
        proto       = icmp
        chksum      = 0xa735
        src         = 10.9.0.5
        dst         = 10.9.0.1
        \options    \
###[ ICMP ]###
            type        = echo-request
            code        = 0
            chksum      = 0x7e34
            id          = 0xe
            seq         = 0x6
###[ Raw ]###
                load                                                                    =
'\xea\xe9\xe2`\x00\x00\x00\x00\xe0\x99\r\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 用户执行时报错：

```
seed@VM:/volumes$ sniffing.py
Traceback (most recent call last):
  File "./sniffing.py", line 5, in <module>
    pkt = sniff(iface='br-91d0ed79147a', 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
```

可以发现存在权限错误。

Task 1.1B: Sniffing Packets

1. 仅捕获 ICMP 报文

与 Task1.1A 代码一致，filter='icmp'，输出结果相同。

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-91d0ed79147a', filter='icmp', prn=print  pkt)
```

2. 捕获从特定 IP 发出的，目的端口为 23 的 TCP 包

ifconfig 查看 host 地址为 10.9.0.5，故设 filter='src host 10.9.0.5 and tcp dst port 23'

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
    pkt.show()
pkt = sniff(iface='br-91d0ed79147a', filter=filter='src host 10.9.0.5 and tcp dst port 23',
prn=print_pkt)
```

在 host 主机上 Telnet attacker 的 IP 地址，运行结果如下：

```
###[ Ethernet ]###
   dst       = 02:42:8e:4c:9f:e5
   src       = 02:42:0a:09:00:05
   type      = IPv4
###[ IP ]###
      version   = 4
      ihl       = 5
      tos       = 0x10
      len       = 60
      id        = 51538
      flags     = DF
      frag      = 0
      ttl       = 64
      proto     = tcp
      chksum    = 0x5d42
      src       = 10.9.0.5
      dst       = 10.9.0.1
      \options   \
###[ TCP ]###
         sport     = 39804
         dport     = telnet
         seq       = 3291973419
         ack       = 0
         dataofs   = 10
         reserved  = 0
         flags     = S
         window    = 64240
         chksum    = 0x1446
         urgptr    = 0
         options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp', (3949259327, 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

设置 filter='net 128.230.0.0/16'。

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
   pkt.show()
pkt = sniff(iface='br-91d0ed79147a', filter='net 128.230.0.0/16', prn=print_pkt)
```

这里由于需要捕获来自或者去往特定子网的数据包，所以我们构造一个源 ip 为 128.230.0.0/16 子网中 128.230.0.1、目的地址为 attacker 的 IP 地址的数据包，并发送后成功捕获。发送数据包代码如下：

```
from scapy.all import *
def Pkt_send():
        pkt = IP(src="128.230.0.1",dst="10.9.0.1")/ICMP()
        send(pkt)
if __name__=="__main__":
        Pkt  send()
```

sniffing.py 输出结果：

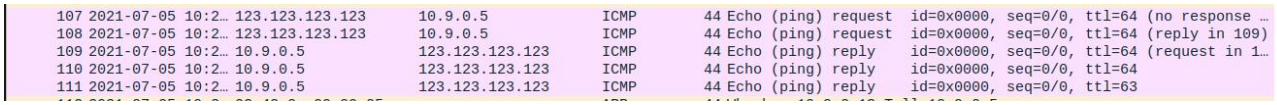
```
###[ Ethernet ]###
   dst            = 02:42:8e:4c:9f:e5
   src            = 02:42:0a:09:00:05
   type           = IPv4
###[ IP ]###
        version     = 4
        ihl         = 5
        tos         = 0x0
        len         = 28
        id          = 1
        flags       =
        frag        = 0
        ttl         = 64
        proto       = icmp
        chksum      = 0xefef
        src         = 128.230.0.1
        dst         = 10.9.0.1
        \options    \
###[ ICMP ]###
            type        = echo-request
            code        = 0
            chksum      = 0xf7ff
            id          = 0x0
            seq         = 0x0
```

Task 1.2: Spoofing ICMP Packets

设置伪装的 ip 地址为 123.123.123.123，dst 为目标地址。
运行代码：

```
from scapy.all import *
a = IP()
a.src = '123.123.123.123'
a.dst = '10.9.0.5'
b = ICMP()
p = a/b
```

Wireshark 查看：



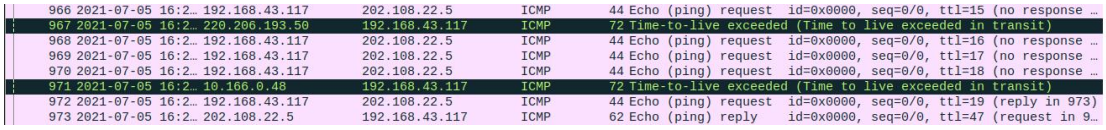
成功回显请求包。

Task 1.3: Traceroute

traceroute 代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
for i in range(1,30):
    a = IP()
    a.dst = '202.108.22.5'
    a.ttl = i
    b = ICMP()
    send(a/b)
```

使用 wireshark 查看：



TTL 为 19 时 Echo 第一个 reply，所以虚拟机和目的地址间隔约为 19 跳。

Task 1.4: Sniffing and-then Spoofing

代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
def sniffing_spoofing(pkt):
    if pkt[ICMP].type == 8: #回显请求
        ip = IP(src=pkt[IP].dst, dst=pkt[IP].src)
        icmp = ICMP(type=0,id=pkt[ICMP].id, seq=pkt[ICMP].seq)
        data = pkt[Raw].load
        packet = ip/icmp/data
        send(packet)
pkt = sniff(iface='br-91d0ed79147a',filter='icmp', prn=sniffing_spoofing)
```

1. ping 1.2.3.4　# a non-existing host on the Internet

结果如下：

```
root@f0424769232c:/# 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=4 ttl=64 time=68.1 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=27.3 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=19.6 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=19.1 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=21.9 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=22.5 ms
64 bytes from 1.2.3.4: icmp_seq=10 ttl=64 time=13.0 ms
^Z
```

```
root@VM:/volumes# task4.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
```

可以发现，原本不存在的 ip 地址原本是应该 ping 不通的，但是现在能 ping 通，说明我们成功伪造 reply 进行了数据包欺骗。

2. ping 10.9.0.99　# a non-existing host on the LAN

结果如下：

```
root@VM:/volumes# task4.py
```

```
root@f0424769232c:/# 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
```

输出结果显示目标主机不可达，且没有发送伪造 reply 数据包。

```
root@f0424769232c:/# ip route add  10.9.0.99  via 10.9.0.1 dev eth0
root@f0424769232c:/# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    U
0.0.0.0         10.9.0.1        0.0.0.0         UG    0      0
10.9.0.0        0.0.0.0         255.255.255.0   U     0      0
10.9.0.99       10.9.0.1        255.255.255.255 UGH   0      0
```

如果增加一条 10.9.0.99 指向 10.9.0.1 网关的路由发现可以 ping 通。

```
root@VM:/volumes# task4.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
root@f0424769232c:/# 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=65.6 ms
From 10.9.0.1: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=2 ttl=64 time=19.5 ms
From 10.9.0.1: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=3 ttl=64 time=18.9 ms
From 10.9.0.1: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.99)
64 bytes from 10.9.0.99: icmp_seq=4 ttl=64 time=19.5 ms
```

3. ping 8.8.8.8　# an existing host on the Internet

输出如下：

```
root@VM:/volumes# task4.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

```
root@f0424769232c:/# 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=68.5 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=127 time=78.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=25.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=127 time=83.4 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=25.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=127 time=79.8 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=27.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=127 time=93.4 ms (DUP!)
```

可以发现能 ping 通，而且出现 DUP！字样，说明受到多个 reply 报文，一个是正常 ping 通的 reply，一个是我们伪造的 reply。