实验二：数据包捕获与分析

一、实验内容

数据包捕获与分析编程实验，要求如下：

1.了解Npcap的架构。

2.学习Npcap的设备列表获取方法、网卡设备打开方法，以及数据包捕获方法。3.通过Npcap编程，实现本机的数据包捕获，显示捕获数据帧的源MAC地址和目的MAC地址，以及类型/长度字段的值。

4.捕获的数据报不要求硬盘存储，但应以简单明了的方式在屏幕上显示。必显字段包括源MAC地址、目的MAC地址和类型/长度字段的值。

5.编写的程序应结构清晰，具有较好的可读性。

二、实验步骤

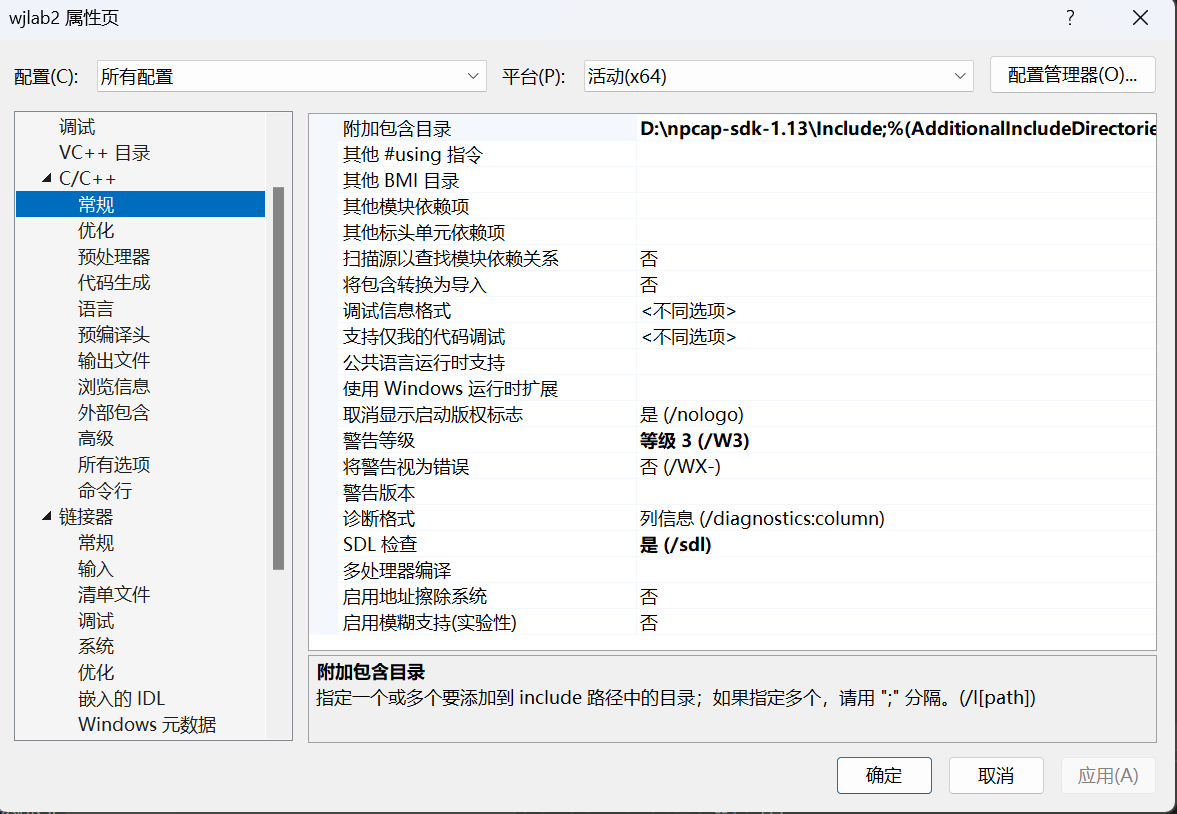
1.Npcap 是一个在 Windows 操作系统上进行网络数据包捕获的工具，是WinPcap的改进版，Npcap 的核心部分是其网络驱动程序NDIS，它与 Windows 网络协议栈集成，允许它在网络协议层捕获数据包。其中还包括NPF，它是一种网络数据包捕获和过滤技术。Npcap还附带一些实用程序和示例应用程序，如wireshark等。Npcap中还包含一些动态链接库文件，如packet.dll，Wpcap.dll。

packet.dll包含了与网络数据包捕获相关的函数和方法，允许应用程序与 NPcap 驱动程序进行交互，以进行数据包捕获和分析。为应用程序提供了捕获和处理数据包的接口。

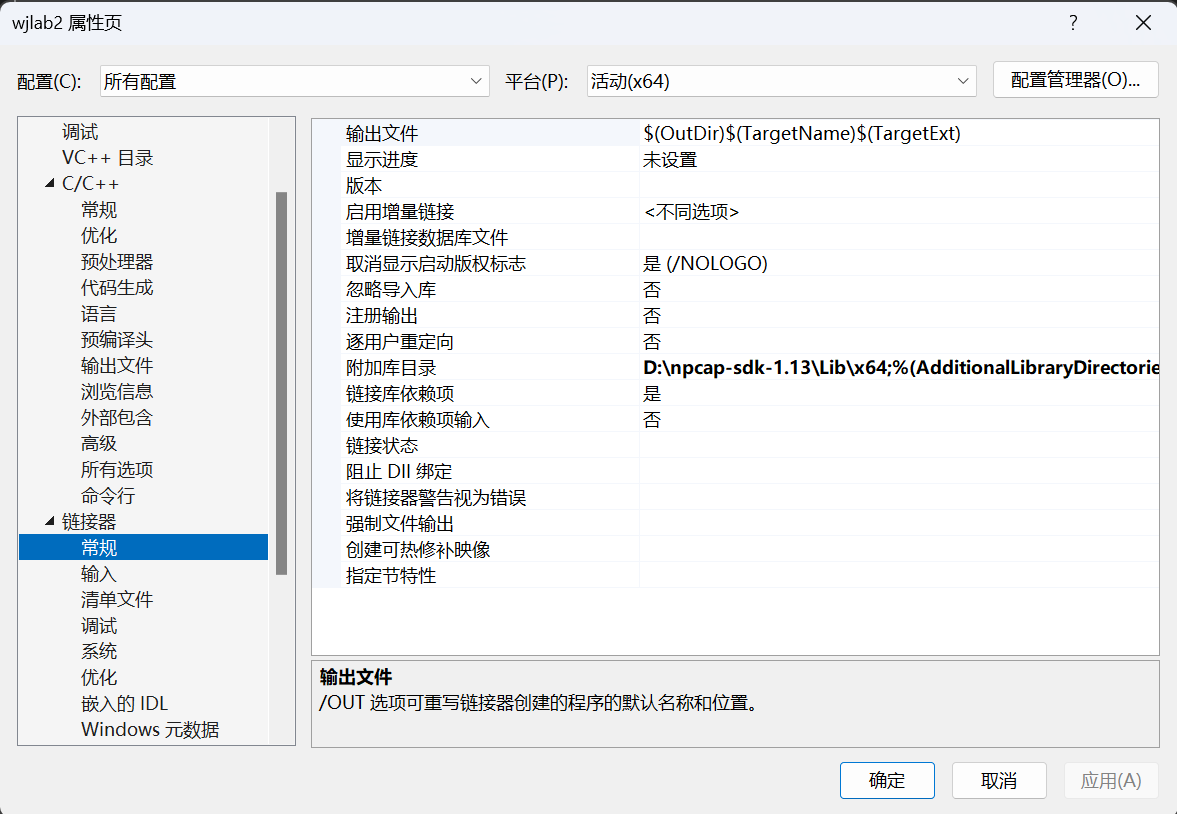
Wpcap.dll包含了与网络数据包捕获和处理相关的功能，为应用程序提供了用于配置捕获参数、设置网络适配器、捕获数据包等功能。

2.使用Npcap前需要配置相关软件包以及头文件。

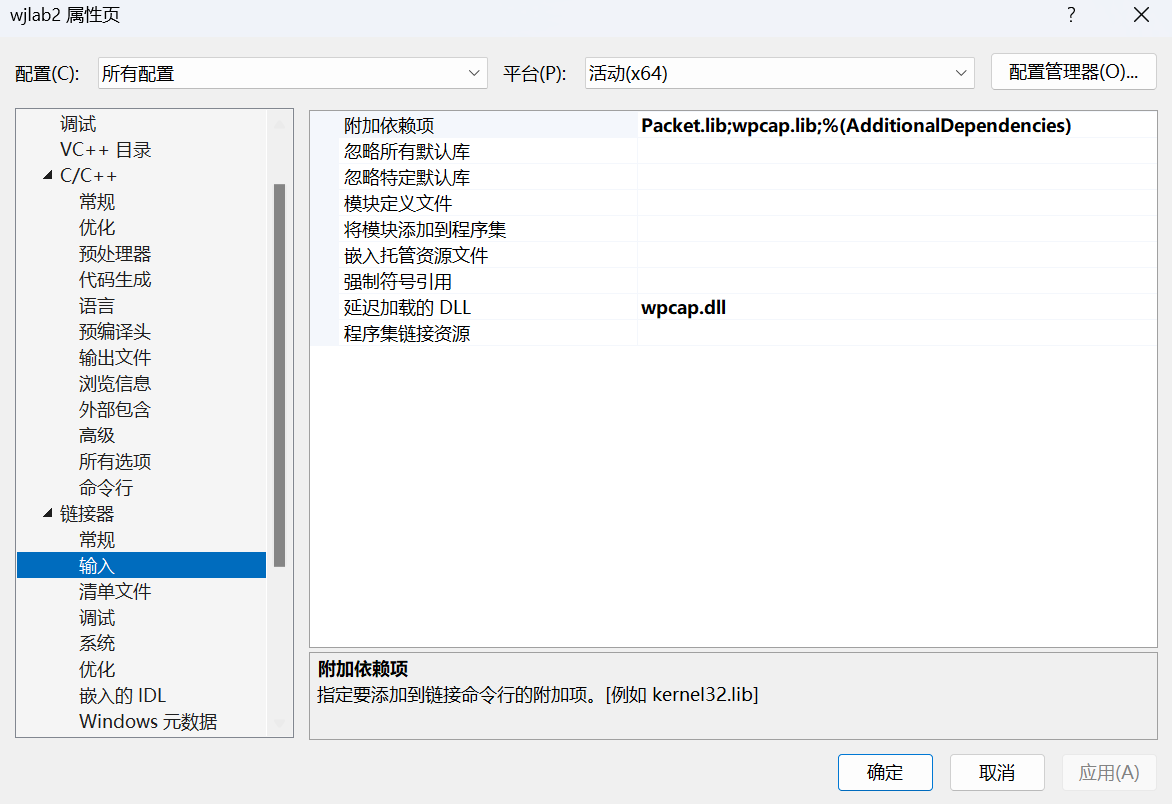
（1）首先需要配置附加包含目录，即npcap-sdk文件中的include文件夹。



（2）配置附加库目录，即npcap-sdk文件中的Lib文件夹。



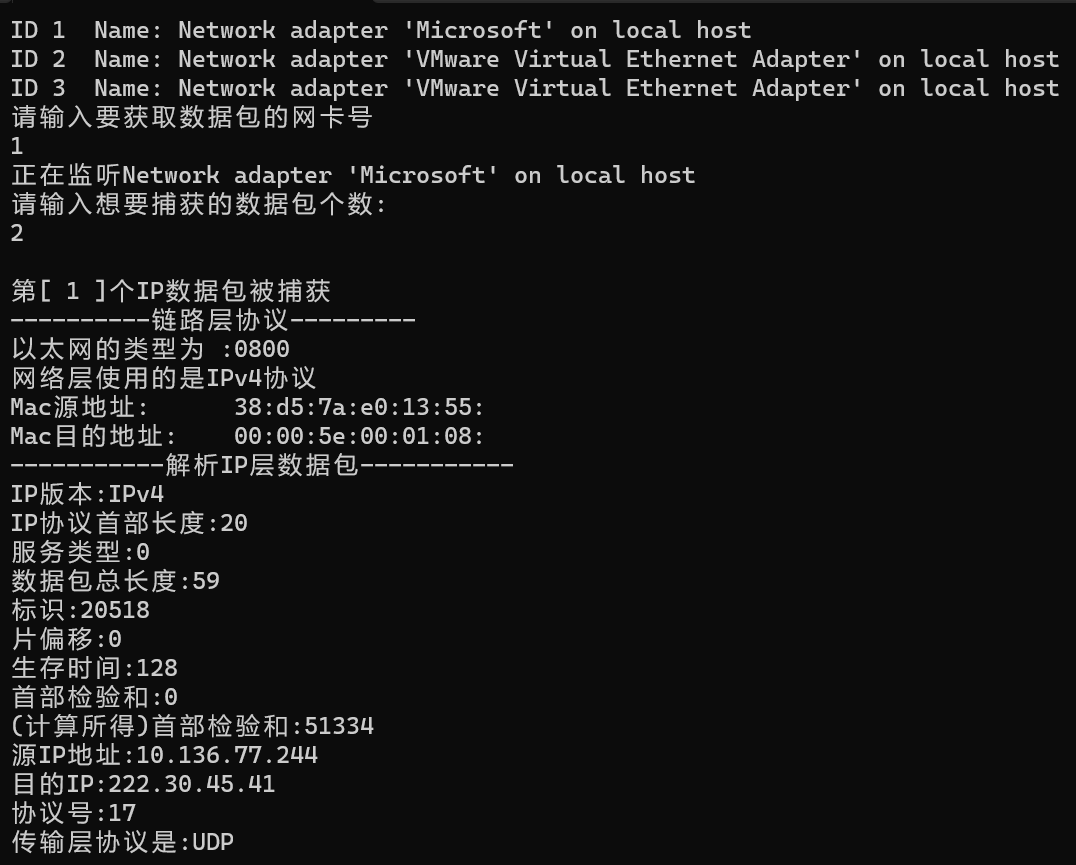
（3）配置附加依赖项，也就是Lib文件夹中的两个动态链接库Packet.dll和wpcap.dll。



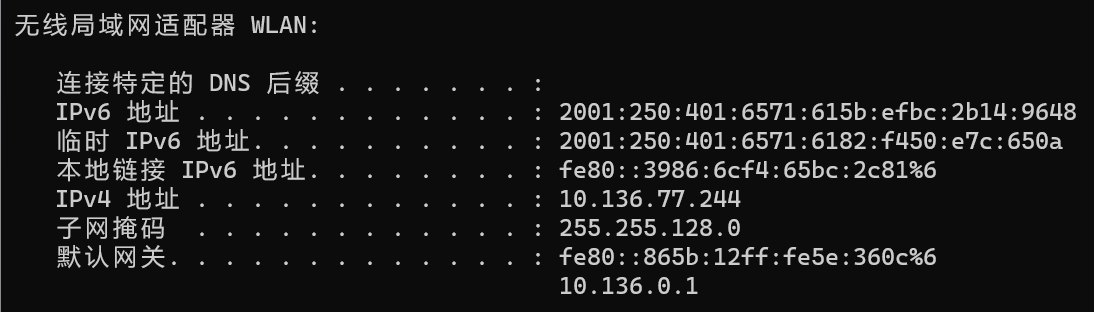
（4）获取设备列表可以使用Npcap中的函数时需要包含头文件 "pcap.h"，然后调用 pcap\_findalldevs 函数来获取网络设备列表。打开网卡设备可以使用 pcap\_open或pcap\_open\_live函数来打开所选的设备以进行数据包捕获。数据包的捕获则可以通过使用 pcap\_next\_ex 函数来捕获一个数据包，

3.编写一段代码捕获本机的数据包，其中通过pcap\_open\_live，pcap\_findalldevs\_ex，pcap\_loop等函数对数据包进行抓取和分析，使用

ip\_protocol\_packet\_callback函数进行IP地址的计算，并且使用了epp\_callback函数对数据链路层进行解析，从而分析出MAC源地址和目的地址。最后使用Catch函数抓取本机的所有网卡，并对整个抓取结果进行合理化输出。



以上是程序的抓取结果，可以看到其中的IPv4地址与本机的地址相同，可见抓取成功。



三、实验感想与研讨

通过数据包捕获与分析实验，我深入了解了网络通信是如何在物理层、数据链路层、网络层和传输层等不同层次上工作的。并了解了数据包如何在网络中传输，以及不同协议（如TCP和UDP）的作用。通过分析捕获的数据包，我学会了如何解读不同协议的数据包头部，以及如何识别数据包中的重要信息。掌握了协议和数据包结构的解析技能，提升了网络安全意识，并提升了实际应用的技能。

四、实验代码

#include <Winsock2.h>

#include<Windows.h>

#include<iostream>

#include <ws2tcpip.h>

#include "pcap.h"

#include "stdio.h"

#include<time.h>

#include <string>

#pragma comment(lib, "Packet.lib")

#pragma comment(lib,"wpcap.lib")

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

#pragma warning( disable : 4996 )

#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS

using namespace std;

#pragma pack(1)

//帧的首部

struct e\_head

{

uint8\_t ether\_dst[6];

uint8\_t ether\_src[6];

uint16\_t ether\_type;

};

//IP的首部

struct ip\_head

{

uint8\_t ip\_header\_length : 4,ip\_version : 4;

uint8\_t tos;

uint16\_t total\_length;

uint16\_t ip\_id;

uint16\_t ip\_offset;

uint8\_t ttl;

uint8\_t ip\_protocol;

uint16\_t ip\_checksum;

uint16\_t cal\_checksum();

struct in\_addr ip\_source\_address;

struct in\_addr ip\_destination\_address;

};

uint16\_t ip\_head::cal\_checksum()

{

uint32\_t cal\_checksum = 0;

uint16\_t var1 = (((this->ip\_version << 4) + this->ip\_header\_length) << 8) + this->tos;

uint16\_t var2 = (this->ttl << 8) + this->ip\_protocol;

uint16\_t var3 = ntohl(this->ip\_source\_address.S\_un.S\_addr) >> 16;

uint16\_t var4 = ntohl(this->ip\_source\_address.S\_un.S\_addr);

uint16\_t var5 = ntohl(this->ip\_destination\_address.S\_un.S\_addr) >> 16;

uint16\_t var6 = ntohl(this->ip\_destination\_address.S\_un.S\_addr);

cal\_checksum = cal\_checksum + var1 + ntohs(this->total\_length) + ntohs(this->ip\_id) + ntohs(this->ip\_offset) + var2 + var3 + var4 + var5 + var6;

cal\_checksum = (cal\_checksum >> 16) + (cal\_checksum & 0xffff);

cal\_checksum += (cal\_checksum >> 16);

return (uint16\_t)(~cal\_checksum);

}

//分析IP数据包

void ip\_protocol\_packet\_callback(u\_char\* argument, const struct pcap\_pkthdr\* packet\_header, const u\_char\* packet\_content)

{

ip\_head\* ip\_protocol;

uint32\_t head\_length;

uint16\_t offset;

uint8\_t tos;

uint16\_t checksum;

ip\_protocol = (struct ip\_head\*)(packet\_content + 14);

checksum = ntohs(ip\_protocol->ip\_checksum);

head\_length = ip\_protocol->ip\_header\_length \* 4;

tos = ip\_protocol->tos;

offset = ntohs(ip\_protocol->ip\_offset);

cout << "-----------解析IP层数据包----------- " << endl;

printf("IP版本:IPv%d\n", ip\_protocol->ip\_version);

cout << "IP协议首部长度:" << head\_length << endl;

printf("服务类型:%d\n", tos);

cout << "数据包总长度:" << ntohs(ip\_protocol->total\_length) << endl;

cout << "标识:" << ntohs(ip\_protocol->ip\_id) << endl;

cout << "片偏移:" << (offset & 0x1fff) \* 8 << endl;

cout << "生存时间:" << int(ip\_protocol->ttl) << endl;

cout << "首部检验和:" << htons(checksum) << endl;

cout << "(计算所得)首部检验和:" << htons(ip\_protocol->cal\_checksum()) << endl;

char src[17];

::inet\_ntop(AF\_INET, (const void\*)&ip\_protocol->ip\_source\_address, src, 17);

cout << "源IP地址:" << src << endl;

char dst[17];

::inet\_ntop(AF\_INET, (const void\*)&ip\_protocol->ip\_destination\_address, dst, 17);

cout << "目的IP:" << dst << endl;

printf("协议号:%d\n", ip\_protocol->ip\_protocol);

cout << "传输层协议是:";

switch (ip\_protocol->ip\_protocol)

{

case 1:

cout << "ICMP" << endl;

break;

case 2:

cout << "IGMP" << endl;

break;

case 3:

cout << "GGP" << endl;

break;

case 6:

cout << "TCP" << endl;

break;

case 8:

cout << "EGP" << endl;

break;

case 17:

cout << "UDP" << endl;

break;

case 89:

cout << "OSPF" << endl;

break;

default:break;

}

}

//解析数据链路层，获取MAC地址

void epp\_callback(u\_char\* argument, const pcap\_pkthdr\* packet\_header, const u\_char\* packet\_content)

{

uint16\_t e\_type;

e\_head\* e\_protocol = (e\_head\*)packet\_content;

uint8\_t\* mac\_src;

uint8\_t\* mac\_dst;

static int packet\_number = 1;

e\_type = ntohs(e\_protocol->ether\_type);

e\_protocol = (e\_head\*)packet\_content;

mac\_src = e\_protocol->ether\_src;

mac\_dst = e\_protocol->ether\_dst;

cout << endl;

printf("第[ %d ]个IP数据包被捕获\n", packet\_number);

cout << "----------链路层协议---------" << endl;;

printf("以太网的类型为 :%04x\n", e\_type);

switch (e\_type)

{

case 0x0800:

cout << "网络层使用的是IPv4协议" << endl;

break;

case 0x0806:

cout << "网络层使用的是ARP协议" << endl;

break;

case 0x8035:

cout << "网络层使用的是RARP协议" << endl;

break;

default: break;

}

printf("Mac源地址:\t%02x:%02x:%02x:%02x:%02x:%02x:\n", \*mac\_src, \*(mac\_src + 1), \*(mac\_src + 2), \*(mac\_src + 3), \*(mac\_src + 4), \*(mac\_src + 5));//X 表示以十六进制形式输出 02 表示不足两位，前面补0输出

printf("Mac目的地址:\t%02x:%02x:%02x:%02x:%02x:%02x:\n", \*mac\_dst, \*(mac\_dst + 1), \*(mac\_dst + 2), \*(mac\_dst + 3), \*(mac\_dst + 4), \*(mac\_dst + 5));

switch (e\_type)

{

case 0x0800:

ip\_protocol\_packet\_callback(argument, packet\_header, packet\_content);

break;

default:

cout << "不是IP数据包，不进行解析" << endl;

break;

}

packet\_number++;

}

void Catch()

{

pcap\_if\_t\* allAdapters;

pcap\_if\_t\* ptr;

pcap\_t\* pcap\_handle;

int index = 0;

int num = 0;

int i = 0;

char errbuf[PCAP\_ERRBUF\_SIZE];

int flag = 0;

char packet\_filter[40] = "";

struct bpf\_program fcode;

u\_int netmask;

if (pcap\_findalldevs\_ex(PCAP\_SRC\_IF\_STRING, NULL, &allAdapters, errbuf) != -1)

{

for (ptr = allAdapters; ptr != NULL; ptr = ptr->next)

{

++index;

if (ptr->description)

printf("ID %d Name: %s \n", index, ptr->description);

}

}

if (index == 0)

{

cout << "没有找到接口，请确认是否安装了Npcap或WinPcap" << endl;

}

cout << "请输入要获取数据包的ID" << endl;

cin >> num;

if (num < 1 || num > index)

{

cout << "ID不在上述列表中" << endl;

pcap\_freealldevs(allAdapters);

}

for (ptr = allAdapters, i = 0; i < num - 1; ptr = ptr->next, i++);

if ((pcap\_handle = pcap\_open\_live(ptr->name,65536,PCAP\_OPENFLAG\_PROMISCUOUS,1000,errbuf)) == NULL)

{

cout << "无法打开适配器,Npcap不支持" << endl;

pcap\_freealldevs(allAdapters);

exit(0);

}

cout << "正在监听" << ptr->description << endl;

pcap\_freealldevs(allAdapters);

int cnt = -1;

cout << "请输入想要捕获的数据包个数:" << endl;

cin >> cnt;

pcap\_loop(pcap\_handle, cnt, epp\_callback, NULL);

cout << "解析ip数据包结束" << endl;

}

int main()

{

Catch();

return 0;

}