Lab1

57118208 吴文婷

Task1.1 Sniffing Packets

Task1.1 A

sniffer.py

```
from scapy.all import*

def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-78e9ebf024ae',filter='icmp',prn=print_pkt)
```

启动 docker，查看网络 ID

```
[07/25/21]seed@VM:~/.../Labsetup$ dockps
547e7b97fb72  seed-attacker
bc2eb5707938  host-10.9.0.5
[07/25/21]seed@VM:~/.../Labsetup$ ifconfig | grep br
br-78e9ebf024ae: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.2
55
        inet 192.168.201.134  netmask 255.255.255.0  broadcast 1
92.168.201.255
[07/25/21]seed@VM:~/.../Labsetup$
```

以 root 权限运行 sniffer.py
打开一个命令行对主机 IP 进行 ping 命令

```
[07/25/21]seed@VM:~/Desktop$ vim sniffer.py
[07/25/21]seed@VM:~/Desktop$ chmod a+x sniffer.py
[07/25/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:a4:c9:74:4d
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 49746
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x643f
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xcc36
        id        = 0x3
        seq       = 0x1
```

```
[07/25/21]seed@VM:~/.../Labsetup$ ping 10.9.0.5
PING 10.9.0.5 (10.9.0.5) 56(84) bytes of data.
64 bytes from 10.9.0.5: icmp_seq=1 ttl=64 time=0.136 ms
64 bytes from 10.9.0.5: icmp_seq=2 ttl=64 time=0.062 ms
64 bytes from 10.9.0.5: icmp_seq=3 ttl=64 time=0.124 ms
^C
--- 10.9.0.5 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2039ms
rtt min/avg/max/mdev = 0.062/0.107/0.136/0.032 ms
```

以 seed 用户运行 sniffer.py 时，系统会报错

```
[07/25/21]seed@VM:~/Desktop$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 8, in <module>
    pkt=sniff(iface='br-78e9ebf024ae',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 __ini
t__
    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
[07/25/21]seed@VM:~/Desktop$
```

Task1.1 B

只抓取 ICMP 报文，见 Task1.1 A 所示。
捕获任何来自特定 IP 的 TCP 数据包，目的端口为 23。

```
from scapy.all import*
Wireshark
def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-78e9ebf024ae',filter='tcp port 23 and host 10.9.0.5',
        prn=print_pkt)
```

利用 docksh 获取 host 的 shell，telnet 任意一个 IP 地址建立连接。

```
[07/25/21]seed@VM:~/.../volumes$ ifconfig | grep br
br-78e9ebf024ae: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet 192.168.201.134  netmask 255.255.255.0  broadcast 192.168.201.255
[07/25/21]seed@VM:~/.../volumes$ docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
8794053f3bfa        bridge              bridge              local
b3581338a28d        host                host                local
78e9ebf024ae        net-10.9.0.0        bridge              local
77acecccbe26        none                null                local
[07/25/21]seed@VM:~/.../volumes$ dockps
547e7b97fb72  seed-attacker
bc2eb5707938  host-10.9.0.5
[07/25/21]seed@VM:~/.../volumes$ docksh bc
root@bc2eb5707938:/# telnet 1.1.1.1
Trying 1.1.1.1...
telnet: Unable to connect to remote host: Connection refused
root@bc2eb5707938:/#
```

在另一处可看到 tcp 数据包

```
[07/25/21]seed@VM:~/Desktop$ vi sniffer.py
[07/25/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:a4:c9:74:4d
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 60273
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x432b
     src       = 10.9.0.5
     dst       = 1.1.1.1
     \options   \
###[ TCP ]###
        sport     = 46824
        dport     = telnet
        seq       = 2322741697
        ack       = 0
```

捕获来自或去特定子网的数据包。可以选择任何子网，如 128.230.0.0/16；不应该 选择 VM 所绑定的子网。

```
from scapy.all import*

def print_pkt(pkt):
    pkt.show()

pkt=sniff(iface='br-78e9ebf024ae',filter='host 10.9.0.8',
        prn=print_pkt)
```

直接 Ping10.9.0.8，可捕获的数据包

```
[07/25/21]seed@VM:~/.../volumes$ docksh bc
root@bc2eb5707938:/# telnet 1.1.1.1
Trying 1.1.1.1...
telnet: Unable to connect to remote host: Connection refused
root@bc2eb5707938:/# ping 10.9.0.8
PING 10.9.0.8 (10.9.0.8) 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
```

```
[07/25/21]seed@VM:~/Desktop$ vi sniffer.py
[07/25/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 02:42:0a:09:00:05
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
     hwlen     = 6
     plen      = 4
     op        = who-has
     hwsrc     = 02:42:0a:09:00:05
     psrc      = 10.9.0.5
     hwdst     = 00:00:00:00:00:00
     pdst      = 10.9.0.8

###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 02:42:0a:09:00:05
  type      = ARP
###[ ARP ]###
     hwtype    = 0x1
     ptype     = IPv4
```

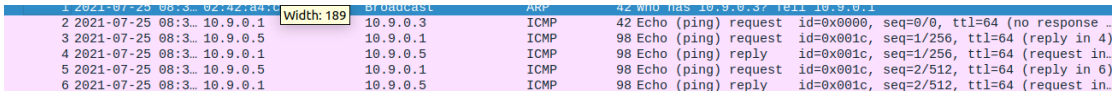
Task1.2 Spoofing ICMP Packets

```
from scapy.all import*

a=IP()
a.dst='10.9.0.3'
b=ICMP()
p=a/b
send(p)
ls(a)
```

第一行创建了一个 ICMP 对象，默认类型为 echo request。在第六行中，我们将 a 和 b 堆叠在一起形成了 一个新对象，“/”操作符被重载，不在表示除法，而是将 b 添加为 a 的有效负载字段，并相应地修改 a 的字 段。最终我们得到一个表示 ICMP 数据包的新对象，报文重组后，向子网内的一个 IP 发送数据包，打开 Wireshark 可观测发送数据包和响应数据包。

```
[07/25/21]seed@VM:~/Desktop$ sudo python3 sniffer.py
.
Sent 1 packets.
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                         = '10.9.0.3'      (None)
options    : PacketListField                     = []              ([])
```

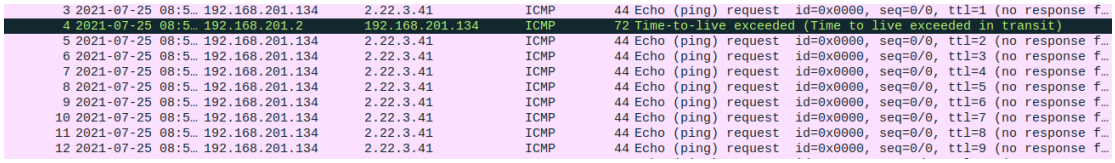


Task1.3 Traceroute

```
from scapy.all import*

a=IP()
b=ICMP()
a.dst='2.22.3.41'
for i in range(30):
    a.ttl=i+1
    p=a/b
    send(p)
```

创建一个文件 trace.py，向目标 IP 发送 ICMP 数据包，一开始设置 TTL (Time-To-Live)值为 1，那么发出 的 ICMP 数据包在经历一个路由结点后，就会失活被抛弃,我们利用循环，不断增加 TTL 的值，最终使得数 据包到达目的地。



Task1.4 Sniffing and then Spoofing

```
from scapy.all import*

def spoof_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type==8:
        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
        newpkt=ip/icmp/data
        send(newpkt)

        pkt=sniff(filter='icmp',prn=spoff_pkt)
~
```

```
root@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=128 time=347 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=361 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=383 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=128 time=382 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=128 time=350 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=128 time=346 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=128 time=381 ms
64 bytes from 8.8.8.8: icmp_seq=11 ttl=128 time=383 ms
^C
--- 8.8.8.8 ping statistics ---
12 packets transmitted, 8 received, 33.3333% packet loss, time 11050m
s
rtt min/avg/max/mdev = 345.556/366.462/383.068/16.282 ms
```

```
PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
^C
--- 1.2.3.4 ping statistics ---
22 packets transmitted, 0 received, 100% packet loss, time 21510ms
```

```
root@VM:/# ping 10.9.0.1
PING 10.9.0.1 (10.9.0.1) 56(84) bytes of data.
64 bytes from 10.9.0.1: icmp_seq=1 ttl=64 time=0.045 ms
64 bytes from 10.9.0.1: icmp_seq=2 ttl=64 time=0.067 ms
64 bytes from 10.9.0.1: icmp_seq=3 ttl=64 time=0.110 ms
64 bytes from 10.9.0.1: icmp_seq=4 ttl=64 time=0.063 ms
64 bytes from 10.9.0.1: icmp_seq=5 ttl=64 time=0.063 ms
64 bytes from 10.9.0.1: icmp_seq=6 ttl=64 time=0.143 ms
64 bytes from 10.9.0.1: icmp_seq=7 ttl=64 time=0.046 ms
64 bytes from 10.9.0.1: icmp_seq=8 ttl=64 time=0.080 ms
64 bytes from 10.9.0.1: icmp_seq=9 ttl=64 time=0.055 ms
64 bytes from 10.9.0.1: icmp_seq=10 ttl=64 time=0.105 ms
64 bytes from 10.9.0.1: icmp_seq=11 ttl=64 time=0.049 ms
64 bytes from 10.9.0.1: icmp_seq=12 ttl=64 time=0.096 ms
64 bytes from 10.9.0.1: icmp_seq=13 ttl=64 time=0.078 ms
64 bytes from 10.9.0.1: icmp_seq=14 ttl=64 time=0.042 ms
64 bytes from 10.9.0.1: icmp_seq=15 ttl=64 time=0.074 ms
64 bytes from 10.9.0.1: icmp_seq=16 ttl=64 time=0.056 ms
^C
--- 10.9.0.1 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15338ms
rtt min/avg/max/mdev = 0.042/0.073/0.143/0.027 ms
```