Packet Sniffifing and Spoofifing Lab

57118203 陈萱妍

Task 1.1: Sniffifing Packets

Task 1.1A.

启动 docker 容器：

```
[07/09/21]seed@VM:~/.../volumes$ dcup
Starting host-10.9.0.5 ... done
Starting seed-attacker ... done
Attaching to seed-attacker, host-10.9.0.5
```

查看网络接口：

```
[07/07/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ ifconfig
br-bef5d932ff01: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:e4ff:feea:1a08  prefixlen 64  scopeid 0x20<link>
        ether 02:42:e4:ea:1a:08  txqueuelen 0  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 77  bytes 9806 (9.8 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

可观察到接口为 br-bef5d932ff01

通过设置 filter 规则为'icmp'过滤其他的数据包；通过指定 prn 的参数为显示数据包的函数在终端打印数据包。在执行 sniffer.py 脚本的同时，使用 ping 发出 icmp 数据包。

sniffer.py
~/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
 pkt.show()
pkt = sniff(iface='br-bef5d932ff01', filter='icmp', prn=print_pkt)
```

当使用 sudo 执行脚本时可以正常捕获 icmp 包，如下图：

```
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ chmod a+x
sniffer.py
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ sudo pytho
n3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:e2:3e:c9:a4
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 63046
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x3853
     src       = 10.9.0.5
```

当在普通用户下运行时，显示操作不被允许。

```
[07/08/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ python3 sniffe
r.py
Traceback (most recent call last):
  File "sniffer.py", line 5, in <module>
    pkt = sniff(iface='br-bef5d932ff01', 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", lin
e 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.h
tons(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
```

Task 1.1B.

1. Capture only the ICMP packet

结果如 Task1.1A 所示

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

修改 sniffer.py

```
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
 pkt.show()
pkt = sniff(iface='br-bef5d932ff01', filter='tcp and src host 10.9.0.5 and dst port 23', prn=print_pkt)
```

登录 10.9.0.5　telnet 10.9.0.1 23

```
[07/09/21]seed@VM:~$ dockps
72954644b97e  seed-attacker
1125f103330e  host-10.9.0.5
[07/09/21]seed@VM:~$ docksh 1
root@1125f103330e:/# telnet 10.9.0.1 23
Trying 10.9.0.1...
Connected to 10.9.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
VM login: seed
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 updates can be installed immediately.
0 of these updates are security updates.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Fri Jul  9 08:59:33 EDT 2021 from www.SeedLabSQLInjection.com on pts/0
```

监听到的报文如下：

```
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:e2:3e:c9:a4
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 53
     id        = 18123
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xdfd0
     src       = 10.9.0.5
     dst       = 10.9.0.1
     \options   \
###[ TCP ]###
        sport     = 59828
        dport     = telnet
        seq       = 3404320496
        ack       = 586356785
```

3. 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.

修改 sniffer.py

```
Open   sniffer.py   ~/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab   Save
sniff_spoof.py   sniffer.py
#!/usr/bin/env python3
from scapy.all import *
def print_pkt(pkt):
 pkt.show()
pkt = sniff(iface='br-bef5d932ff01', filter='dst net 128.230.0.0/16', prn=print_pkt)
```

登录 10.9.0.5 ping 128.230.0.1

```
[07/09/21]seed@VM:~$ docksh 1
root@1125f103330e:/# ping 128.230.0.1
PING 128.230.0.1 (128.230.0.1) 56(84) bytes of data.
From 10.9.0.1 icmp_seq=1 Destination Net Unreachable
From 10.9.0.1 icmp_seq=2 Destination Net Unreachable
From 10.9.0.1 icmp_seq=3 Destination Net Unreachable
From 10.9.0.1 icmp_seq=4 Destination Net Unreachable
^C
--- 128.230.0.1 ping statistics ---
7 packets transmitted, 0 received, +4 errors, 100% packet loss, time 6121ms
```

监听到的报文如下：

```
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:e2:3e:c9:a4
  src       = 02:42:0a:09:00:05
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 1717
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xa8ff
     src       = 10.9.0.5
     dst       = 128.230.0.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x9b79
        id        = 0x27
        seq       = 0x1
```

Task 1.2: Spoofifing ICMP Packets

编写脚本 spoofer.py 构造报文

```
from scapy.all import *

send(IP(dst='10.9.0.5')/ICMP())
```

执行程序发送数据包

```
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ chmod a+x spoofer.py
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ sudo python3 spoofer.py
.
Sent 1 packets.
```

使用 wireshark 捕获往返数据包

3	2021-07-09 09:4…	10.9.0.1	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, tt
4	2021-07-09 09:4…	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, tt

Task 1.3: Traceroute

编写脚本 tarceroute.py 发送 syn 包并接收返回包

```
from scapy.all import *

ans,unans=sr(IP(dst='www.baidu.com', ttl=(4,25))/-
 TCP(flags=0x2))
for snd,rcv in ans:
 print(snd.ttl, rcv.src, isinstance(rcv.payload, TCP))
```

运行程序结果如下

```
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ sud
o python3 traceroute.py
Begin emission:
Finished sending 22 packets.
.**********..................^C
Received 30 packets, got 10 answers, remaining 12 packets
4 183.207.22.145 False
5 10.203.195.6 False
6 183.207.55.114 False
7 36.152.44.96 True
8 182.61.216.72 False
9 36.152.44.96 True
10 36.152.44.96 True
11 36.152.44.96 True
12 36.152.44.96 True
13 36.152.44.96 True
```

Task 1.4: Sniffifing and-then Spoofifing

编写脚本 sniff_spoof.py 截获本网段的数据包并发出回应包

sniff_spoof.py
~/Desktop/Labs_20.04/Network Security/Packet Sniffing and Spoofing Lab

```python
from scapy.all import *

def print_pkt(pkt):
    if ICMP in pkt and pkt[ICMP].type==8:
            print('origin packet ...')
            print('src ip:',pkt[IP].src)
            print('dst ip:',pkt[IP].dst)

            a=IP()
            a.src=pkt[IP].dst
            a.dst=pkt[IP].src
            a.ihl=pkt[IP].ihl
            b=ICMP()
            b.type=0
            b.id=pkt[ICMP].id
            b.seq=pkt[ICMP].seq
            data=pkt[Raw].load
            send(a/b/data)

pkt=sniff(iface='br-bef5d932ff01',filter='icmp',prn=print_pkt)
```

在运行程序前，分别 ping 三个地址

ping 1.2.3.4 # a non-existing host on the Internet

```
[07/09/21]seed@VM:~$ dockps
72954644b97e  seed-attacker
1125f103330e  host-10.9.0.5
[07/09/21]seed@VM:~$ docksh 11
root@1125f103330e:/# 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 7147ms
```

ping 10.9.0.99 # a non-existing host on the LAN

```
root@1125f103330e:/# 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
^C
--- 10.9.0.99 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6143ms
pipe 4
```

ping 8.8.8.8 # an existing host on the Internet

```
root@1125f103330e:/# 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=108 time=102 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=97.3 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=108 time=105 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=108 time=94.7 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=108 time=101 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4007ms
rtt min/avg/max/mdev = 94.738/100.015/105.085/3.624 ms
```

1.2.3.4 路由寻址

```
root@1125f103330e:/# ip route get 1.2.3.4
1.2.3.4 via 10.9.0.1 dev eth0 src 10.9.0.5 uid 0
    cache
```

可以观察到，无法 ping 通 1.2.3.4 和 10.9.0.99，可以 ping 通 8.8.8.8

运行 sniff_spoof.py，再次 ping 三个地址

```
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ chmod a+x sniff_spoof.py
[07/09/21]seed@VM:~/.../Packet Sniffing and Spoofing Lab$ sudo python3 sniff_spoof.py
origin packet ...
src ip: 10.9.0.5
dst ip: 1.2.3.4
.
Sent 1 packets.
origin packet ...
src ip: 10.9.0.5
dst ip: 1.2.3.4
.
Sent 1 packets.
origin packet ...
src ip: 10.9.0.5
dst ip: 1.2.3.4
.
Sent 1 packets.
origin packet ...
```

ping 1.2.3.4 # a non-existing host on the Internet

```
root@1125f103330e:/# 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=76.8 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=33.0 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=26.3 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=34.8 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=24.4 ms
^C
--- 1.2.3.4 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4008ms
rtt min/avg/max/mdev = 24.406/39.065/76.837/19.288 ms
```

ping 10.9.0.99 # a non-existing host on the LAN

```
root@1125f103330e:/# 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
^C
--- 10.9.0.99 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6149ms
pipe 4
```

ping 8.8.8.8 # an existing host on the Internet

```
root@1125f103330e:/# 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=21.8 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=108 time=120 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=30.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=108 time=375 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=36.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=108 time=653 ms (DUP!)
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=22.1 ms
^C
--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, +3 duplicates, 0% packet loss, time 3003ms
rtt min/avg/max/mdev = 21.832/179.867/653.103/226.195 ms
```

可以观察到，运行 sniff_spoof.py 后，可以 ping 通 1.2.3.4 和 8.8.8.8，但仍然无法 ping 通 10.9.0.99.

结论：在运行程序之前，网关 10.9.0.5 无法通过 ARP 协议找到 1.2.3.4 和 10.9.0.99 对应的 mac 地址，无法 ping 通。而 8.8.8.8 为互联网上存在的真实地址，因此可以 ping 通。运行程序之后，ping1.2.3.4 需要经过网关 10.9.0.5 向外转发，网关拦截 ICMP 报文并欺骗主机可以 ping 通 1.2.3.4，而 10.9.0.99 和主机在同一个网段内，寻找 mac 地址时不需要经过网关，网关无法拦截报文欺骗主机，所以无法 ping 通 10.9.0.99.