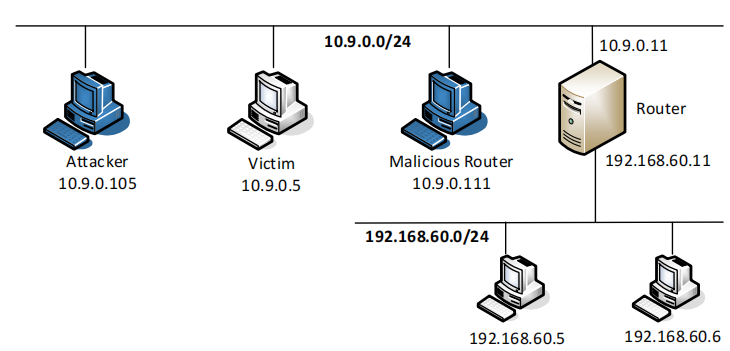
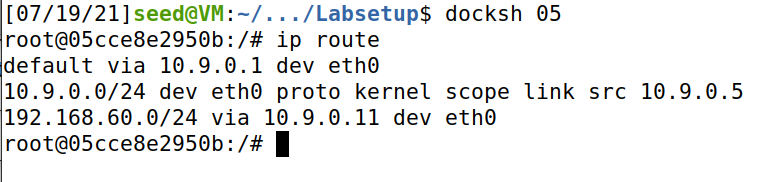
网络结构：



**Task1**

登录到10.9.0.5上查看ip路由信息#ip route，默认网关10.9.0.11

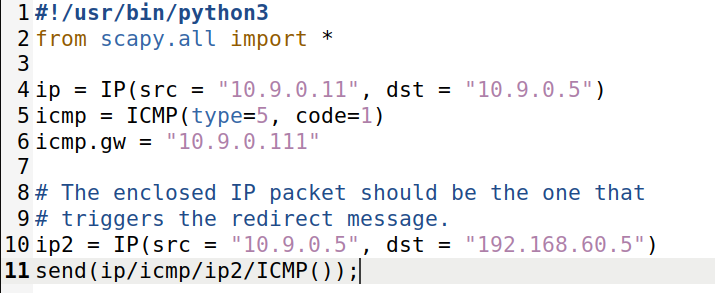


查看路由缓存#ip route show cahe，看到没有写入任何cache

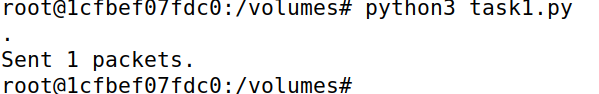


在attacker上部署以下攻击代码

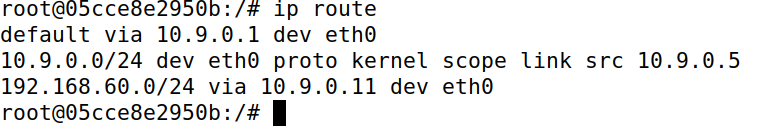
首先伪装成网关10.9.0.11向被攻击者10.9.0.5发送ICMP重定向报文，type=5为重定向报文，code=1为主机重定向，然后推荐网关为10.9.0.111(某恶意网关)，然后包含一个能够触发重定向的IP报文，再附上ICMP()协议段



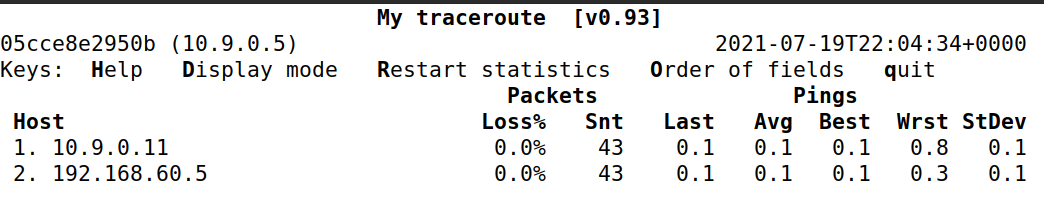
部署完成后，在attacker上实施攻击#ICMP\_redir.py



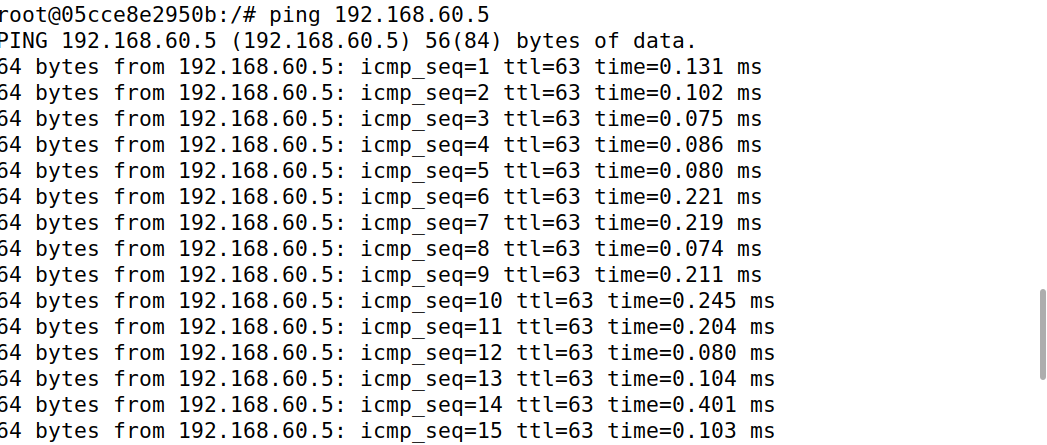
在victim上查看路由缓存和路由表#ip route show cache; #ip route，均没有变化，怀疑失效



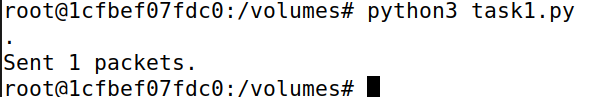
在victim上验证#mtr -n 192.168.60.5，观察到路由任然是通过10.9.0.11，确认攻击失效



ping 192.168.60.5的同时执行攻击代码

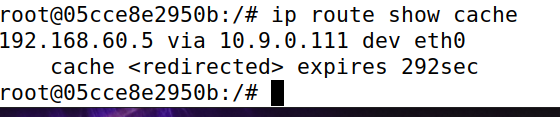


然后在attacker上再次进行攻击，#ICMP\_redir.py

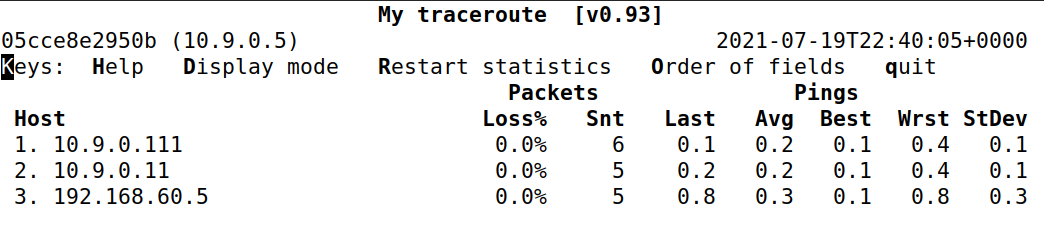


在victim上再次查看路由缓存和路由表#ip route show cache; #ip route

观察到路由缓存被改写了，发向192.168.60.5的ICMP报文被重定向给10.9.0.111，（此时ping还在继续），然而路由表并未被更改，如要更改，需在配置文件中更改。

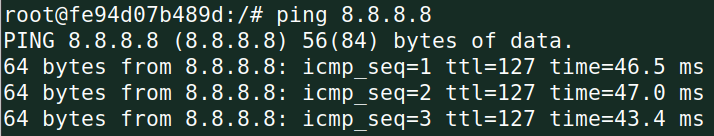


在victim上验证#mtr -n 192.168.60.5，观察到路由的确被更改



首先更改虚拟机网络为桥接模式，windows主机连接手机热点，如果更改ICMP\_redir.py程序中的推荐网关为8.8.8.8，该网关在互联网上

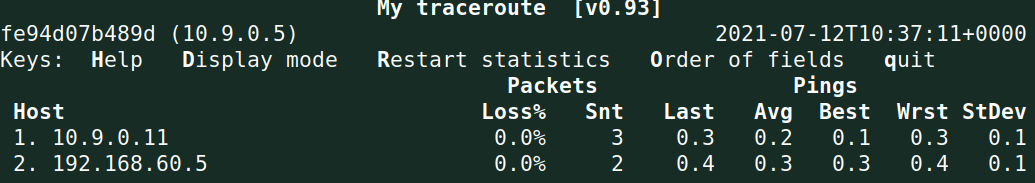
icmp.gw = "8.8.8.8"



则在#ping 192.168.60.5的同时在attacker上进行攻击，观察到路由缓存没有被更改

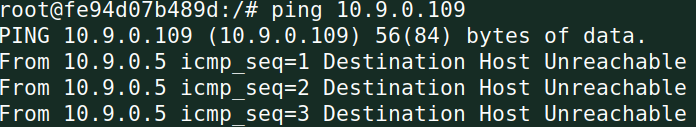


验证#mtr -n 192.168.60.5，路由也的确没有被更改



如果更改ICMP\_redir.py程序中的推荐网关为10.9.0.109，该网关在同一局域网内但不存在

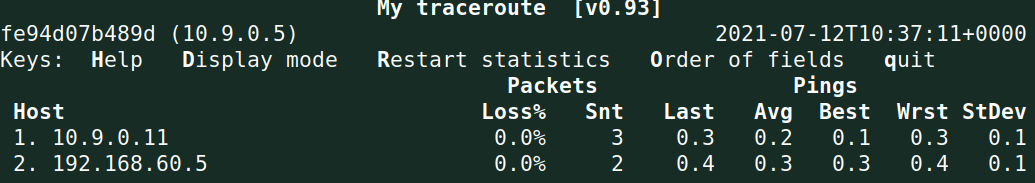
icmp.gw = "10.9.0.109"



则在#ping 192.168.60.5的同时在attacker上进行攻击，观察到路由缓存没有被更改



验证#mtr -n 192.168.60.5，路由也的确没有被更改



结合以上两点得出结论，ICMP重定向攻击只能定向到局域网内存在的主机

更改docker配置文件中恶意路由器的配置，

malicious-router:

image: handsonsecurity/seed-ubuntu:large

container\_name: malicious-router-10.9.0.111

tty: true

cap\_add:

- ALL

sysctls:

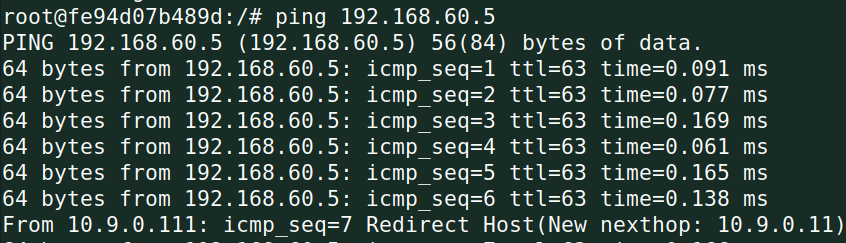
- net.ipv4.ip\_forward=1

- net.ipv4.conf.all.send\_redirects=1

- net.ipv4.conf.default.send\_redirects=1

- net.ipv4.conf.eth0.send\_redirects=1

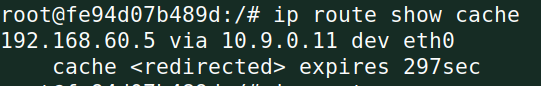
然后重新打开docker，在victim上#ping 192.168.60.5

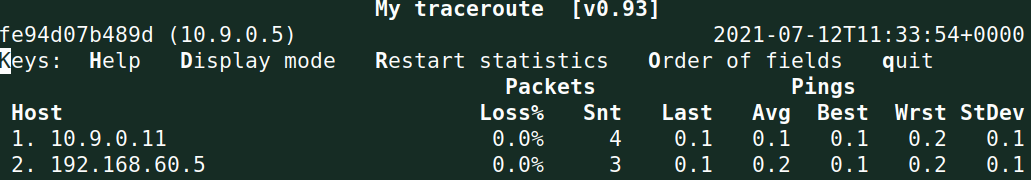


注意这个消息，它提示了我们10.9.0.111又发送了重定向信息，告诉10.9.0.5应该把10.9.0.11作为网关



然后在attacker上攻击，再在victim上查看，路由缓存出现了本来设置中的网关10.9.0.11

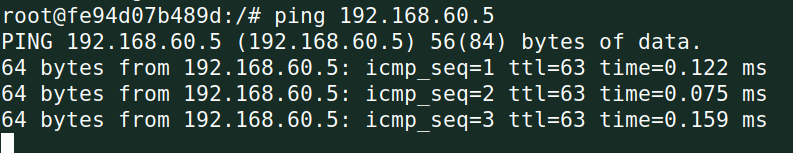




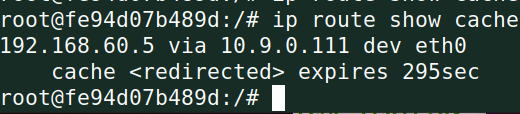
在google上搜索，得出初步结论，send\_redirects=1可以让该节点发送ICMP重定向报文，而10.9.0.111发现直接将10.9.0.11作为网关更近，所以发送此重定向报文

**Task2**

在中间人攻击之前，将docker配置10.9.0.111中的ip转发选项禁用，先在attacker上发动ICMP重定向攻击，指定victim的网关为10.9.0.111，由于ip转发选项被禁用，所以ping收不到回复卡住

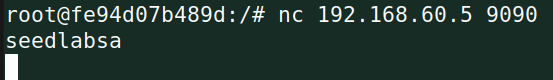


检查路由缓存，确认重定向攻击成功#ip route show cache，随后可以发起攻击



然后在victim上执行#nc 192.168.60.5 9090连接到192.168.60.5

在192.168.60.5上执行#nc -lp 9090监听9090端口文本





10.9.0.111作为中间人，没有开启ip转发，则victim的所有发给192.168.60.5的报文被它收到，同时不会被发送给192.168.60.5主机

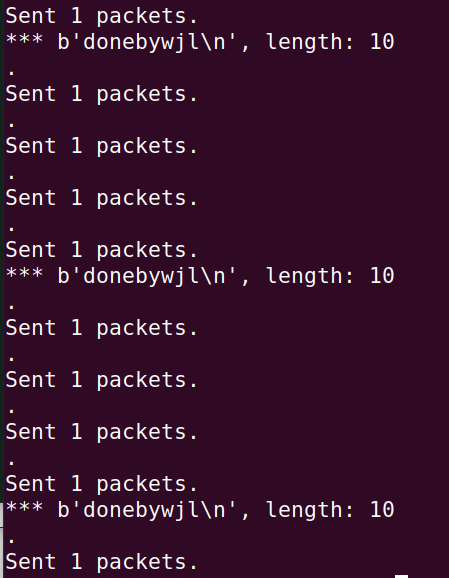
将mitm\_sample.py文件做如下更改，放置到volume文件夹中共享给10.9.0.111

Q4意为监听10.9.0.5发给192.168.60.5的报文，因为我们要伪造的是发送信息的一方发送假消息，注意替换的字符串长度要相等，后续的ack和seq才能对应上

f = 'tcp and src host 10.9.0.5'

data.replace(b'seedlabsa', b'donebywjl')

然后在10.9.0.111上发动攻击#mitm\_sample.py，随后在victim连接到192.168.60.5的shell上输入#seedlabsa，观察到192.168.60.5监听的端口出现被替换后的#donebywjl字符串，攻击成功，但是10.9.0.111的mitm\_sample.py程序不断发送相同的报文（因为自己发送的报文也被监听），浪费了性能，同时，观察到这种方式破坏了攻击的可持续性，在192.168.60.5上部署一个sniff程序，检测所有的tcp报文，观察到此时mitm\_sample.py不停地发送报文导致序列号错乱，所以victim后续的输入都无效

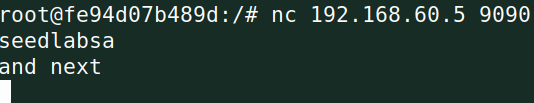


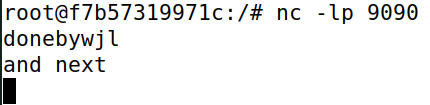
Q5 根据以上结论，根据IP地址过滤行不通，只能根据MAC地址过滤

将mitm\_sample.py更改如下

f = 'tcp and ether src 02:42:0a:09:00:05'

重新发动攻击，观察到除了第一次攻击成功替换字符串，后续在victim的输入任正常，#and next字符串被传输，攻击的可持续性得到保证，攻击者可以根据自己想法替换或不替换





同时，多次发送重复报文的问题也得到解决，mitm\_samply.py执行如下图，提高了程序性能

