ICMP 重定向攻击实验

1 概述

ICMP 重定向是一种由路由器发出的 ICMP 消息。当路由器认为一个数据包被错误地路由时，它会通知发送者使用不同的路由器来转发后续到同一目的地的数据包。ICMP 重定向可以被攻击者用来改变数据包的路由。

本任务的目标是针对受害者发起一个 ICMP 重定向攻击，使得当受害者向 `192.168.60.5` 发送数据包时，它会使用恶意路由器容器（`10.9.0.111`）作为它的路由器。由于恶意路由器由攻击者控制，攻击者可以拦截这些数据包、对其进行修改，然后重新发送出去。这是一种中间人（MITM）攻击的形式。本实验涵盖了以下部分：

- IP 和 ICMP 协议
- ICMP 重定向攻击
- 路由

视频 关于 IP 协议及其攻击的详细内容，可以在以下部分找到：

- SEED Book, *Internet Security: A Hands-on Approach*, 3rd Edition, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net`.
- SEED Lecture 第 4 节，*Internet Security: A Hands-on Approach*, by Wenliang Du. 详情请见 `https://www.handsonsecurity.net/video.html`.

实验环境 本实验在 SEED Ubuntu 20.04 VM 中测试可行。您可以从 SEED 网站上下载我们预先构建好的镜像并在您自己的电脑上运行 SEED VM。然而，大多数 SEED 实验可以在云端进行，您可以按照我们的说明在云端创建 SEED VM。

2 使用容器设置环境

在本实验中，我们需要几台机器。实验环境的搭建如图 1所示。我们将使用容器来搭建此环境。

2.1 容器搭建和命令

请从实验的网站下载 `Labsetup.zip` 文件到你的 VM 中，解压它，进入 `Labsetup` 文件夹，然后用 `docker-compose.yml` 文件安装实验环境。对这个文件及其包含的所有 `Dockerfile` 文件中的内容的详

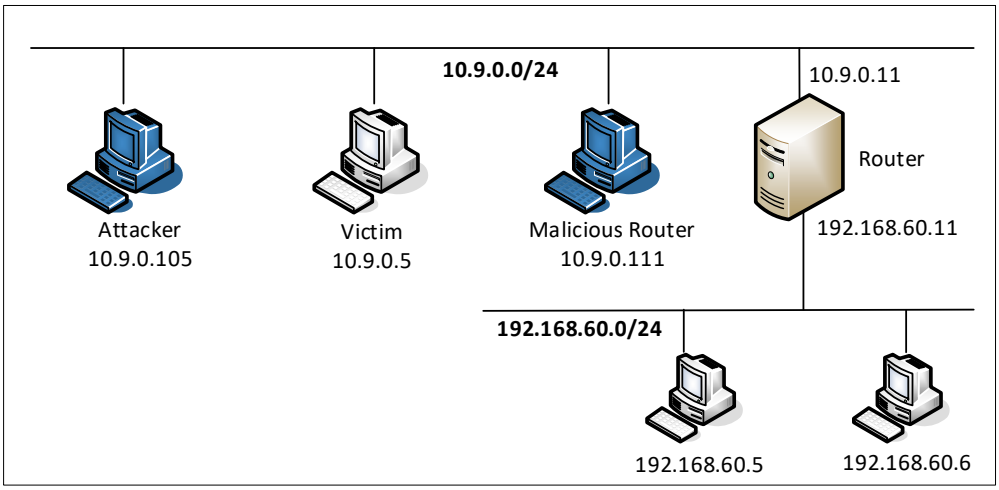

图 1: 实验环境设置

细解释都可以在链接到本实验网站的用户手册 [1] 中找到。如果这是您第一次使用容器设置 SEED 实验环境，那么阅读用户手册非常重要。

在下面，我们列出了一些与 Docker 和 Compose 相关的常用命令。由于我们将非常频繁地使用这些命令，因此我们在 `.bashrc` 文件（在我们提供的 SEED Ubuntu 20.04 虚拟机中）中为它们创建了别名。

```
$ docker-compose build  # 建立容器镜像
$ docker-compose up     # 启动容器
$ docker-compose down   # 关闭容器

// 上述 Compose 命令的别名
$ dcbuild       # docker-compose build 的别名
$ dcup          # docker-compose up 的别名
$ dcdown        # docker-compose down 的别名
```

所有容器都在后台运行。要在容器上运行命令，我们通常需要获得容器里的 Shell 。首先需要使用 `docker ps` 命令找出容器的 ID ，然后使用 `docker exec` 在该容器上启动 Shell 。我们已经在 `.bashrc` 文件中为这两个命令创建了别名。

```
$ dockps        // docker ps --format "{{.ID}}  {{.Names}}" 的别名
$ docksh <id>   // docker exec -it <id> /bin/bash 的别名

// 下面的例子展示了如何在主机 C 内部得到 Shell
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7
```

[1] 如果你在部署容器的过程中发现从官方源下载容器镜像非常慢，可以参考手册中的说明使用当地的镜像服务器

```
$ docksh 96
root@9652715c8e0a:/#

// 注: 如果一条 docker 命令需要容器 ID, 你不需要
//     输入整个 ID 字符串。只要它们在所有容器当中
//     是独一无二的, 那只输入前几个字符就足够了。
```

如果你在设置实验环境时遇到问题，可以尝试从手册的 “Miscellaneous Problems” 部分中寻找解决方案。

2.2 关于攻击者容器

在本实验中，我们可以选择使用虚拟机或攻击者容器作为攻击机器。如果查看 Docker Compose 文件，你会看到攻击者容器的配置与其他容器不同。以下是其中的区别：

- *共享文件夹*。当使用攻击者容器发起攻击时，我们需要将攻击代码放入攻击者容器中。在虚拟机中进行代码编辑比在容器中更为方便，因为我们可以使用我们喜欢的编辑器。为了使虚拟机和容器共享文件，我们使用 Docker `volumes` 在虚拟机和容器之间创建了一个共享文件夹。如果你查看 Docker Compose 文件，就会发现我们已经在某些容器中添加了以下条目。它表示将主机（即 VM）上的 `./volumes` 文件夹挂载到容器内的 `/volumes` 文件夹。我们在虚拟机上将代码写入 `./volumes` 文件夹，就可以在容器内使用它们。

  ```
  volumes:
        - ./volumes:/volumes
  ```

- *特权模式*。为了能够在运行时修改内核参数（使用 `sysctl`），例如为了启用 IP 转发，容器需要被赋予特权。这是通过在容器中的 Docker Compose 文件包含以下条目来实现的。

  ```
  privileged: true
  ```

3 任务 1：发起 ICMP 重定向攻击

在 Ubuntu 操作系统中，有针对 ICMP 重定向攻击的防范措施。在 Compose 文件中，我们已经关闭了这种防范措施，也就是允许容器接受 ICMP 重定向消息。

```
// 在docker-compose.yml
sysctls:
  - net.ipv4.conf.all.accept_redirects=1

// 要开启防护, 请将其值设置为0
# sysctl net.ipv4.conf.all.accept_redirects=0
```

对于本任务，我们将从攻击者容器对受害者的容器发起攻击。当前的配置中，受害者将使用路由器容器（192.168.60.11）作为通往 192.168.60.0/24 网络的路由器。如果在受害者的容器上运行 ip route，我们将看到以下内容

```
# 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
```

代码框架。 我们提供了一个代码框架，学生们需要在标记为 @@@@ 的地方填入正确的值。

```
#!/usr/bin/python3

from scapy.all import *

ip = IP(src = @@@@,  dst = @@@@)
icmp = ICMP(type=@@@@, code=@@@@)
icmp.gw = @@@@

# ICMP 重定向包必须携带触发它的那个原始IP数据包。
ip2 = IP(src = @@@@, dst = @@@@)
send(ip/icmp/ip2/ICMP());
```

验证 ICMP 重定向消息不会影响路由表，而是会改变路由缓存。路由缓存中的条目比路由表中的条目优先级高，会被先用，直到过期为止。请使用以下命令显示和清理缓存内容。

```
// 显示路由缓存
# ip route show cache
192.168.60.5 via 10.9.0.111 dev eth0
    cache <redirected> expires 296sec

// 清理路由缓存
# ip route flush cache
```

请在受害者的机器上进行 traceroute 操作，看看数据包是否被重定向。

```
# mtr -n 192.168.60.5
```

注意事项： 如果我们伪造重定向包，但受害者的机器在攻击期间没有发送过 ICMP 数据包，则该攻击将不会成功。这是因为操作系统内核会在接受 ICMP 重定向消息之前进行一些类型的检查。它会验证该 ICMP 重定向是否由其自身发出的数据包触发，即它会检查重定向包内的 ip2。这种检查的严格程度取决于操作系统。

对于 Ubuntu 20.04，内核只是验证重定向包中的 ip2 是否与触发 ICMP 重定向的实际数据包的类型和目的 IP 地址匹配。然而，如果你在苹果机器上进行此实验，你的虚拟机版本可能是 Ubuntu 22.04

或更高版本，检查会更加严格。要确保伪造包中的 ip2 能够通过检查，最简单的方法是从受害者的机器中捕获一个数据包。不过这并不是必要的。鼓励学生们使用其他方法进行攻击。

问题。 在成功发动攻击后，请做以下实验，观察你的攻击是否仍然成功。请解释你观察到的现象：

- 问题 1：能否使用 ICMP 重定向攻击将流量重定向到远程机器？即，给 icmp.gw 赋予的 IP 地址是一个不在本地局域网上的计算机。请展示实验结果并解释你观察到的现象。
- 问题 2：能否使用 ICMP 重定向攻击将流量重定向到同一网络中的不存在的机器上？即，给 icmp.gw 赋予的 IP 地址是一个在本地局域网上不存在的计算机。请展示实验结果并解释你观察到的现象。
- 问题 3：查看 docker-compose.yml 文件，你会找到恶意路由器容器的以下条目。这些条目的目的是什么？将它们的值改为 1，并再次启动攻击。请描述并解释你观察到的现象。

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

4 任务 2：发起 MITM 攻击

通过 ICMP 重定向攻击,可以使受害者使用我们的恶意路由器(10.9.0.111)作为通往 192.168.60.5 的目标路由。因此，从受害者机器发送到此目的地的所有数据包都将通过恶意路由器进行转发。我们希望能够修改受害者的数据包。

在发起 MITM 攻击之前，我们在受害者的容器中使用 netcat 启动一个 TCP 客户端和服务端程序。命令如下：

```
// 在目标容器192.168.60.5上启动netcat服务：
# nc -lp 9090

// 在受害者的容器中连接到服务器：
# nc 192.168.60.5 9090
```

一旦建立连接，你可以在受害者机器上输入消息。每行消息都会被放入一个 TCP 数据包发送至目的地，目的地会简单地显示这些消息。你的任务是将每条消息中的你名字（拼音）出现的地方替换为一系列 A。序列的长度应与你名字相同，否则可能会扰乱 TCP 序号，从而导致整个 TCP 连接失败。你需要使用真实的名字，以便我们知道工作是由谁完成的。

禁用 IP 转发。 在设置中，恶意路由器启用了 IP 转发功能，因此它像一个路由器一样为其他机器转发数据包。当我们发起 MITM 攻击时，我们必须停止转发 IP 数据包。我们将拦截这些数据包，对其进行修改，然后重新发送出去。要做到这一点，我们只需在恶意路由器上禁用 IP 转发即可。

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

MITM 代码。 一旦禁用了 IP 转发，我们的程序需要接管数据包转发角色，当然是在修改数据包后再发送出去。由于数据包的目的地不是我们自己，内核不会将这个数据包传递给我们，而是会丢弃这个数据包。然而，如果我们的程序是一个嗅探器程序，我们将从内核那里获取到这个数据包。因此，我们将使用嗅探和伪造技术来实施这种 MITM 攻击。以下是一个示范程序，用于捕获 TCP 数据包、对其进行修改后再重新发送。你可以从实验配置文件中找到代码。

Listing 1: 示范代码: `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)))

       # 字符串替换
       newdata = data.replace(b'seedlabs', b'AAAAAAAA')

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

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

请注意，上述代码捕获了所有 TCP 数据包，包括由程序本身生成的数据包。这是不可取的，因为它会导致无限循环，影响性能。学生需要更改过滤器以确保自己发出的数据包不会被捕捉到。

问题。 成功发动攻击后，请回答以下问题：

- 问题 4：在你的 MITM 程序中，你只需要捕捉一个方向的数据流量。请指出你选的方向，并解释为什么。
- 问题 5：在 MITM 程序中，当你从 A（`10.9.0.5`）捕获 `nc` 流量时，你可以使用 A 的 IP 地址或 MAC 地址作为过滤器的一部分。其中有一种选择是不好的，它会引发问题。请尝试两种方法，通过实验结果展示哪种选择是正确的，并解释你的结论。

5 提交

你需要提交一份带有截图的详细实验报告来描述你所做的工作和你观察到的现象。你还需要对一些有趣或令人惊讶的观察结果进行解释。请同时列出重要的代码段并附上解释。只是简单地附上代码不加以解释不会获得学分。