Packet Sniffing and Spoofing Lab

57118211 谢瑞

Lab Task Set 1： Using Tools to Sniff and Spoof Packets

Task 1.1： Sniffing Packets

Task 1.1A

使用命令 ifconfig 查看广播地址如下：

```
[07/05/21]seed@VM:~/.../Labsetup$ dockps
a4c0db4f6b0c  host-10.9.0.5
9b8ce36b0233  seed-attacker
[07/05/21]seed@VM:~/.../Labsetup$ docksh 9b
root@VM:/# ifconfig
br-b5d69ac66bd5: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1
500
        inet 10.9.0.1  netmask 255.255.255.0  broadcast 10.9.0.255
        inet6 fe80::42:68ff:feef:13b7  prefixlen 64  scopeid 0x20<
link>
        ether 02:42:68:ef:13:b7  txqueuelen 0  (Ethernet)
```

sniffer.py 测试程序代码如下：

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

```python
from scapy.all import *

def print_pkt(pkt):
        pkt.show()

pkt = sniff(iface='br-b5d69ac66bd5', filter='icmp', prn=print_pkt)
```

root 权限下运行结果如下：

seed@VM: ~/.../volumes

```
[07/05/21]seed@VM:~/.../volumes$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:68:ef:13:b7
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 63741
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x2d94
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
```

普通 user 权限运行结果如下，程序报错：

```
[07/05/21]seed@VM:~/.../volumes$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 7, in <module>
    pkt = sniff(iface='br-b5d69ac66bd5', 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, so
cket.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
```

通过对 Trace 的观察可知，报错的原因为普通用户没有权限创建 socket 。

Task 1.1B

1） 仅捕获 ICMP 报文

filter 测试程序代码如下：

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

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

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='icmp', prn=print_pkt)
```

运行结果如下：

```
[07/05/21]seed@VM:~/.../volumes$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:58:68:66:91
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 7788
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x826
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
```

2) 捕获从特定 IP 发出的，目的端口为 23 的 TCP 包

tcp_sniffer.py 代码如下：

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

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='tcp and src host 10.9.0.1 and dst port 23', prn=print_pkt)
```

tcp_send.py 代码如下：

```
from scapy.all import *

ip=IP()
ip.src='10.9.0.1'
ip.dst='10.9.0.5'
tcp=TCP()
tcp.dport=23
send(ip/tcp)
```

运行 send 后结果如下：

```
root@VM:/volumes# python3 tcp_sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:58:68:66:91
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 40
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x66b8
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
```

可见成功捕获。

3) 捕获从特定子网中发起或前往特定子网的报文

sniffer 代码如下：

subnet_sniffer.py
~/Desktop/Labs_20.04/Network Security/Pac...niffing and Spoofing Lab/Labsetup/volumes

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

def print_pkt(pkt):
        pkt.show()

pkt = sniff(filter='dst net 128.230.0.0/16', prn=print_pkt)
```

send 代码如下：

subnet_send.py
~/Desktop/Labs_20.04/Network Security/Pack... Sniffing and Spoofing Lab/Labsetup/volumes

```python
from scapy.all import *

ip=IP()
ip.dst='128.230.0.0/16'
send(ip)
```

运行 send 后结果如下：

```
[07/07/21]seed@VM:~/.../volumes$ sudo python3 subnet_sniffer.py
###[ Ethernet ]###
  dst       = 52:54:00:12:35:02
  src       = 08:00:27:26:f0:fe
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 20
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = hopopt
     chksum    = 0xedf4
     src       = 10.0.2.15
     dst       = 128.230.0.0
     \options   \
```

可见成功捕获。

Task 1.2：Spoofing ICMP Packets

spoofing 代码如下：

icmp_spoofing.py
~/Desktop/Labs_20.04/Network Security/Pac...Sniffing and Spoofing Lab/Labsetup/volumes

```python
from scapy.all import *

ip = IP()
ip.dst = '10.0.2.3'
b = ICMP()
p = ip/b
send(p)
```

将 ip 的 src 设置为想要伪装的源地址， dst 设置为目标的 IP 地址后，即可使用 Wireshark 查看。

运行 spoofing 代码前结果如下：

2021-07-05 22:3…	10.9.0.1	10.9.0.5	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 4)
2021-07-05 22:3…	10.9.0.5	10.9.0.1	ICMP	42	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 3)

运行 spoofing 代码后结果如下：

2021-07-05 22:4…	10.0.2.15	10.0.2.3	ICMP	42	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 4)
2021-07-05 22:4…	10.0.2.3	10.0.2.15	ICMP	60	Echo (ping) reply id=0x0000, seq=0/0, ttl=64 (request in 3)

可见成功伪装，可以利用此方法伪装成其他任意 ip 地址。

Task 1.3： Traceroute

使用 Scapy 来估计虚拟机与目标地址之间的路由器跳数。

测试程序 traceroute 如下：

traceroute.py
~/Desktop/Labs_20.04/Network Security/Packet Sniffing and S...

```python
from scapy.all import *

ttl = 1
while True:
        a = IP()
        a.dst = '1.2.3.4'
        a.ttl = ttl
        b = ICMP()
        send(a/b)
        ttl += 1
```

该程序通过一个无限循环，每次将 TTL 递增，然后使用 Wireshark 查看：

1	2021-07-08 01:4…	PcsCompu_26:f0:fe	Broadcast	ARP	42 Who has 10.0.2.2? Tell 10.0.2.15
2	2021-07-08 01:4…	RealtekU_12:35:02	PcsCompu_26:f0:fe	ARP	60 10.0.2.2 is at 52:54:00:12:35:02
3	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=1 (no response f…
4	2021-07-08 01:4…	10.0.2.2	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
5	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=2 (no response f…
6	2021-07-08 01:4…	172.20.10.1	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
7	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=3 (no response f…
8	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=4 (no response f…
9	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=5 (no response f…
10	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=6 (no response f…
11	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=7 (no response f…
12	2021-07-08 01:4…	10.136.174.2	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
13	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=8 (no response f…
14	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=9 (no response f…
15	2021-07-08 01:4…	183.207.223.17	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
16	2021-07-08 01:4…	183.207.25.193	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
17	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=10 (no response …
18	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=11 (no response …
19	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=12 (no response …
20	2021-07-08 01:4…	111.24.6.33	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
21	2021-07-08 01:4…	111.24.16.206	10.0.2.15	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
22	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=13 (no response …
23	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=14 (no response …
24	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=15 (no response …
25	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=16 (no response …
26	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=17 (no response …
27	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=18 (no response …
28	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=19 (no response …
29	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=20 (no response …
30	2021-07-08 01:4…	10.0.2.15	1.2.3.4	ICMP	42 Echo (ping) request id=0x0000, seq=0/0, ttl=21 (no response …

可以观察到该过程途经的 IP 地址有：10.0.2.2，172.30.10.1，10.136.174.2，183.207.223.17，183.207.25.193，111.24.6.33，111.24.16.206，最终到达 目的地址 1.2.3.4。

Task 1.4：Sniffing and-then Spoofing

程序代码如下：

Open sniff_spoof.py ~/Desktop/Labs_20.04/Network Security/Pack... Sniffing and Spoofing Lab/Labsetup/volumes Save

```python
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=spoof_pkt)
```

通过捕获 ICMP 报文，并将其源宿地址对调，并设置 ICMP 类型为 Reply ，再发出后，就可以实现伪造。

1） 1.2.3.4

在运行本代码之前，在宿主机中 ping 1.2.3.4 ：

```
root@a4c0db4f6b0c:/# 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 ---
1958 packets transmitted, 0 received, 100% packet loss, time 20
08115ms
```

无法 ping 通，因为这个地址是不存在的网络地址。

在虚拟机中运行上述脚本后，再次在宿主机中进行相同的操作：

```
root@a4c0db4f6b0c:/# 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=55.2 ms
64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=20.8 ms
64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=28.1 ms
64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=25.6 ms
64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=23.6 ms
64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=20.7 ms
64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=32.1 ms
64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=21.0 ms
64 bytes from 1.2.3.4: icmp_seq=9 ttl=64 time=25.7 ms
```

成功 ping 通。

同时，虚拟机中出现相应的输出：

```
[07/08/21]seed@VM:~/.../volumes$ sudo python3 sniff_spoof.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

2) 10.9.0.99

在运行本代码之前和之后， 在宿主机中 ping 10.9.0.99 ：

```
root@a4c0db4f6b0c:/# 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
From 10.9.0.5 icmp_seq=7 Destination Host Unreachable
From 10.9.0.5 icmp_seq=8 Destination Host Unreachable
From 10.9.0.5 icmp_seq=9 Destination Host Unreachable
From 10.9.0.5 icmp_seq=10 Destination Host Unreachable
From 10.9.0.5 icmp_seq=11 Destination Host Unreachable
From 10.9.0.5 icmp_seq=12 Destination Host Unreachable
```

都无法 ping 通，因为这个地址是不存在的本机地址，用到 ARP 协议，不会经过路由器。

3) 8.8.8.8

在运行本代码之前， 在宿主机中 ping 8.8.8.8 ：

```
root@a4c0db4f6b0c:/# 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=31.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=38.9 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=18.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=16.2 ms
```

在运行本代码之后， 在宿主机中 ping 8.8.8.8 ：

```
root@a4c0db4f6b0c:/# 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=23.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=64 time=23.6 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=64 time=84.5 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=64 time=16.2 ms
```

都能 ping 通，因为是存在的主机。