网络安全综合实践

（二）

ARP 缓存中毒实验

华中科技大学网络空间安全学院

二零二一年十二月

实验 8 ARP 缓存中毒实验

1.1 实验目的

地址解析协议 （ARP） 是一种通信协议，用于发现给定 IP 地址的链路层地址（如 MAC 地址）。ARP 协议是一个非常简单的协议，它没有任何安全措施。ARP 缓存中毒攻击是针对 ARP 协议的常见攻击。在这种攻击下，攻击者可以欺骗受害者接受伪造的 IP 到 MAC 映射。这可能会导致受害者的数据包被重定向到计算机与伪造的 MAC 地址。

本实验的目的是让学生获得 ARP 缓存中毒攻击的第一手经验，并了解此类攻击可造成哪些损害。特别是，学生将使用 ARP 攻击来发起中间人攻击，攻击者可以在攻击中拦截和修改两个受害者 A 和 B 之间的数据包。

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，至少需要 3 台机器，一台做网关，一台做攻击者，另一台做受害者。考虑到实验室上外网存在困难，因此，另外搭建了一台机器作为外网服务器。因此，考虑采用虚拟机+docker 来搭建网络实验环境。

容器 Server2 用作外网服务器，虚拟机作为网关，容器 HostM 用于攻击，容器 HostA 用作受害者。本实验的网络拓扑如图 1.1 所示：

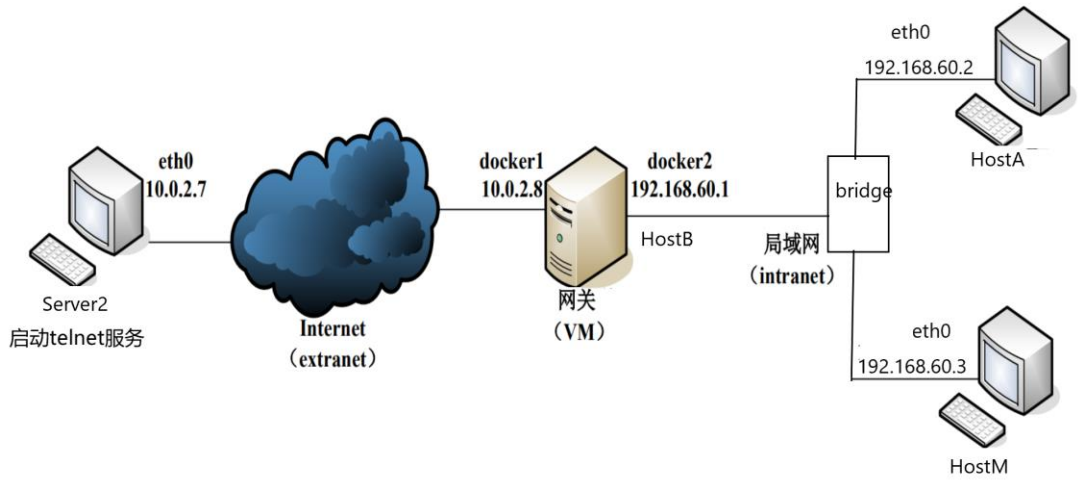

图 1.1 实验网络环境

为了实验上面的网络环境，我们需要执行以下操作：

```
在 VM 上创建 docker 网络 extranet
$ sudo docker network create --subnet=10.0.2.0/24 --gateway=10.0.2.8 --opt
"com.docker.network.bridge.name"="docker1" extranet

在 VM 上创建 docker 网络 intranet
$ sudo docker network create --subnet=192.168.60.0/24 --gateway=192.168.60.1 --
opt "com.docker.network.bridge.name"="docker2" intranet

在 VM 上新开一个终端，创建并运行容器 Server2
$sudo docker run -it --name=Server2 --hostname=Server2 --net=extranet --
ip=10.0.2.7 --privileged "seedubuntu" /bin/bash

在 VM 上新开一个终端，创建并运行容器 HostA
$sudo docker run -it --name=HostA --hostname=HostA --net=intranet --
ip=192.168.60.2 --privileged "seedubuntu" /bin/bash

在 VM 上新开一个终端，创建并运行容器 HostM
$sudo docker run -it --name=HostM --hostname=HostM --net=intranet --
ip=192.168.60.3 --privileged "seedubuntu" /bin/bash
```

1.3 实验内容

地址解析协议（ARP）用于发现给定 IP 地址的链路层通信协议地址（如 MAC 地址）。ARP 协议是一个非常简单的协议，它没有实施任何安全措施。ARP 缓存中毒攻击是一种常见的针对 ARP 协议的攻击。在这种攻击下，攻击者可以欺骗受害者接受伪造的 IP 到 MAC 映射，这会导致受害者的数据包被重定向到带有伪造 MAC 的计算机地址。

本实验的目的是让学生了解 ARP 缓存攻击原理，并了解这种攻击会造成什

么伤害。特别是，学生将使用 ARP 攻击发起中间人攻击，攻击者可以在中间拦截和修改两个受害者 A 和 B 之间的数据包。

1.3.1 任务 1：ARP 缓存中毒

此任务的目的是使用数据包欺骗来对目标发起 ARP 缓存中毒攻击，这样当两台受害计算机 A 和 B 尝试相互通信时，攻击者会截获其数据包并对数据包作更改，从而成为介于 A 和 B 之间的中间人，这就是所谓的中间人（MITM） 攻击。在这个实验室中，我们使用 ARP 缓存中毒来进行 MITM 攻击。

下面的代码框架展示了如何使用 Scapy 构造 ARP 包。

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

E = Ether()
A = ARP()

pkt = E/A
sendp(pkt)
```

上面的程序构造并发送一个 ARP 包。在构造自己的 ARP 包时，请将必要的属性名称/值设置为自定义的值。我们可以使用 ls（ARP）来查看 ARP 类的属性名。如果一个字段未设置，则将使用默认值（请参见输出的第三列）：

```
>>> from scapy.all import *
>>> ls(ARP)
hwtype     : XShortField                         = (1)
ptype      : XShortEnumField                     = (2048)
hwlen      : ByteField                           = (6)
plen       : ByteField                           = (4)
op         : ShortEnumField                      = (1)
hwsrc      : ARPSourceMACField                   = (None)
psrc       : SourceIPField                       = (None)
hwdst      : MACField                            = ('00:00:00:00:00:00')
pdst       : IPField                             = ('0.0.0.0')
```

在这个任务中，我们有三个机器，A（User）、B(Gateway)和 M(Attacker)。我们想攻击 A 的 ARP 缓存，这样在 A 的 ARP 缓存中实现了以下结果：

B 的 IP 地址----->M 的 MAC 地址

进行 ARP 缓存中毒攻击的方法有很多种。学生需要尝试以下三种方法，并报告每个方法是否有效。

- **任务 1A（使用 ARP 请求）**

在主机 M 上，构造一个 ARP 请求包并发送给主机 A。在 A 的 ARP 缓存中检查 B 的 IP 地址是否映射为 M 的 MAC 地址。

- **任务 1B（使用 ARP 答复）**

在主机 M 上，构造一个 ARP 响应包并发送给主机 A。在 A 的 ARP 缓存中检查 B 的 IP 地址是否映射为 M 的 MAC 地址。

- **任务 1C（使用免费 ARP）**

在主机 M 上构造一个 ARP 免费报文。ARP 免费报文是一种特殊的 ARP 请求包。当主机需要向所有其他机器的 ARP 缓存更新过期信息时使用。免费 ARP 报文具有以下特征：

（1） 源和目的 IP 地址均为发布免费 ARP 的主机地址

（2） ARP 头部和以太帧头部的目的 MAC 地址都是广播 MAC 地址（FF:FF:FF:FF:FF:FF）

（3） 不需要回应

1.3.2 任务 2：用 ARP 缓存中毒对 telnet 进行中间人（MITM）攻击

telnet 中间人攻击选做，但是此部分的文档需要看，后面 netcat 中间人攻击采取的攻击步骤是一样的。

主机 A 和 B 正在使用 Telnet 进行通信，而主机 M 想要截获它们的通信，因此可以对 A 和 B 之间发送的数据进行更改。设置如图 1 所示。

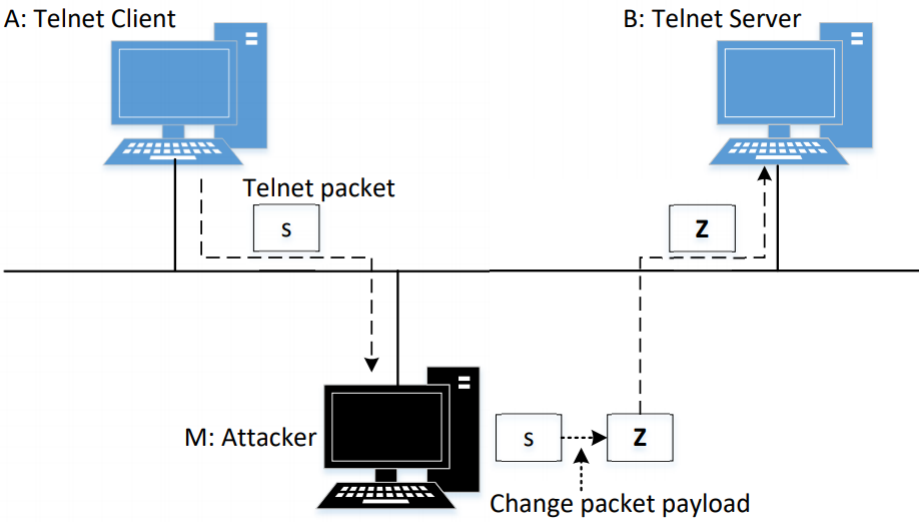

图 1 针对 telnet 的中间人攻击

第 1 步（实施 ARP 缓存中毒攻击）。首先，主机 M 在 A 和 B 之间实施 ARP 缓存中毒攻击，使得 A 的 ARP 缓存中，B 的 IP 地址映射为 M 的 MAC 地址，在 B 的 ARP 缓存中，A 的 IP 地址也映射到 M 的 MAC 地址。在此步骤之后，在 A 和 B 之间发送数据包将全部发送给 M。我们将使用任务 1 中的 ARP 缓存中毒攻击来实现这个目标。

第 2 步（测试）。攻击成功后，请尝试在主机 A 和主机 B 之间 ping 对方，然后报告你的观察结果。请在报告中显示 Wireshark 结果。

第 3 步（打开 ip 转发）。打开主机 M 上的 IP 转发，因此 M 将转发 A 和 B 的报文。 运行下面的命令，再重做第 2 步，描述你的观察结果。

```
$sudo sysctl net.ipv4.ip_forward=1
```

第 4 步（发动 MITM 攻击）。我们准备对 A 和 B 之间的 Telnet 数据进行更改。假设 A 是 Telnet 客户机，B 是 Telnet 服务器。在 A 连接到 B 上的 Telnet 服务后, 对于在 A 的 Telnet 窗口中键入的每一个按键，都会生成一个 TCP 包并发送给 B。我们想要截获 TCP 数据包，并用固定字符（如 Z）替换每个键入的字符。这样，用户不管在 A 上输入什么字符，Telnet 总是显示 Z。

在前面的步骤中，我们可以将 TCP 数据包重定向到主机 M，但不转发，用

一个伪造的数据包来替换它们。我们编写一个嗅探和欺骗程序完成这个目标。特别是，我们要做以下工作：

（1） 我们首先保持 IP 转发打开，这样就可以成功地在 A 和 B 之间创建 Telnet 连接。一旦建立了连接，我们将使用以下命令关闭 IP 转发。请在 A 的 Telnet 窗口中键入一些内容，并报告你的观察结果。

```
$sudo sysctl net.ipv4.ip_forward=0
```

（2）在主机 M 上运行嗅探和欺骗程序，对捕获到的从 A 发送到 B 的报文，伪造一个数据包，但是使用了不同的 TCP 数据。对于从 B 到 A 的数据包（Telnet 响应），我们不做进行任何更改，使伪造的数据包与原始数据包完全相同。

嗅探和监听程序框架如下：

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

def spoof_pkt(pkt):
    print("Original Packet.....")
    print("Source IP : ", pkt[IP].src)
    print("Destination IP :", pkt[IP].dst)

    a = IP()
    b = TCP()
    data = pkt[TCP].payload
    newpkt = a/b/data

    print("Spoofed Packet.....")
    print("Source IP : ", newpkt[IP].src)
    print("Destination IP :", newpkt[IP].dst)
    send(newpkt)
```

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

上面的程序嗅探所有的 TCP 包，然后根据捕获的 TCP 包伪造一个新的 TCP 包。请进行必要的更改以区分数据包是从 A 还是 B 发送的。如果从 A 发送的，将新数据包的所有属性名称/值设置为与原始数据包相同，并将有效负载中的每个字母数字字符（通常每个数据包中只有一个字符）替换为字符 Z。如果捕获的数据包是从 B 发送的，则不进行任何更改。

在 Telnet 中，我们在 Telnet 窗口中键入的每个字符都将触发一个 TCP 包。因此，客户端到服务器的一个典型的 Telnet 数据包，有效负载仅包含一个字符。这个字符会在服务器回显，客户端在其窗口中显示该字符。因此，我们在客户端窗口中看到的并不是输入字符的直接结果，在客户端窗口中输入的内容，在显示之前进行了一次往返。如果网络断开，无论我们在客户端键入什么，都不会在窗口显示，直到网络恢复。同样，如果攻击者在这个往返过程中，将字符更改为 Z， Z 将显示在 Telnet 客户端窗口中。

下面我们总结一下 MITM 攻击步骤：

- 对主机 A 和 B 执行 ARP 缓存中毒攻击。
- 在主机 M 上打开 IP 转发。
- 从主机 A telnet 到主机 B
- 建立 Telnet 连接后，关闭 IP 转发。
- 主机 M 上 进行嗅探和欺骗攻击。

1.3.3 任务 3：用 ARP 缓存中毒对 netcat 进行中间人（MITM）攻击

因为 telnet 服务的特点，在 telnet 会话建立之后，客户端和服务器之间每次传输的字符为单个字符，针对 telnet 的中间人攻击，修改的是单个字符。除了 telnet，还可以用 netcat 来进行实验。

netcat 是网络工具中的瑞士军刀，它能通过 TCP 和 UDP 在网络中读写数据。通过与其他工具结合和重定向，可以在脚本中以多种方式使用它。使用 netcat 命令所能完成的事情令人惊讶。

netcat 所做的就是在两台电脑之间建立链接并返回两个数据流，在这之后所能做的事就看你的想像力了。你能建立一个服务器，传输文件，与朋友聊天，传输流媒体或者用它作为其它协议的独立客户端。下面给出 netcat 的使用例子。

（1）聊天：一个在端口监听（Server），另一个主机（Client）向该端口建立连接，两台主机之间互发信息进行通信，实现聊天功能

Server：nc -l -p 3000　　//表示在 tcp 3000 端口监听

Client：nc　serverIP　3000　//连接服务器的 3000 端口

在两台机器之间就可以发送信息了，一方在屏幕上发送的消息，另一方可以在屏幕上显示出来。

（2）　传输文件，将客户端的 file.txt 文件传到 server 上

Server：nc -l -p 1234 >file.txt　　//将从端口 1234 获得的数据写入 file.txt

Client:　nc serverip 1234 <file.txt　//将 file.txt 文件内容传到服务器的 1234 端口

Netcat 也可以将客户端输入的数据，从服务器进行回显，跟 telnet 协议不一样的是，netcat 在 TCP 连接建立以后，没有复杂的终端协商过程，服务器根据客户端的输入进行回显，而且可以一次传输多个字符，当客户端输入多个字符，敲入回车以后，一次性将输入的数据传输给服务器，服务器再将相同的数据回传到客户端。

针对 netcat 的中间人攻击过程和针对 1.3.2 节中 telnet 中间人攻击的过程一样，这里就不赘述了。

此环节的任务是对 netcat 传输的字符串进行修改，修改为“学号_姓名拼音”，可以做一下如下的测试：

（1）　被修改的字符串和修改后的字符串长度相等；

（2）　被修改的字符串和修改后的字符串长度不等。

观察一下上述两种情况发生以后，是否还能持续通信？

1.4 实验小结

- ARP 协议：ARP（Address Resolution Protocol，地址解析协议）是一个位于 TCP/IP 协议栈中的网络层，负责将某个 IP 地址解析成对应的 MAC 地址
- ARP 协议的基本功能：通过目标设备的 IP 地址，查询目标设备的 MAC 地址，以保证通信的进行。
- ARP 攻击的局限性：ARP 攻击仅能在局域网进行，无法对外网进行攻击。
- ARP 攻击的攻击原理：ARP 攻击就是通过伪造 IP 地址和 MAC 地址实现 ARP 欺骗，能够在网络中产生大量的 ARP 通信量使网络阻塞，攻击者只要持续不断的发出伪造的 ARP 响应包就能更改目标主机 ARP 缓存中的 IP-MAC 条目，造成网络中断或中间人攻击

针对于以上的攻击手段，最简单的方法就是设置静态 ARP，这样就无法修改对应的的 ARP 条目。除非必要，否则停止 ARP 使用，把 ARP 作为永久条目保存在对应表中。除此之外，还可以使用 ARP 服务器，但是要确保 ARP 服务器不会被入侵。还可以使用代理、定期轮询等手段。