Packet Sniffing and Spoofing Lab

57118212 晏宇珂

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

Task 1.1: Sniffing Packets

Task 1.1A

使用`ifconfig`查看广播地址

```
[07/05/21]seed@VM:~/.../Labsetup$ docksh 65
root@VM:/# ifconfig
br-1227ca8c5b08: 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::42:58ff:fe68:6691  prefixlen 64  scopeid 0x20<link>
        ether 02:42:58:68:66:91  txqueuelen 0  (Ethernet)
```

`sniffer.py`代码如下：

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

```
1 from scapy.all import *
2
3 def print_pkt(pkt):
4         pkt.show()
5
6 pkt = sniff(iface='br-1227ca8c5b08', filter='icmp', prn=print_pkt)
```

`root`权限下运行结果如下：

```
[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        = 6315
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0xde7
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xa598
        id        = 0x1
        seq       = 0x1
###[ Raw ]###
           load      = '\xe8\x1a\xe3`\x00\x00\x00\x00\xbe\x16\n\x00\x00\x00\x00\
x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+
,-./01234567'
```

使用普通用户权限运行程序报错：

```
[07/05/21]seed@VM:~/.../volumes$ python3 sniffer.py
Traceback (most recent call last):
  File "sniffer.py", line 6, in <module>
    pkt = sniff(iface='br-1227ca8c5b08', 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
```

从`Trace`可以看出，报错的原因在于普通用户没有权限创建`socket`。

Task 1.1B

仅捕获ICMP报文

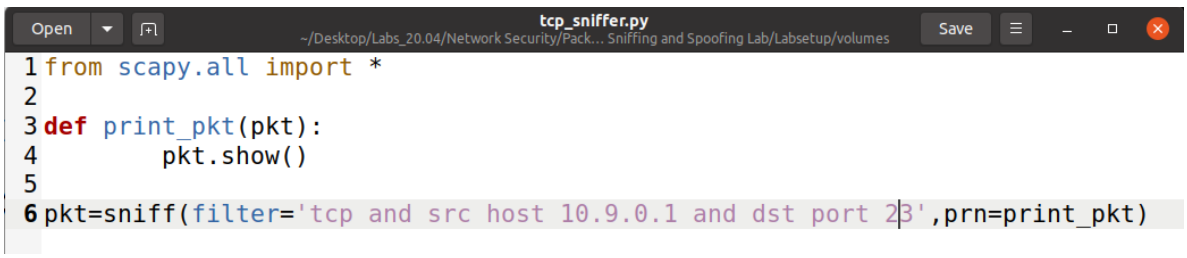
`filter`代码如下：

运行结果如下：

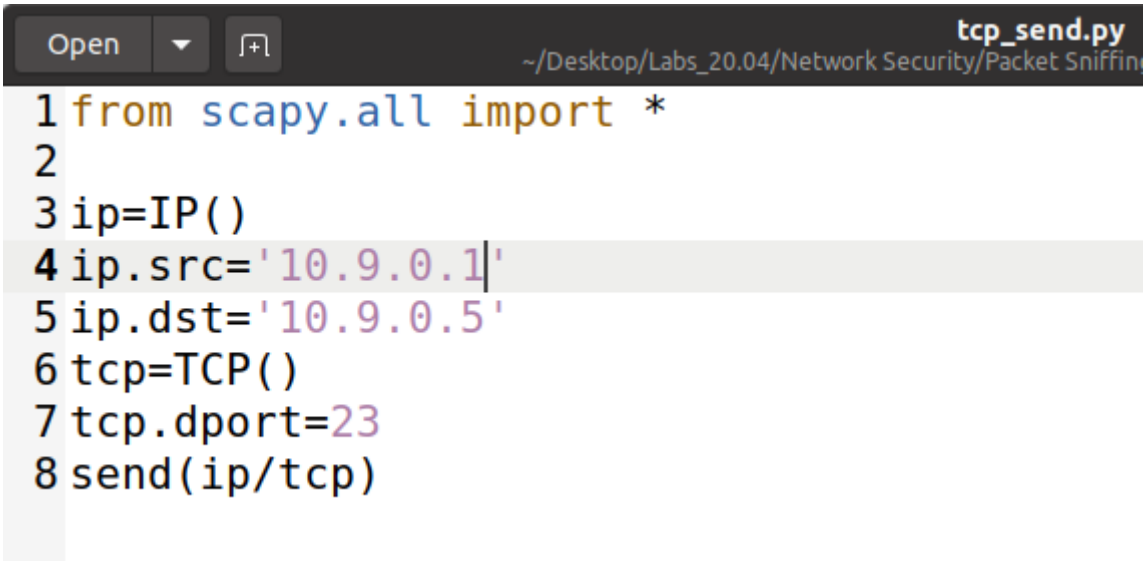
```
[07/05/21]seed@VM:~/.../volumes$ sudo python3 sniffer.py
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:05
  src       = 02:42:58:68:66:91
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 7788
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x826
     src       = 10.9.0.1
     dst       = 10.9.0.5
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xc49b
        id        = 0x2
        seq       = 0x1
###[ Raw ]###
           load      = '\xcd\x1d\xe3`\x00\x00\x00\x00\xbb\x0f\t\x00\x00\x00\x00\
x00\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+
,-./01234567'
```

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：



`tcp_send.py`代码如下：



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   \
###[ TCP ]###
        sport     = ftp_data
        dport     = telnet
        seq       = 0
        ack       = 0
        dataofs   = 5
        reserved  = 0
        flags     = S
        window    = 8192
        chksum    = 0x7ba0
        urgptr    = 0
        options   = []
```

可见成功捕获。

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：

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

```python
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：

Open subnet_send.py ~/Desktop/Labs_20.04/Network Security/Pack... Sniffing and S

```python
from scapy.all import *

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

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       = 10.9.0.5
     dst       = 128.230.164.1
     \options   \
```

成功捕获。

Task 1.2: Spoofing ICMP Packets

`spoofing`代码如下：

Open icmp_spoofing.py ~/Desktop/Labs_20.04/Network Security/Pac...Sniffing and Spoofi

```
1 from scapy.all import *
2
3 ip= IP()
4 ip.dst = '10.9.0.5'
5 b = ICMP()
6 p = ip/b
7 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

```
2021-07-05 13:4… 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 13:4… 10.9.0.5          10.9.0.1          ICMP      42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 3)
```

成功伪装。

```
2021-07-05 14:3… 22.22.22.22       10.9.0.5          ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=64 (reply in 4)
2021-07-05 14:3… 10.9.0.5          22.22.22.22       ICMP      42 Echo (ping) reply    id=0x0000, seq=0/0, ttl=64 (request in 3)
```

也可以伪装成其他任意`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。

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

```
1 from scapy.all import *
2
3 ttl = 1
4 while True:
5         a = IP()
6         a.dst = '1.2.3.4'
7         a.ttl = ttl
8         b = ICMP()
9         send(a/b)
10        ttl += 1
```

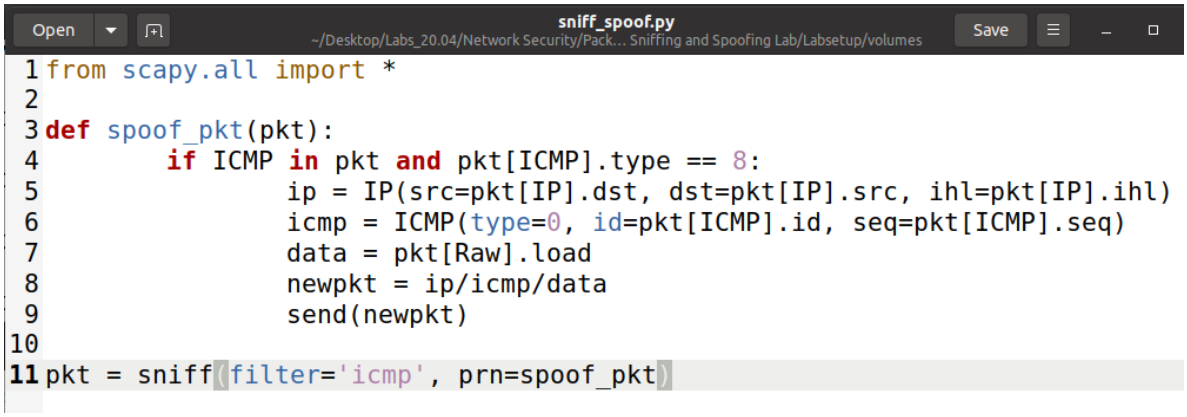
通过一个无限循环，每次将TTL递增，然后使用`Wireshark`查看：

```
1  2021-07-05 18:3… VMware_c0:60:9a   Broadcast         ARP       42 Who has 192.168.43.1? Tell 192.168.43.112
2  2021-07-05 18:3… 72:56:c5:05:b8:5e VMware_c0:60:9a   ARP       60 192.168.43.1 is at 72:56:c5:05:b8:5e
3  2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=1 (no response f…
4  2021-07-05 18:3… 192.168.43.1      192.168.43.112    ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
5  2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=2 (no response f…
6  2021-07-05 18:3… 10.208.64.1       192.168.43.112    ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
7  2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=3 (no response f…
8  2021-07-05 18:3… 10.80.128.141     192.168.43.112    ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
9  2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=4 (no response f…
10 2021-07-05 18:3… 10.80.128.149     192.168.43.112    ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
11 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=5 (no response f…
12 2021-07-05 18:3… 10.80.3.10        192.168.43.112    ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
13 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=6 (no response f…
14 2021-07-05 18:3… 153.3.60.1        192.168.43.112    ICMP      70 Time-to-live exceeded (Time to live exceeded in transit)
15 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=7 (no response f…
16 2021-07-05 18:3… 221.6.2.137       192.168.43.112    ICMP     110 Time-to-live exceeded (Time to live exceeded in transit)
17 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=8 (no response f…
18 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=9 (no response f…
19 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=10 (no response …
20 2021-07-05 18:3… 192.168.43.112    1.2.3.4           ICMP      42 Echo (ping) request  id=0x0000, seq=0/0, ttl=11 (no response …
```

可以观察到途经的`IP`地址有:`192.168.43.1, 10.208.64.1, 10.80.128.141, 10.80.128.149, 10.80.3.10, 153.3.60.1, 221.6.2.137`,最终到达`1.2.3.4`目的地。

Task 1.4: Sniffing and-then Spoofing

代码如下:




```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,就可以伪造了。

- `1.2.3.4`

  在运行本代码之前, 在宿主机中`ping 1.2.3.4`:

  ```
  root@7866d624c949:/# ping 1.2.3.4
  PING 1.2.3.4 (1.2.3.4) 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
  ```

  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@7866d624c949:/# ping 1.2.3.4
  PING 1.2.3.4 (1.2.3.4) 56(84) bytes of data.
  From 10.9.0.1 icmp_seq=1 Destination Net Unreachable
  64 bytes from 1.2.3.4: icmp_seq=1 ttl=64 time=18.3 ms
  64 bytes from 1.2.3.4: icmp_seq=2 ttl=64 time=16.8 ms
  64 bytes from 1.2.3.4: icmp_seq=3 ttl=64 time=21.9 ms
  64 bytes from 1.2.3.4: icmp_seq=4 ttl=64 time=12.8 ms
  64 bytes from 1.2.3.4: icmp_seq=5 ttl=64 time=14.9 ms
  64 bytes from 1.2.3.4: icmp_seq=6 ttl=64 time=15.9 ms
  64 bytes from 1.2.3.4: icmp_seq=7 ttl=64 time=166 ms
  64 bytes from 1.2.3.4: icmp_seq=8 ttl=64 time=15.8 ms
  ```

  成功`ping`通。

  同时,我们在虚拟机中,也可以看到相应的输出:

```
[07/05/21]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.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

说明伪造成功。

- `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@90cc187c9717:/# 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。

- `8.8.8.8`

  在运行本代码之前和之后(上面运行代码了，下面没有运行代码)， 在宿主机中`ping 8.8.8.8`:

```
root@90cc187c9717:/# 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=110 time=77.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=77.0 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=84.7 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=110 time=76.8 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=110 time=63.3 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=110 time=57.4 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=110 time=53.7 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=110 time=55.9 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=110 time=67.0 ms
^Z
[3]+  Stopped                 ping 8.8.8.8
root@90cc187c9717:/# 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=110 time=68.6 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=110 time=61.2 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=110 time=65.8 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=110 time=74.6 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=110 time=76.8 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=110 time=77.9 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=110 time=79.9 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=110 time=76.5 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=110 time=72.0 ms
```

都能`ping`通，因为是存在的主机。