## 网络技术与应用第三次实验报告

网络空间安全学院 物联网工程 2111673 岳志鑫

**一、实验要求**

通过编程获取IP地址与MAC地址的对应关系实验，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NPcap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。

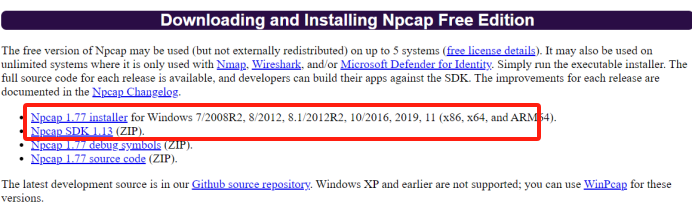
（3）程序要具有输入IP地址，显示输入IP地址与获取的MAC地址对应关系界面。界面可以是命令行界面，也可以是图形界面，但应以简单明了的方式在屏幕上显示。

（4）编写的程序应结构清晰，具有较好的可读性。

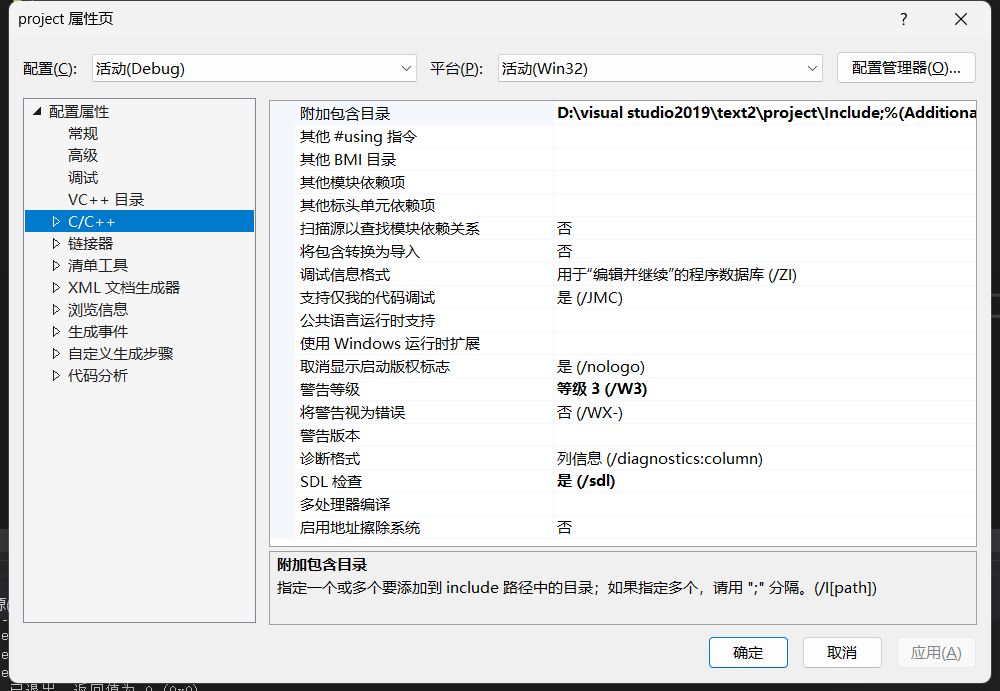
**二、实验内容**

**1.环境配置**

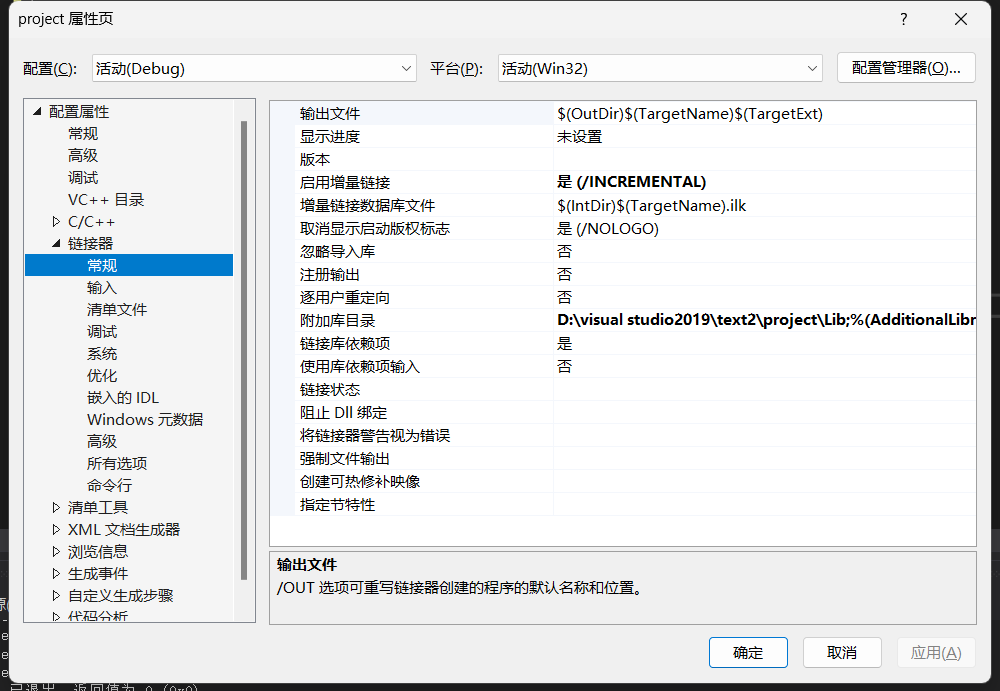
在官网（https://npcap.com/#download）下载Npcap和Npcap-SDK



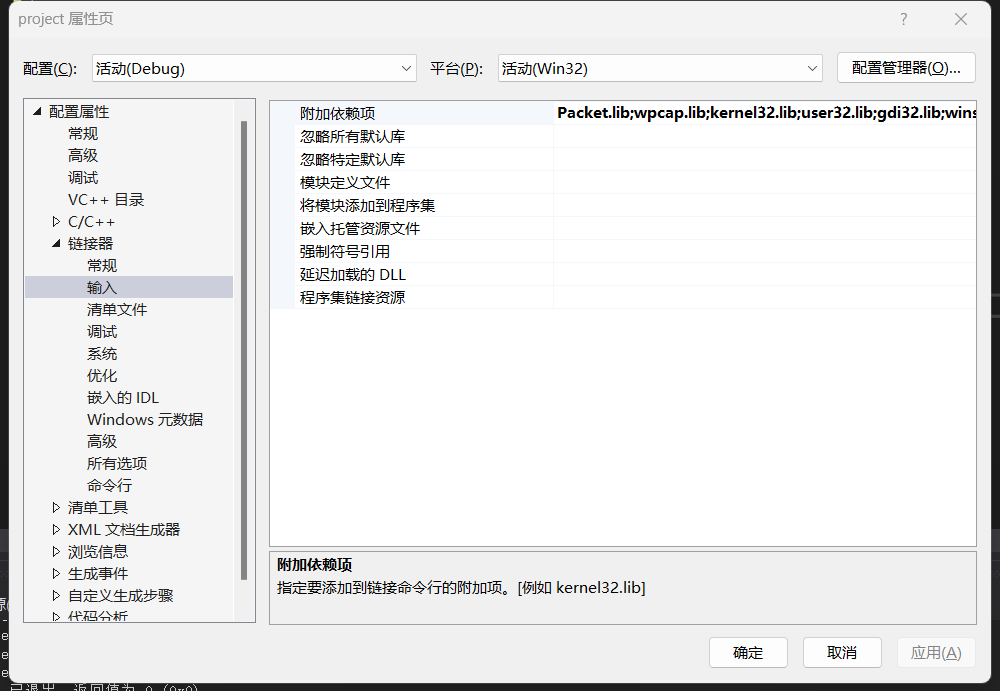
项目→属性→C/C++→常规→附加包含目录：添加sdk中的Include目录



项目→属性→链接器→常规→附加库目录：添加sdk中的Lib目录



项目→属性→链接器→输入→附加依赖项：添加Packet.lib;wpcap.lib;



输入官网的测试代码，若成功检测则环境配置成功

#include "pcap.h"

void main()

{

    pcap\_if\_t \*alldevs;

    pcap\_if\_t \*d;

    int i = 0;

    char errbuf[PCAP\_ERRBUF\_SIZE];

    /\* Retrieve the device list from the local machine \*/

    if (pcap\_findalldevs\_ex(PCAP\_SRC\_IF\_STRING, NULL /\* auth is not needed \*/, &alldevs, errbuf) == -1)

    {

        fprintf(stderr, "Error in pcap\_findalldevs\_ex: %s\n", errbuf);

        exit(1);

    }

    /\* Print the list \*/

    for (d = alldevs; d != NULL; d = d->next)

    {

        printf("%d. %s", ++i, d->name);

        if (d->description)

            printf(" (%s)\n", d->description);

        else

            printf(" (No description available)\n");

    }

    if (i == 0)

    {

        printf("\nNo interfaces found! Make sure Npcap is installed.\n");

        return;

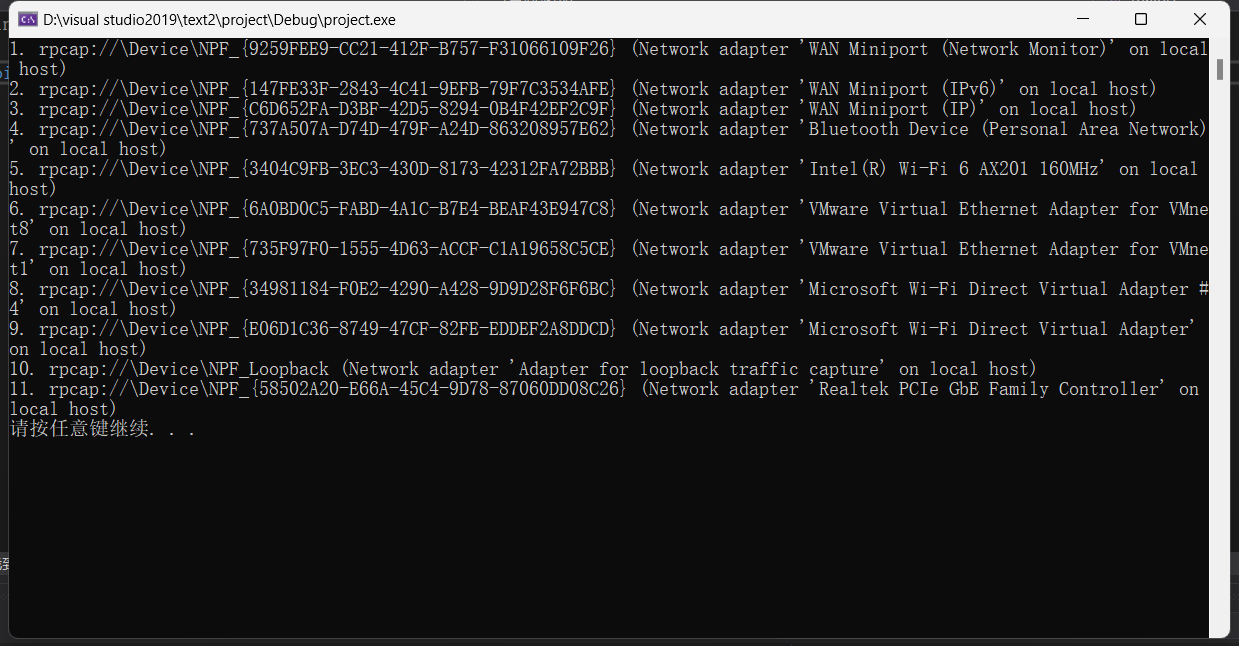
    }

    /\* We don't need any more the device list. Free it \*/

    pcap\_freealldevs(alldevs);

    system("pause");

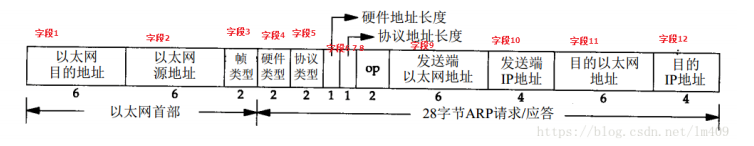
}



1. **ARP数据包**

地址解析协议（Address Resolution Protocol），其基本功能为透过目标设备的IP地址，查询目标设备的MAC地址，以保证通信的顺利进行。它是IPv4中网络层必不可少的协议，不过在IPv6中已不再适用，并被邻居发现协议（NDP）所替代。

ARP报文格式:



(1). 字段1是ARP请求的目的以太网地址，全1时代表广播地址。

(2). 字段2是发送ARP请求的以太网地址。

(3). 字段3以太网帧类型表示的是后面的数据类型，ARP请求和ARP应答这个值为0x0806。

(4). 字段4表示硬件地址的类型，硬件地址不只以太网一种，是以太网类型时此值为1。

(5). 字段5表示要映射的协议地址的类型，要对IP地址进行映射，此值为0x0800。

(6). 字段6和7表示硬件地址长度和协议地址长度，MAC地址占6字节，IP地址占4字节。

(7). 字段8是操作类型字段，值为1，表示进行ARP请求；值为2，表示进行ARP应答；值为3，表示进行 RARP请求；值为4，表示进行RARP应答。

(8). 字段9是发送端ARP请求或应答的硬件地址，这里是以太网地址，和字段2相同。

(9). 字段10是发送ARP请求或应答的IP地址。

(10). 字段11和12是目的端的硬件地址和协议地址

1. **主要函数分析**
2. .输出设备和描述信息

利用pcap包的指针和pcap\_findalldevs函数可以查找当前所有网卡设备，并输出查找到的所有设备信息

//打印设备信息

     //打印设备列表中设备信息

    pcap\_if\_t\* count; //遍历用的指针

    char srcip[INET\_ADDRSTRLEN];//本机ip

    //输出设备名和描述信息

    for (count = devices; count; count = count->next)//借助count指针从第一个设备开始访问到最后一个设备

    {

        cout << ++i << ". " << count->name;//输出设备信息和描述

        if (count->description) {

            cout << "描述：(" << count->description << ")" << endl;

        }

        for (a = count->addresses; a != NULL; a = a->next) {

            if (a->addr->sa\_family == AF\_INET) {

                char str[INET\_ADDRSTRLEN];

                inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)a->addr), str, sizeof(str));//将 a->addr 强制转换为 struct sockaddr\_in 类型的指针，并访问 sin\_addr 成员，其中包含了 IPv4 地址。

                cout << "IP地址：" << str << endl;

                inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)a->netmask), str, sizeof(str)); //将 a->netmask 强制转换为 struct sockaddr\_in 类型的指针，从a->netmask这个结构中提取子网掩码。

                cout << "子网掩码：" << str << endl;

                inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)a->broadaddr), str, sizeof(str));//将 a->netmask 强制转换为 struct sockaddr\_in 类型的指针，从a->broadaddr这个结构中提取广播地址。

                cout << "广播地址：" << str << endl;

            }

        }

    }

    //设备数量为0

    if (i == 0) {

        cout << endl << "存在错误！无查找设备！" << endl;

        return 0;

    }

1. .组装报文，利用ARP\_Frame结构体定义ARP数据报内部的所有信息，例如目标IP地址和源IP地址等信息

//组装报文

    unsigned char mac[48];

    for (int i = 0; i < 6; i++) {

        send\_ARPFrame.FrameHeader.DesMAC[i] = 0xFF; //DesMAC设置为广播地址

        send\_ARPFrame.DesMAC[i] = 0x00; //DesMAC设置为0

        //SrcMAC用不到可以不设置

    }

    send\_ARPFrame.FrameHeader.FrameType = htons(0x0806); //帧类型为ARP

    send\_ARPFrame.HardwareType = htons(0x0001); //硬件类型为以太网

    send\_ARPFrame.ProtocolType = htons(0x0800); //协议类型为IP

    send\_ARPFrame.HLen = 6; //硬件地址长度为6

    send\_ARPFrame.PLen = 4; //协议地址长度为4

    send\_ARPFrame.op = htons(0x0001); //操作为ARP请求

    send\_ARPFrame.DesIP = inet\_addr(srcip); //设置为本机IP地址

其中ARP数据结构体和数据帧头部定义如下

typedef struct Frame\_Header//帧首部

{

    BYTE DesMAC[6];  //目的地址

    BYTE SrcMAC[6];  //源地址

    WORD FrameType;  //帧类型

};

typedef struct ARP\_Frame//ARP数据

{

    Frame\_Header FrameHeader;

    WORD HardwareType; //硬件类型

    WORD ProtocolType; //协议类型

    BYTE HLen; //硬件长度

    BYTE PLen; //协议长度

    WORD op; //操作类型

    BYTE SrcMAC[6]; //源MAC地址

    DWORD SrcIP; //源IP地址

    BYTE DesMAC[6]; //目的MAC地址

    DWORD DesIP; //目的IP地址

};

1. 发送构造的数据包后需要对反馈的数据包进行抓捕并拆包分析，参考ARP数据包的结构按字节进行拆分

if (\*(unsigned short\*)(packetData + 12) == htons(0x0806) //帧类型为ARP（htons(0x0806)）

            && \*(unsigned short\*)(packetData + 20) == htons(0x0002)) //操作类型为ARP响应（htons(0x0002)）

        {

            cout << endl;

            cout << "-----------------------------------------------" << endl;

            cout << "ARP数据包内容：" << endl;

            //打印数据包

            cout << "源IP地址:\t ";

// 提取ip地址（28-32字节）

            for (int i = 28; i < 32; ++i) {

                printf("%d", packetData[i]);

                if (i < 31) cout << ".";

            }

            cout << endl;

            // 提取MAC地址（6-12字节）

            cout << "源MAC地址:\t ";

            for (int i = 6; i < 12; ++i) {

                printf("%02X", packetData[i]);

                if (i < 11) cout << "-";

            }

            cout << endl;

            //用mac数组记录本机的MAC地址

            for (int i = 0; i < 6; i++)

            {

                mac[i] = \*(unsigned char\*)(packetData + 22 + i);

            }

            cout << "获取MAC地址成功，MAC地址为：";

            for (int i = 6; i < 12; ++i) {

                printf("%02X", packetData[i]);

                if (i < 11) cout << "-";

            }

            cout << endl;

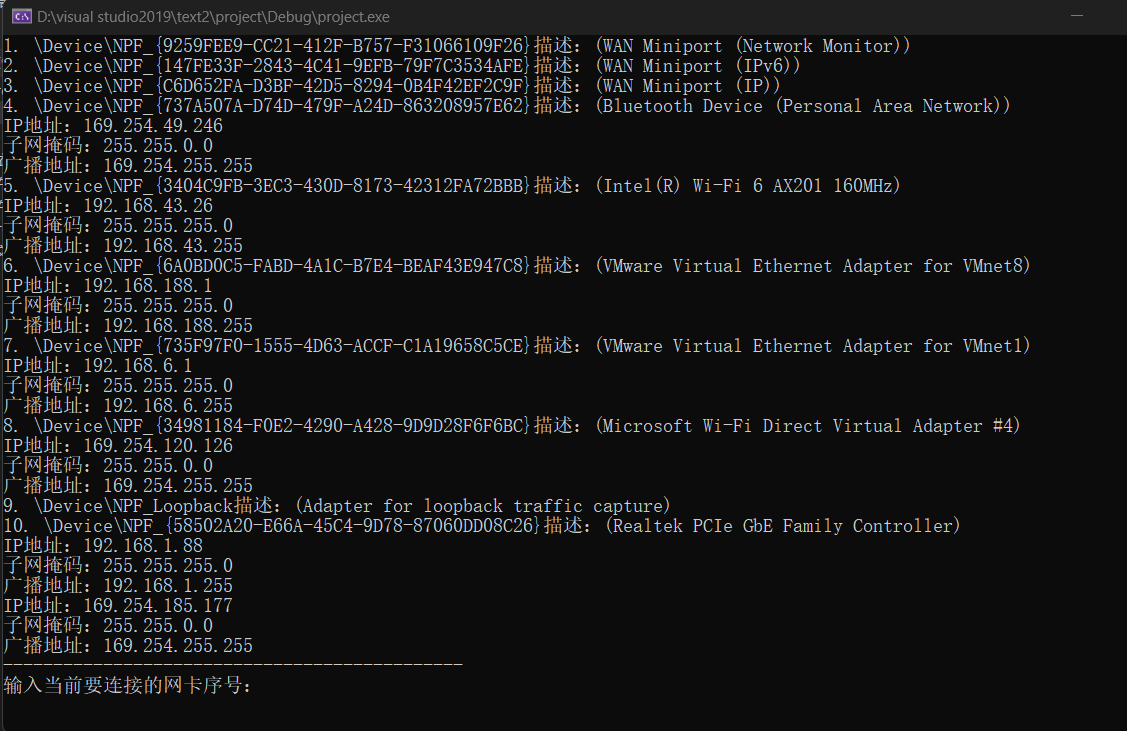
            cout << "----------------------------------------------" << endl;

            break;

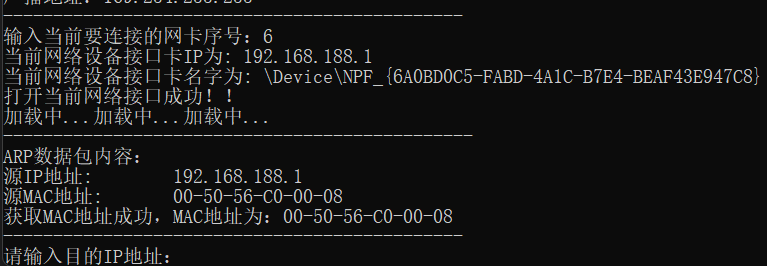
        }

1. **实验结果**

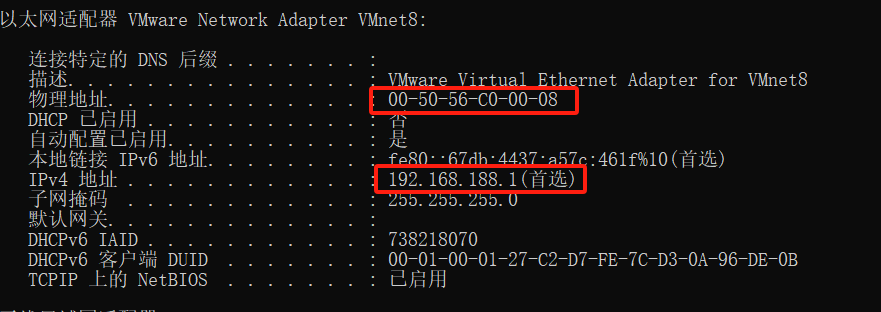
1.打印设备信息如图所示：将所有通信网卡设备的名称及IP地址等相关信息打印在屏幕上



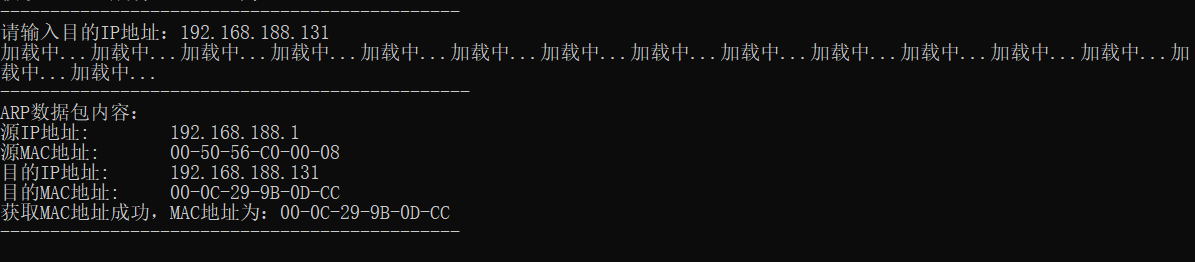
1. 输入网卡的序号，这里输入vmware虚拟机的网卡序号6，显示相对应网卡的信息以及IP地址和Mac地址



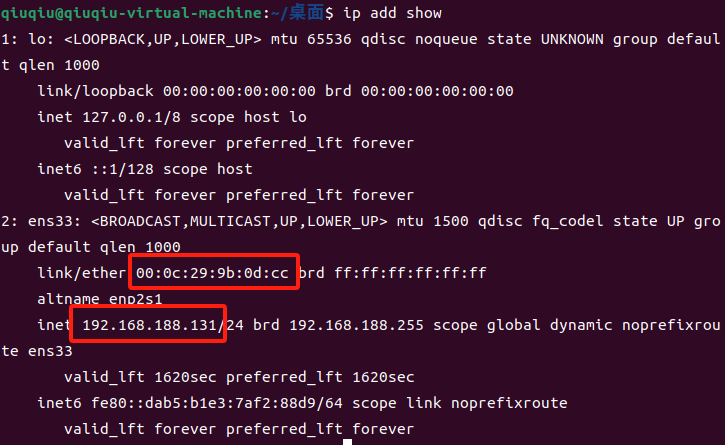
用windows的命令行ipconfig/all查询自己的IP地址和mac地址发现与抓捕到的mac地址相同



1. 输入目的IP地址为192.168.188.131，这是我在虚拟机中搭建的网站服务器的IP地址，可以用来检测是否能正确获取到mac地址



从虚拟机中查询mac地址如下，发现与抓捕到的mac地址也相同，实验成功



**四、实验总结**

1.对NPcap的架构有了更深入的了解与认识，学习到了NPcap的设备列表获取方法、网卡设备打开方法，以及数据包构造方法、数据包捕获方法，增强了动手实践的能力。

2.通过此次实验也发现一些问题，比如如果不设置#pragma pack(1)指定结构体按照一个字节对齐，那么抓捕数据包会出现速度上的问题，较为浪费时间。

3.对于ARP数据包的结构是需要重点记忆的，否则在拆分数据包的时候就会恍惚到底拆分的是哪些数据，因此还需要在未来继续学习，继续实践。

**五、实验源码**

#define WIN32

#define HAVE\_REMOTE

#include "pcap.h"

#include <iostream>

#include <WinSock2.h>

using namespace std;

#pragma comment(lib,"ws2\_32.lib")

#pragma warning(disable:4996)

#pragma pack(1)

typedef struct Frame\_Header//帧首部

{

    BYTE DesMAC[6];  //目的地址

    BYTE SrcMAC[6];  //源地址

    WORD FrameType;  //帧类型

};

typedef struct ARP\_Frame//ARP数据

{

    Frame\_Header FrameHeader;

    WORD HardwareType; //硬件类型

    WORD ProtocolType; //协议类型

    BYTE HLen; //硬件长度

    BYTE PLen; //协议长度

    WORD op; //操作类型

    BYTE SrcMAC[6]; //源MAC地址

    DWORD SrcIP; //源IP地址

    BYTE DesMAC[6]; //目的MAC地址

    DWORD DesIP; //目的IP地址

};

void\* getaddress(struct sockaddr\* sa)//得到对应的IP地址

{

    if (sa->sa\_family == AF\_INET)

    {

        return &(((struct sockaddr\_in\*)sa)->sin\_addr);//IPV4地址

    }

    return &(((struct sockaddr\_in6\*)sa)->sin6\_addr);//IPV6地址

}

int main() {

    /\*获取设备列表，打印信息\*/

    pcap\_if\_t\* d; //遍历用的指针

    pcap\_addr\_t\* a; //地址指针

    pcap\_if\_t\* devices; //指向设备列表第一个

    int i = 0; //统计设备数量

    char errbuf[PCAP\_ERRBUF\_SIZE]; //错误信息缓冲区

    //输出错误信息

    if (pcap\_findalldevs(&devices, errbuf) == -1)

    {

        cout << stderr << "查找设备失败: " << errbuf << endl;

        return 0;

    }

    //打印设备信息

     //打印设备列表中设备信息

    pcap\_if\_t\* count; //遍历用的指针

    char srcip[INET\_ADDRSTRLEN];//本机ip

    //输出设备名和描述信息

    for (count = devices; count; count = count->next)//借助count指针从第一个设备开始访问到最后一个设备

    {

        cout << ++i << ". " << count->name;//输出设备信息和描述

        if (count->description) {

            cout << "描述：(" << count->description << ")" << endl;

        }

        for (a = count->addresses; a != NULL; a = a->next) {

            if (a->addr->sa\_family == AF\_INET) {

                char str[INET\_ADDRSTRLEN];

                inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)a->addr), str, sizeof(str));//将 a->addr 强制转换为 struct sockaddr\_in 类型的指针，并访问 sin\_addr 成员，其中包含了 IPv4 地址。

                cout << "IP地址：" << str << endl;

                inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)a->netmask), str, sizeof(str)); //将 a->netmask 强制转换为 struct sockaddr\_in 类型的指针，从a->netmask这个结构中提取子网掩码。

                cout << "子网掩码：" << str << endl;

                inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)a->broadaddr), str, sizeof(str));//将 a->netmask 强制转换为 struct sockaddr\_in 类型的指针，从a->broadaddr这个结构中提取广播地址。

                cout << "广播地址：" << str << endl;

            }

        }

    }

    //设备数量为0

    if (i == 0) {

        cout << endl << "存在错误！无查找设备！" << endl;

        return 0;

    }

    cout << "----------------------------------------------" << endl;

    /\*选择设备及打开网卡\*/

    pcap\_if\_t\* count2; //遍历用的指针2

    int num = 0;

    cout << "输入当前要连接的网卡序号：";

    cin >> num;

    while (num < 1 || num>11) {

        cout << "请检查网卡序号输入是否正确！" << endl;

        cout << "重新输入当前要连接的网卡序号：";

        cin >> num;

    }

    count2 = devices;

    for (int i = 1; i < num; i++) {//循环遍历指针选择第几个网卡

        count2 = count2->next;

    }

    inet\_ntop(AF\_INET, getaddress((struct sockaddr\*)count2->addresses->addr), srcip, sizeof(srcip));

    //将 a->addr 强制转换为 struct sockaddr\_in 类型的指针，并访问 sin\_addr 成员，其中包含了 IPv4 地址。

    cout << "当前网络设备接口卡IP为: " << srcip << endl << "当前网络设备接口卡名字为: " << count2->name << endl;

    //打开网络接口

    //指定获取数据包最大长度为65536,可以确保程序可以抓到整个数据包

    //指定时间范围为200ms

    pcap\_t\* point = pcap\_open(count2->name, 65536, PCAP\_OPENFLAG\_PROMISCUOUS, 200, NULL, errbuf);

    if (point == NULL) {

        cout << "打开当前网络接口失败" << endl;  //打开当前网络接口失败

        pcap\_freealldevs(devices);

        return 0;

    }

    else {

        cout << "打开当前网络接口成功！！" << endl;

    }

    ARP\_Frame send\_ARPFrame;

    //获取本机的MAC地址

    //组装报文

    unsigned char mac[48];

    for (int i = 0; i < 6; i++) {

        send\_ARPFrame.FrameHeader.DesMAC[i] = 0xFF; //DesMAC设置为广播地址

        send\_ARPFrame.DesMAC[i] = 0x00; //DesMAC设置为0

        //SrcMAC用不到可以不设置

    }

    send\_ARPFrame.FrameHeader.FrameType = htons(0x0806); //帧类型为ARP

    send\_ARPFrame.HardwareType = htons(0x0001); //硬件类型为以太网

    send\_ARPFrame.ProtocolType = htons(0x0800); //协议类型为IP

    send\_ARPFrame.HLen = 6; //硬件地址长度为6

    send\_ARPFrame.PLen = 4; //协议地址长度为4

    send\_ARPFrame.op = htons(0x0001); //操作为ARP请求

    send\_ARPFrame.DesIP = inet\_addr(srcip); //设置为本机IP地址

    struct pcap\_pkthdr\* pkt\_header;

    const u\_char\* packetData;

    int ret;

    while ((ret = pcap\_next\_ex(point, &pkt\_header, &packetData)) >= 0)

    {

        cout << "加载中...";

        //发送构造好的数据包

        pcap\_sendpacket(point, (u\_char\*)&send\_ARPFrame, sizeof(ARP\_Frame));

        if (ret == 0) {  //未捕获到数据包

            continue;

        }

        //通过报文内容比对判断是否是要发打印的ARP数据包内容

         //result=1，捕获成功

        else if (\*(unsigned short\*)(packetData + 12) == htons(0x0806) //帧类型为ARP（htons(0x0806)）

            && \*(unsigned short\*)(packetData + 20) == htons(0x0002)) //操作类型为ARP响应（htons(0x0002)）

        {

            cout << endl;

            cout << "-----------------------------------------------" << endl;

            cout << "ARP数据包内容：" << endl;

            //打印数据包

            cout << "源IP地址:\t ";

            for (int i = 28; i < 32; ++i) {

                printf("%d", packetData[i]);

                if (i < 31) cout << ".";

            }

            cout << endl;

            // 提取MAC地址（0-6字节）

            cout << "源MAC地址:\t ";

            for (int i = 6; i < 12; ++i) {

                printf("%02X", packetData[i]);

                if (i < 11) cout << "-";

            }

            cout << endl;

            //用mac数组记录本机的MAC地址

            for (int i = 0; i < 6; i++)

            {

                mac[i] = \*(unsigned char\*)(packetData + 22 + i);

            }

            cout << "获取MAC地址成功，MAC地址为：";

            for (int i = 6; i < 12; ++i) {

                printf("%02X", packetData[i]);

                if (i < 11) cout << "-";

            }

            cout << endl;

            cout << "----------------------------------------------" << endl;

            break;

        }

    }

    //输出错误信息

    if (ret == -1) {  //调用过程发生错误

        cout << "捕获数据包出错" << endl;

        pcap\_freealldevs(devices);

        return 0;

    }

    ARP\_Frame rev\_ARPFrame;

    /\*获取目的主机的MAC地址\*/

    for (int i = 0; i < 6; i++) {

        rev\_ARPFrame.FrameHeader.DesMAC[i] = 0xff; //广播地址

        rev\_ARPFrame.FrameHeader.SrcMAC[i] = mac[i]; //本机MAC地址

        rev\_ARPFrame.DesMAC[i] = 0x00; //设置为0

        rev\_ARPFrame.SrcMAC[i] = mac[i]; //本机MAC地址

    }

    rev\_ARPFrame.FrameHeader.FrameType = htons(0x0806);

    rev\_ARPFrame.HardwareType = htons(0x0001);

    rev\_ARPFrame.ProtocolType = htons(0x0800);

    rev\_ARPFrame.HLen = 6;

    rev\_ARPFrame.PLen = 4;

    rev\_ARPFrame.op = htons(0x0001);

    rev\_ARPFrame.SrcIP = inet\_addr(srcip);

    cout << "请输入目的IP地址：";

    char ip[INET\_ADDRSTRLEN];

    cin >> ip;

    rev\_ARPFrame.DesIP = inet\_addr(ip);

    while ((ret = pcap\_next\_ex(point, &pkt\_header, &packetData)) >= 0)//判断获取报文

    {

        //发送构造好的数据包

        cout << "加载中...";

        pcap\_sendpacket(point, (u\_char\*)&rev\_ARPFrame, sizeof(ARP\_Frame));

        if (ret == 0) {  //未捕获到数据包

            continue;

        }

        //result=1，捕获成功

        else if (\*(unsigned short\*)(packetData + 12) == htons(0x0806) //帧类型为ARP（htons(0x0806)）

            && \*(unsigned short\*)(packetData + 20) == htons(0x0002) //操作类型为ARP响应（htons(0x0002)）

            && \*(unsigned long\*)(packetData + 28) == rev\_ARPFrame.DesIP)//ip地址为填入的目标IP地址

        {

            cout << endl;

            cout << "-----------------------------------------------" << endl;

            cout << "ARP数据包内容：" << endl;

            //打印数据包

            cout << "源IP地址:\t ";

            for (int i = 38; i < 42; ++i) {

                printf("%d", packetData[i]);

                if (i < 41) cout << ".";

            }

            cout << endl;

            // 提取MAC地址（0-6字节）

            cout << "源MAC地址:\t ";

            for (int i = 0; i < 6; ++i) {

                printf("%02X", packetData[i]);

                if (i < 5) cout << "-";

            }

            cout << endl;

            cout << "目的IP地址:\t ";

            for (int i = 28; i < 32; ++i) {

                printf("%d", packetData[i]);

                if (i < 31) cout << ".";

            }

            cout << endl;

            // 提取目的MAC地址（后6字节）

            cout << "目的MAC地址:\t ";

            for (int i = 6; i < 12; ++i) {

                printf("%02X", packetData[i]);

                if (i < 11) cout << "-";

            }

            cout << endl;

            cout << "获取MAC地址成功，MAC地址为：";

            for (int i = 6; i < 12; ++i) {

                printf("%02X", packetData[i]);

                if (i < 11) cout << "-";

            }

            cout << endl;

            cout << "----------------------------------------------" << endl;

            break;

        }

    }

    //输出错误信息

    if (ret == -1) {  //调用过程发生错误

        cout << "捕获数据包出错" << endl;

        pcap\_freealldevs(devices);

        return 0;

    }

    // 关闭设备

    pcap\_close(point);

    pcap\_freealldevs(devices);

    return 0;

}