南 开 大 学

计算机与网络空间安全学院

网络技术与应用课程报告

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**第三次实验报告**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

学号：2012911

姓名：马永田

年级：2020级

专业：计算机科学与技术

2022年11月8日

1. **实验内容说明**

**（一）通过编程获取IP地址与MAC地址的对应关系，要求如下：**

1. 在IP数据报捕获与分析编程实验的基础上，学习WinPcap的数据包发送方法。
2. 通过Npcap编程，获取IP地址与MAC地址的映射关系。
3. 程序要具有输入IP地址，显示输入IP地址与获取的MAC地址对应关系界面。界面可以是命令行界面，也可以是图形界面，但应以简单明了的方式在屏幕上显示。
4. 编写的程序应结构清晰，具有较好的可读性。
5. **实验准备**
6. **学习WinPcap的数据包发送方法**

使用**pcap\_sendpacket()**来发送一个数据包

函数声明：int pcap\_sendpacket(pcap\_t \*p, const u\_char \*buf, int size);

函数的三个参数：

• 一个装有要发送数据的缓冲区；

• 要发送的长度；

• 一个适配器；

注意缓冲区中的数据将不被内核协议处理，只是作为最原始的数据流被发送，所以我们必须填充好正确的协议头以便正确的将数据发送

1. **配置实验环境并构建项目**

安装 Npcap1.71、npcap-sdk-1.13,并对Visual Studio的项目进行如下配置：

• 在 C/C++ 中添加附加包含目录：D:\npcap\Include;

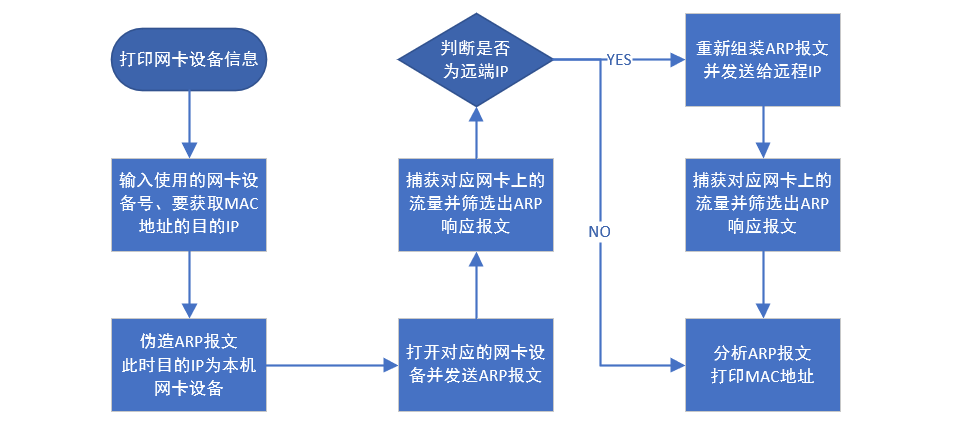
• 在预处理器中添加预处理器定义：WPCAP;HAVE\_REMOTE;

• 在链接器-> 常规中添加附加库目录：D:\npcap\Lib;

• 在链接器-> 输入中添加附加依赖项：Packet.lib;wpcap.lib

1. **实验过程**
2. **项目设计思路**

可以利用ARP请求方法获取对应IP的MAC地址，本次实验的流程如下图所示：



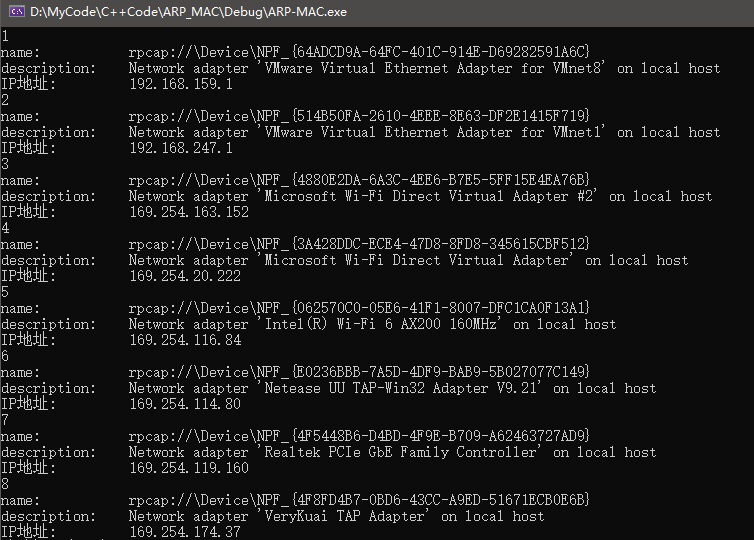
1. 首先获取网络接口卡列表，并将信息打印
2. 选择相应的网卡设备，输入需要获取MAC地址的IP地址
3. 伪造ARP请求报文，内容要求如下：
   1. MAC地址设置为全1 广播
   2. 伪造源MAC地址和源IP地址
   3. 目的IP地址为对应网卡的IP地址
4. 打开对应网卡并发送伪造的ARP报文
5. 对该网卡进行流量监听，筛选其中的ARP报文（类型为0x806），捕获网卡A的ARP响应报文
6. 判断目的IP是否为远端IP地址：
   1. 若为远端IP地址，重新组装ARP包，将本机网卡的MAC地址填入对应位置，目的IP改为远端IP地址，之后重新发送和捕获ARP报文。
   2. 若为本机网卡的IP地址，直接进入打印阶段。
7. 根据捕获的ARP报文打印MAC地址：在响应报文的帧首部源MAC地址部分可以看到发送该ARP响应的网卡对应的MAC地址。
8. **关键代码分析**
9. **声明定义要用到的结构体和全局变量以及函数**

|  |
| --- |
| #pragma pack (1)//进入字节对齐方式  //以太网帧 14字节  typedef struct FrameHeader\_t {  BYTE DesMAC[6];// 目的地址  BYTE SrcMAC[6];//源地址  WORD FrameType;//帧类型  }FrameHeader\_t;  //ARP帧 28字节  typedef struct ARPFrame\_t {  FrameHeader\_t FrameHeader;//以太网帧头  WORD HardwareType;//硬件类型  WORD ProtocolType;//协议类型  BYTE HLen;//硬件地址长度  BYTE PLen;//协议地址长度  WORD Operation;  BYTE SendHa[6]; //发送端以太网地址  DWORD SendIP; //发送端IP地址  BYTE RecvHa[6]; //目的以太网地址  DWORD RecvIP; //目的IP地址  } ARPFrame\_t;  #pragma pack ()  //全局  pcap\_t\* adhandle; //捕捉实例,是pcap\_open返回的对象  string IPList[100]; //存储网卡设备IP地址  string DesIP; //要获取MAC地址的目的IP  int dev\_nums = 0; //适配器计数变量  bool Remote = 0; //是否为远端IP地址  BYTE MyMAC[6]; //本机设备MAC地址  //判断两个MAC地址是否相同  bool CompareMAC(BYTE\* MAC\_1, BYTE\* MAC\_2) {  for (int i = 0; i < 6; i++) {  if (MAC\_1[i] != MAC\_2[i]) {  return false;  }  }  return true;  }  //拷贝MAC地址  void CopyMAC(BYTE\* MAC\_1, BYTE\* MAC\_2) {  for (int i = 0; i < 6; i++) {  MAC\_2[i] = MAC\_1[i];  }  }  //打印ARP数据包的源MAC地址  void PrintPacketMAC(ARPFrame\_t\* RecPacket) {  printf("%s:\t%02x-%02x-%02x-%02x-%02x-%02x\n", "目标MAC地址",  RecPacket->FrameHeader.SrcMAC[0],  RecPacket->FrameHeader.SrcMAC[1],  RecPacket->FrameHeader.SrcMAC[2],  RecPacket->FrameHeader.SrcMAC[3],  RecPacket->FrameHeader.SrcMAC[4],  RecPacket->FrameHeader.SrcMAC[5]);  return;  } |

1. **获取设备列表并打印设备信息**

|  |
| --- |
| //遍历接口列表  void DevsList(pcap\_if\_t\* alldevs) {  for (pcap\_if\_t\* d = alldevs; d != nullptr; d = d->next)//显示接口列表  {  //获取该网络接口设备的ip地址信息  for (pcap\_addr\* a = d->addresses; a != nullptr; a = a->next)  {  if (((struct sockaddr\_in\*)a->addr)->sin\_family == AF\_INET && a->addr)  {//打印ip地址  dev\_nums++;  //打印相关信息  //inet\_ntoa将ip地址转成字符串格式  printf("%d\n", dev\_nums);  printf("%s\t\t%s\n%s\t%s\n", "name:", d->name, "description:", d->description);  printf("%s\t\t%s\n", "IP地址:", inet\_ntoa(((struct sockaddr\_in\*)a->addr)->sin\_addr));  IPList[dev\_nums] = inet\_ntoa(((struct sockaddr\_in\*)a->addr)->sin\_addr);  }  }  }  } |

运行结果如下图所示：



1. **伪造ARP报文**

|  |
| --- |
| //伪造ARP包  ARPFrame\_t MakeARP(pcap\_addr\* a) {  ARPFrame\_t ARPFrame;  for (int i = 0; i < 6; i++)  ARPFrame.FrameHeader.DesMAC[i] = 0xff;//表示广播  //将APRFrame.FrameHeader.SrcMAC设置为本机网卡的MAC地址 最开始未知 伪造  for (int i = 0; i < 6; i++)  ARPFrame.FrameHeader.SrcMAC[i] = 0x0f;  CopyMAC(ARPFrame.FrameHeader.SrcMAC, MyMAC);  ARPFrame.FrameHeader.FrameType = htons(0x806);//帧类型为ARP  ARPFrame.HardwareType = htons(0x0001);//硬件类型为以太网  ARPFrame.ProtocolType = htons(0x0800);//协议类型为IP  ARPFrame.HLen = 6;//硬件地址长度为6  ARPFrame.PLen = 4;//协议地址长为4  ARPFrame.Operation = htons(0x0001);//操作为ARP请求  //将ARPFrame.SendHa设置为本机网卡的MAC地址  for (int i = 0; i < 6; i++)  ARPFrame.SendHa[i] = 0x0f;  //将ARPFrame.SendIP设置为本机网卡上绑定的IP地址  ARPFrame.SendIP = inet\_addr("122.122.122.122");  //将ARPFrame.RecvHa设置为0  for (int i = 0; i < 6; i++)  ARPFrame.RecvHa[i] = 0;//表示目的地址未知  //将ARPFrame.RecvIP设置为请求的IP地址 最开始为本机网卡地址  ARPFrame.RecvIP = inet\_addr(inet\_ntoa(((struct sockaddr\_in\*)a->addr)->sin\_addr));  return ARPFrame;  } |

1. **打开设备并发送报文**

|  |
| --- |
| //打开该网卡的网络接口  adhandle = pcap\_open(d->name, 655340, PCAP\_OPENFLAG\_PROMISCUOUS, 1000, 0, 0);  if (adhandle == NULL) { cout << "打开接口失败"; return; }  //发包  if (pcap\_sendpacket(adhandle, (u\_char\*)&ARPFrame, sizeof(ARPFrame\_t))== 0){ break; }; |

1. **捕获报文**

将伪造的ARP数据包广播之后，本机的网卡会发出一个响应ARP数据包，其中会包含网卡设备的MAC地址，捕获该响应ARP数据包。

|  |
| --- |
| //收包  ARPFrame\_t\* Recv(pcap\_t\* adhandle) {  struct pcap\_pkthdr\* pkt\_header;  const u\_char\* pkt\_data;  int res;  while ((res = pcap\_next\_ex(adhandle, &pkt\_header, &pkt\_data)) >= 0) {  ARPFrame\_t\* RecPacket = (ARPFrame\_t\*)pkt\_data;  //PrintPacketMAC(RecPacket);  if (  \*(unsigned short\*)(pkt\_data + 12) == htons(0x0806) //0x0806为以太网帧类型表示后面数据的类型，对于ARP请求或应答来说，该字段的值为x0806  && \*(unsigned short\*)(pkt\_data + 20) == htons(2) //ARP应答  &&!CompareMAC(RecPacket->FrameHeader.SrcMAC, MyMAC) //若不为最初发的ARP包  )  {  return RecPacket;  }  }  } |

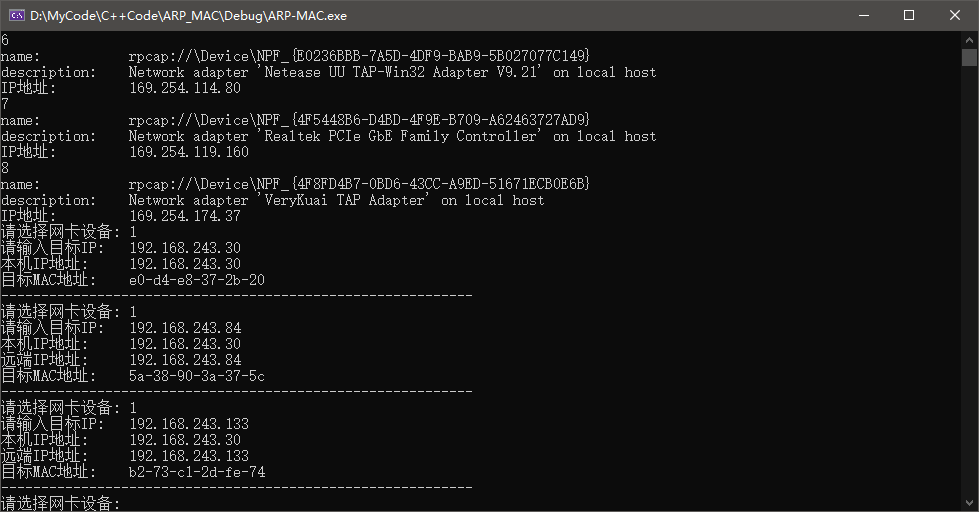
1. **获取远程设备的MAC地址**

对于远程IP的MAC地址获取，可以先利用前文中部分请求本地网卡的MAC地址，之后将将本机IP和MAC填入报文中，即封装ARP请求时使用本机网卡的IP和MAC地址，最后再重新发送ARP请求，这样本机网卡就会受到目的IP设备发送的带有MAC地址的响应ARP报文。

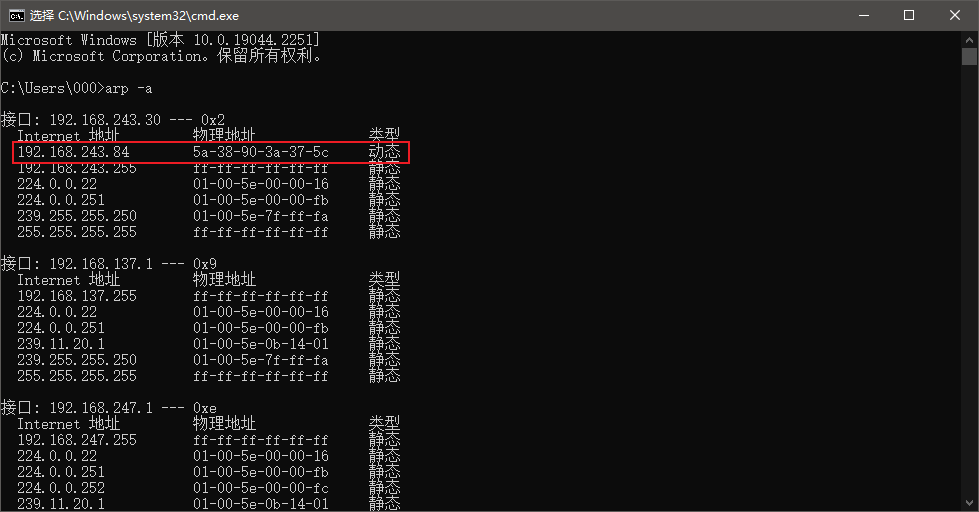
|  |
| --- |
| //若为远端IP  if (Remote) {  printf("%s\t%s\n", "远端IP地址:",DesIP.c\_str());  //更新伪造的ARP包 将本机的MAC填入  CopyMAC(MyMAC, ARPFrame.FrameHeader.SrcMAC);  CopyMAC(MyMAC, ARPFrame.SendHa);  ARPFrame.RecvIP = inet\_addr(DesIP.c\_str());  //重新发包  if (Send(adhandle, ARPFrame) == 0) { break; };  //收包  RecPacket = Recv(adhandle);  } |

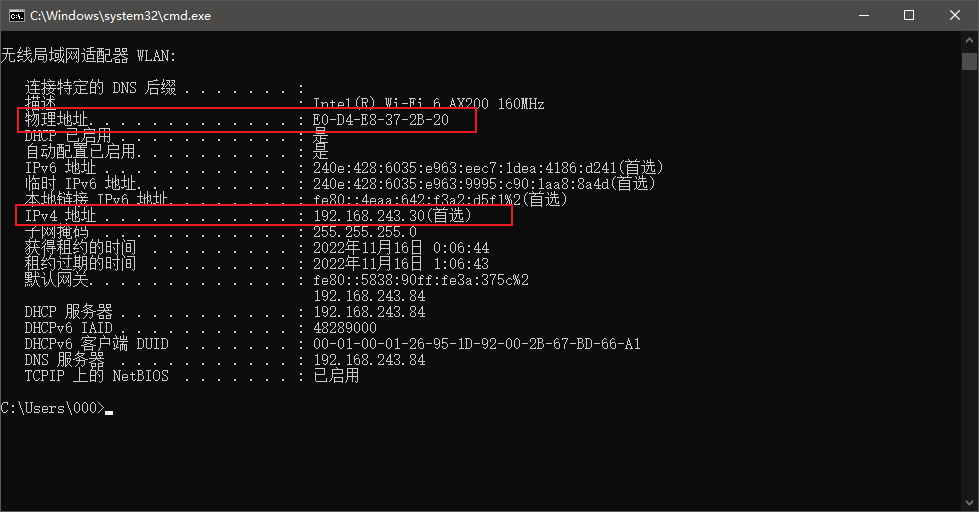
1. **程序运行结果**

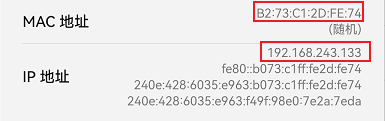
程序运行结果如下图可见，成功的获取到了三个目的IP对应的MAC地址：



查看设备的网络配置与路由表等对结果进行验证，可以看到获取的MAC地址均是正确的。







1. **特殊现象分析**

|  |
| --- |
| if (pcap\_sendpacket(adhandle, sendbuf, 42) == 0) {  printf("PacketSend succeed\n");  }  else {  printf("PacketSend failed Error: %d\n", GetLastError());  return 0;  } |

如果adhandle是“Network adapter 'Intel(R) Wi-Fi 6 AX201 160MHz' on local host”这种接口，pcap\_sendpacket会返回-1，表示发包遇到了问题。GetLastError会返回错误码31，含义是“连到系统上的设备没有发挥作用”，但用其他的接口不会有问题。出于调试方便，往往采用返回值判断来看是否执行成功，并在错误时直接打印错误信息并退出，但实际上数据包是正发出的。