实验二 ARP 与 DNS 协议分析

2.1 实验目的

分析 ARP 协议报文首部格式以及在同一网段内和不同网段间的解析过程，分析 DNS 协议的工作过程。

2.2 实验内容

（1）利用校园网及云服务器搭建内网、外网环境；

（2）用 Wireshark 截获 ARP 报文，分析报文结构及 ARP 协议在同一网段和不同网段间的解析过程；

（3）用 Wireshark 截取 DNS 报文，分析 DNS 工作过程。

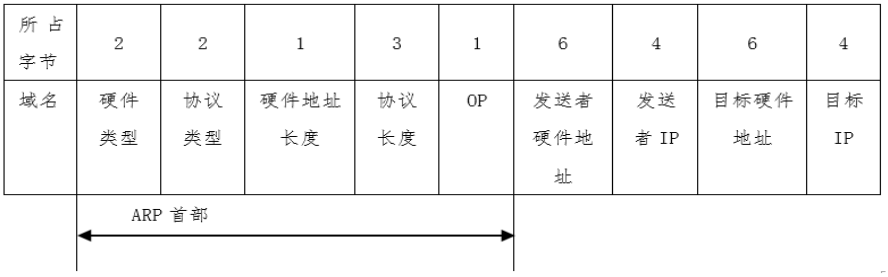
2.3 ARP 协议的实验

2.3.1 ARP 协议概述

地址解析协议（ARP，Address Resolution Protocol）工作在数据链路层，在本层和硬件接口联系，同时对上层提供服务。IP 数据包在以太网传送中，以太网设备并不识别 IP 地址，而是以 48 位以太网地址传输数据包。因此，必须把 IP 目的地址转换成以太网目的地址。

在以太网中，一个主机要和另一个主机进行直接通信，必须要知道目标主机的 MAC 地址。但这个目标 MAC 地址是通过地址解析协议 ARP 获得的。ARP 协议用于将网络中的 IP 地址解析为的硬件地址（MAC 地址），以保证通信的顺利进行。反向地址转换协议 RARP 则用 MAC 地址在 RARP 服务器请求相应的 IP 地址。

2.3.2 ARP 的报文格式



硬件类型字段指明了发送方想知道的硬件接口类型，以太网的值为 1。协议类型字段指明了发送方提供的高层协议类型，IP 为 0800（16 进制）。硬件地址长度和协议长度指明了硬件地址和高层协议地址的长度，这样 ARP 报文就可以在任意硬件和任意协议的网络中使用。操作字段 OP 用来表示这个报文的目的，ARP 请求为 1，ARP 响应为 2，RARP 请求为 3，RARP 响应为 4。

2.3.3 ARP 的工作原理

首先，每台主机都会在自己的 ARP 缓冲区 (ARP Cache)中建立一个 ARP 列表，以表示 IP 地址和 MAC 地址的对应关系。

当源主机需要将一个数据包要发送到目的主机时，会首先检查自己 ARP 列表中是否存在该 IP 地址对应的 MAC 地址，如果有，就直接将数据包发送到这个 MAC 地址；如果没有，就向本地网段发起一个 ARP 请求的广播包，查询此目的主机对应的 MAC 地址。此 ARP 请求数据包里包括源主机的 IP 地址、硬件地址、以及目的主机的 IP 地址。

网络中所有的主机收到这个 ARP 请求后，会检查数据包中的目的 IP 是否和自己的 IP 地址一致。如果不相同就忽略此数据包；如果相同，该主机首先将发送端的 MAC 地址和 IP 地址添加到自己的 ARP 列表中，如果 ARP 表中已经存在该 IP 的信息，则将其覆盖，然后给源主机发送一个 ARP 响应数据包，告诉对方自己是它需要查找的 MAC 地址。

源主机收到这个 ARP 响应数据包后，将得到的目的主机的 IP 地址和 MAC 地址添加到自己的 ARP 列表中，并利用此信息开始数据的传输。如果源主机一直没有收到 ARP 响应数据包，表示 ARP 查询失败。

ARP 在同一网段和不同网段解析过程有所不同，以上步骤说明了目的地址与源地址在同一网段时的解析过程。当不在同一网段时，主机首先查询的是它的默认网关的硬件地址，数据包也是先送到默认网关。

2.4 DNS 协议概述

域名系统（服务）协议（DNS，Domain Name System）是一种分布式网络目录服务，主要用于域名与 IP 地址的相互转换，以及控制因特网的电子邮件的发送。

2.4.1 DNS 报文格式

DNS协议报文格式

2.4.2 DNS 工作原理

用浏览器访问域名（如 www.yahoo.com），操作系统首先查找本地 DNS 解析器缓存看是否存在这个网址映射，如果有则返回，完成域名解析。

如果本地 DNS 解析器缓存没有相应网址映射，则会发消息给本机首选 DNS 服务器解析，如果要查询的域名包含在本地配置区域资源中，则返回解析记过给客户端，完成域名解析，此解析具有权威性。

如果要查询域名不由本地 DNS 服务器区域解析，但该服务器已缓存了此网址映射关系，则调用这个 IP 地址映射，完成域名解析，此解析不具有权威性。

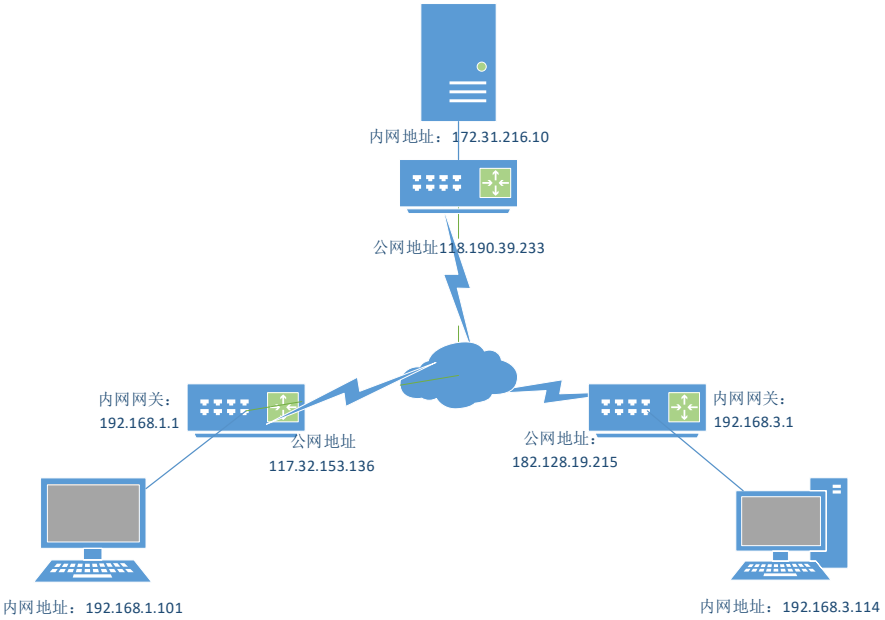
如果本地 DNS 服务器解析失败，则根据本地 DNS 服务器的设置进行转发解析或者根查询，结果返回给本地 DNS 服务器，由此 DNS 服务器返回给客户机

2.5 实验环境与分组

每 2 名同学一组，以现有的校园网络环境及云服务器搭建内网、外网网络。

2.6 实验网拓扑结构

以各组现有网络实际情况为准，标注内网、公网地址。



2.7 ARP 协议分析

2.7.1 同一网段的 ARP 协议分析

步骤 1：在计算机终端的命令行窗口执行命令：

执行“arp –a”观察 arp 缓存；

执行“arp –d”命令清空 arp 缓存。

步骤 2：在计算机终端上运行 Wireshark 截获报文，在命令行窗口 ping 同一网段的另一设备地址。执行完之后，停止报文截获，分析截获的报文。

步骤 3：在命令行窗口执行“arp –a”，记录结果。

2.7.2 不同网段的 ARP 协议分析

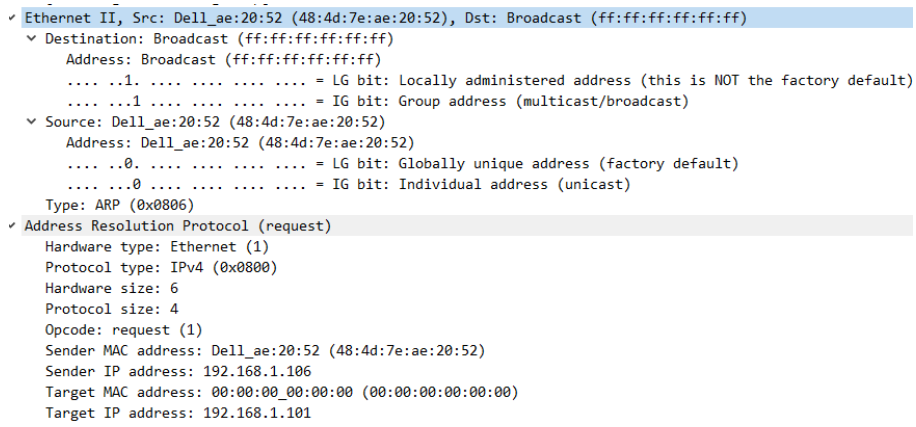
步骤 1：执行“arp –d”清空缓存。在计算机终端运行 Wireshark 捕获报文，在命令窗口 ping 外网地址（云服务器地址）。执行完之后，停止报文截获，分析截获的报文。

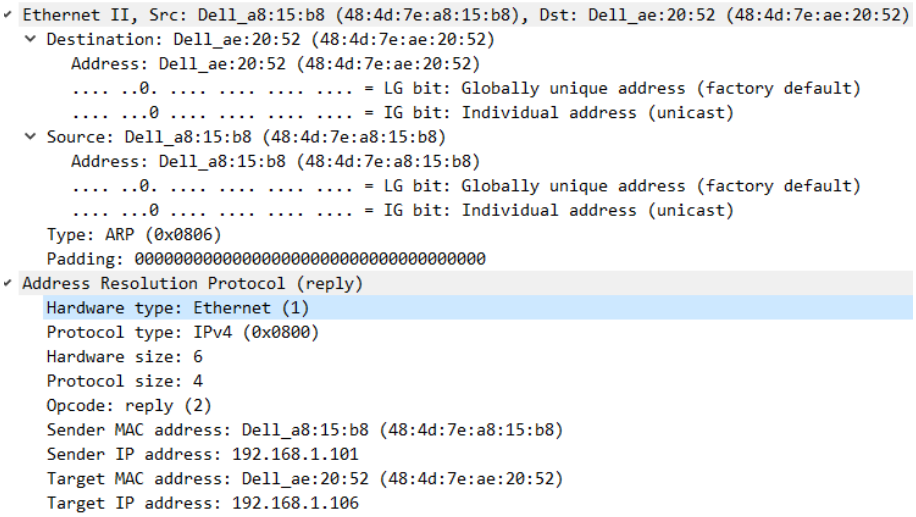
步骤 2：执行“arp –a”命令，记录结果。

步骤 3：分析捕获的报文，选中第一条 ARP 请求报文和第一条应答报文，填写表 2-1。

表 2-1 ARP 请求报文和应答报文的字段信息

字段	请求报文的值	应答报文的值
以太网链路层 Destination 项		
以太网链路层 Source 项		
ARP 报文发送者硬件地址		
ARP 报文发送者 IP		
ARP 报文目标硬件地址		
ARP 报文目标 IP		





步骤 4：比较 ARP 协议在不同网段和相同网段内解析过程的异同。

2.8 DNS 协议分析

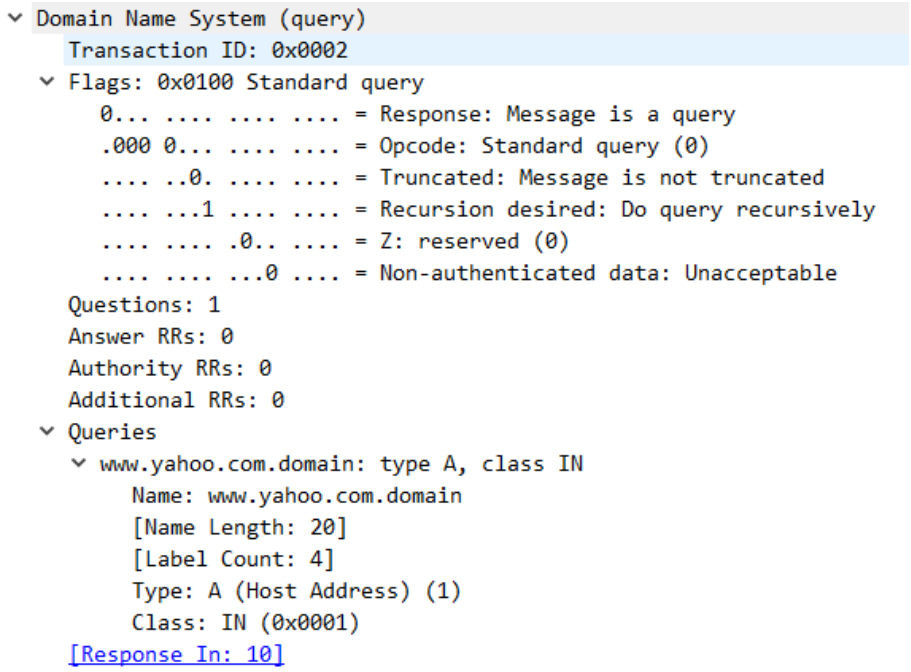
2.8.1默认 DNS 域名解析

步骤 1：在命令窗口执行命令：

执行“ipconfig /displaydns”观察本地缓存；

执行“ipconfig /flushdns”清除本地缓存。

步骤 2：在计算机终端上运行 Wireshark 截获报文，浏览器访问域名（如 http://www.yahoo.com），网站打开后停止报文截获，观察分析 DNS 查询、回复报文分别包含哪些主要内容（UDP 还是 TCP、目的地址与本机默认 DNS 是否相同、源端口和目的端口、域名类型、解析出的 IP 地址等）。



```
Domain Name System (response)
    Transaction ID: 0x0005
  Flags: 0x8180 Standard query response, No error
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... .0.. .... = Z: reserved (0)
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not
      .... .... ...0 .... = Non-authenticated data: Unacceptable
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 3
    Authority RRs: 4
    Additional RRs: 4
  Queries
    www.yahoo.com: type AAAA, class IN
        Name: www.yahoo.com
        [Name Length: 13]
        [Label Count: 3]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
  Answers
    www.yahoo.com: type CNAME, class IN, cname new-fp-shed.wg1.b.yahoo.com
        Name: www.yahoo.com
        Type: CNAME (Canonical NAME for an alias) (5)
        Class: IN (0x0001)
        Time to live: 32 (32 seconds)
        Data length: 20
        CNAME: new-fp-shed.wg1.b.yahoo.com
    new-fp-shed.wg1.b.yahoo.com: type AAAA, class IN, addr 2406:2000:ec:815::4
        Name: new-fp-shed.wg1.b.yahoo.com
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
        Time to live: 60 (1 minute)
        Data length: 16
        AAAA Address: 2406:2000:ec:815::4
    new-fp-shed.wg1.b.yahoo.com: type AAAA, class IN, addr 2406:2000:ec:815::3
        Name: new-fp-shed.wg1.b.yahoo.com
```

2.8.2 指定 DNS 域名解析

步骤 1：在命令窗口执行命令：

执行“ipconfig /displaydns”观察本地缓存；

执行“ipconfig /flushdns”清除本地缓存。

步骤 2：在计算机终端上运行 Wireshark 截获报文，在命令窗口执行指定 DNS 服务器解析域名命令（如 nslookup www.github.com 223.6.6.6），解析完毕后停止报文截获，观察分析 DNS 查询、回复报文分别包含哪些主要内容（UDP 还是 TCP、目的地址与本机默认 DNS 是否相同、源端口和目的端口、域名类型、解析出的 IP 地址等）。

2.9 互动讨论主题

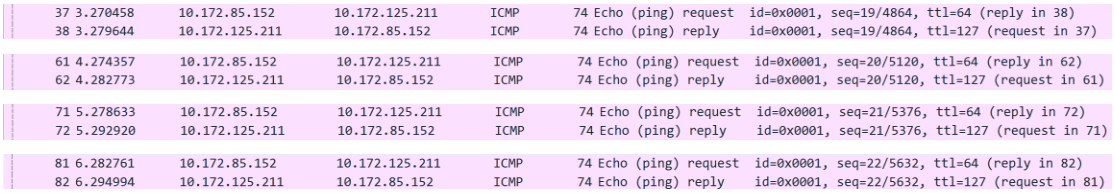
（1）发送方与接收方 ARP 与 ICMP 报文出现的次序成因；
（2）ARP 的安全性问题；
（3）DNS 的欺骗带来的安全性问题；

2.10 进阶自设计

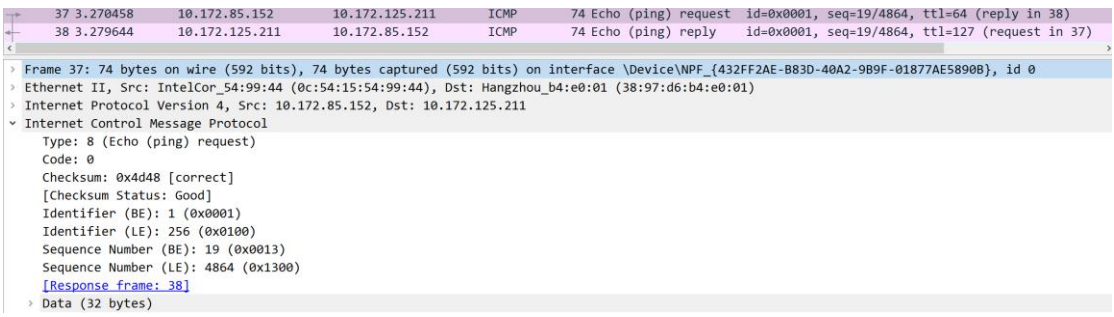
Scapy 是一个 Python 程序，它允许用户发送、嗅探、分析和伪造网络包。这种能力允许构建能够探测、扫描或攻击网络的工具。换句话说，Scapy 是一个强大的交互式包操作程序。它能够伪造或解码大量协议的数据包，在网络上发送它们，捕获它们，匹配请求和响应，等等。Scapy 可以轻松地处理大多数经典任务，如扫描、跟踪、探测、单元测试、攻击或网络发现。它可以代替 hping、arpsoof、arp-sk、arping、p0f 甚至 Nmap、tcpdump 和 tshark 的某些部分。

(1) 使用 scapy 在 Linux 下写程序来模拟完成一个简单的 ARP 欺骗。
(2) 使用 scapy 在 Linux 下写程序来模拟完成一个简单的 DNS 欺骗。完整的攻击实现工作量和难度都很大。为了降低难度，可以不实现中间人攻击，而是直接让受害者把 DNS 服务器修改为欺骗者的地址。

初步检查，可以发现一共出现了 8 个相关数据包，其中 ping 请求包和响应包各 4 个。



仔细分析其中一个请求包：



可以看出，这是 ping 命令发送的一个 ICMP 协议包。ICMP 协议是 TCP/IP 协议的一个子协议，用于在 IP 主机、路由器之间传递控制消息。该数据包的源 ip 地址和目的 ip 地址与我们的主机相吻合，它的类型号为 8，代表一个 ping 请求命令。

```
    Protocol: ICMP (1)
    Header Checksum: 0xf239 [validation disabled]
```

```
0000  38 97 d6 b4 e0 01 0c 54  15 54 99 44 08 00 45 00   8·····T ·T·D··E·
0010  00 3c 9f c4 00 00 40 01  f2 39 0a ac 55 98 0a ac   ·<····@· ·9··U···
0020  7d d3 08 00 4d 48 00 01  00 13 61 62 63 64 65 66   }···MH·· ··abcdef
0030  67 68 69 6a 6b 6c 6d 6e  6f 70 71 72 73 74 75 76   ghijklmn opqrstuv
0040  77 61 62 63 64 65 66 67  68 69                    wabcdefg hi
```

这里的 01 也代表着该 ip 协议封装着一个 ICMP 协议数据包。

再来看一个响应包：

```
    37 3.270458    10.172.85.152    10.172.125.211   ICMP    74 Echo (ping) request  id=0x0001, seq=19/4864, ttl=64 (reply in 38)
    38 3.279644    10.172.125.211   10.172.85.152    ICMP    74 Echo (ping) reply    id=0x0001, seq=19/4864, ttl=127 (request in 37)

> Frame 38: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{432FF2AE-B83D-40A2-9B9F-01877AE5890B}, id 0
> Ethernet II, Src: Hangzhou_b4:e0:01 (38:97:d6:b4:e0:01), Dst: IntelCor_54:99:44 (0c:54:15:54:99:44)
> Internet Protocol Version 4, Src: 10.172.125.211, Dst: 10.172.85.152
v Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
    Checksum: 0x5548 [correct]
    [Checksum Status: Good]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence Number (BE): 19 (0x0013)
    Sequence Number (LE): 4864 (0x1300)
    [Request frame: 37]
    [Response time: 9.186 ms]
  > Data (32 bytes)
```

可以看出，响应包和请求包的结构基本相同，唯一不同的类型值为 0，代表一个 ping 响应包。同时通过 wireshark 也能直接看出响应时间为 9.186ms。

在这两个数据包中，下面四个属性值得我们注意。可以看出，对应值的 BE 和 LE 的十六进制格式是前后颠倒的。通过查阅资料我们得知，这是因为 wireshark 考虑到 window 系统与 Linux 系统发出的 ping 报文的字节顺序不一样，windows 为 LE：小端编码，Linux 为 BE：大端编码，因此它将同一数据按两种不同方式进行了表示。在实际的报文中，Identifier 就是 0x0001 这一个值，只不过为了方便软件自动将他表示成了两种不同形式。

```
Identifier (BE): 1 (0x0001)
Identifier (LE): 256 (0x0100)
Sequence Number (BE): 19 (0x0013)
Sequence Number (LE): 4864 (0x1300)
```

这里还要说一下，ICMP 协议其实是用 Type 字段和 Code 字段共同表示报文类型的，ping 请求 Type 是 8，Code 是 0；ping 响应 Type 是 0，Code 是 0。