



南开大学
Nankai University

南 开 大 学

计 算 机 学 院

网络技术与应用实验报告

IP 数据报的捕获与分析

姓 名：董伊萌

年 级：2020 级

专 业：信息安全

指导教师：张建忠

2022 年 10 月 21 日

摘要

本次实验通过 Npcap 编制一个简单的 IP 网络数据报捕获与分析程序，学习 IP 数据报校验和计算方法，通过 NPcap 编程，实现本机的 IP 数据报捕获，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。

关键字：Npcap，源 MAC 地址，目的 MAC 地址

目录

一、 实验内容说明	1
二、 实验前期准备	1
三、 实验具体过程	2
(一) 实验整体思路	2
(二) 程序中用到的主要数据结构	3
(三) 程序中用到的主要函数	3
(四) 程序中用到的主要代码部分解释	3
四、 实验程序代码	6
五、 实验程序运行结果	10
六、 总结	10

一、实验内容说明

在对网络的安全性和可靠性进行分析时，网络管理员通常需要对网络中传输的数据包进行监听和分析，本次实验要求通过 Npcap 编制一个简单的 IP 网络数据报捕获与分析程序，学习 IP 数据报校验和计算方法，初步掌握网络监听与分析技术的实现过程，加深对网络协议的理解。具体实验要求如下：

- 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 IP 数据报捕获，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、实验前期准备

1. 安装 WinPcap 驱动程序和 DLL 程序。
2. 创建基于 WinPcap 的应用程序，主要包括：
 - 添加 pcap.h 包含文件：需要在该文件的开始位置增加 pcap.h 包含文件 `#include "pcap.h"`。
 - 增加与 WinPcap 有关的预处理器定义：将标号 WPCAP 和 HAVE_REMOTE 添加到预处理器定义。

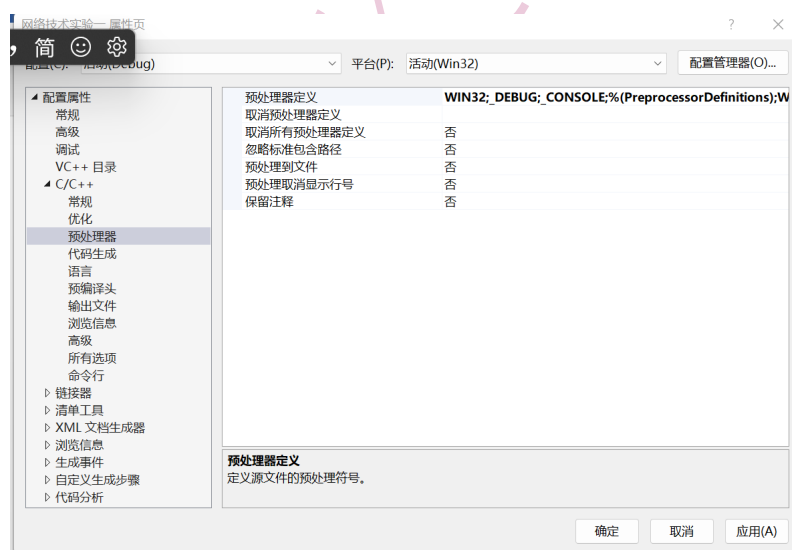


图 1: 前期准备

- 添加包含文件目录：将 pcap.h 所在的 Include 文件夹添加到 IDE 中；
- 添加 wpcap.lib 库文件：将 wpcap.lib 所在的 Lib 文件夹添加到 IDE 中。

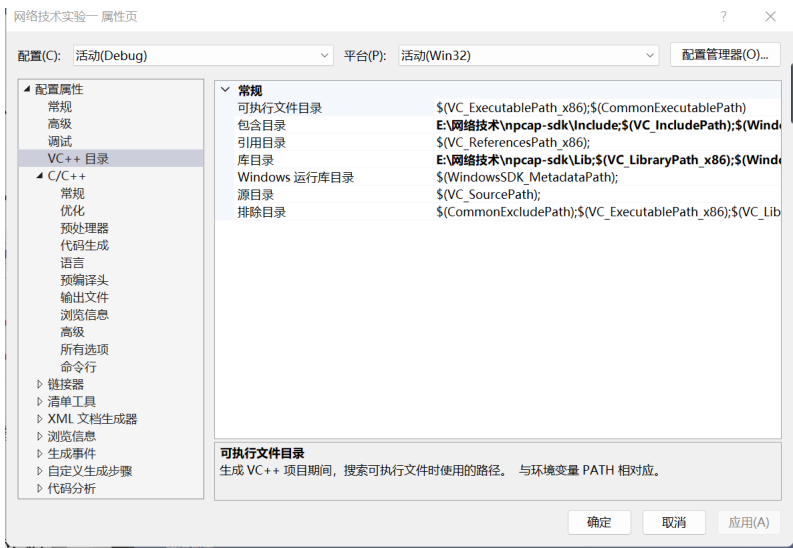


图 2: 前期准备

3. 学习 WinPcap 的设备列表获取方法、网卡设备打开方法、数据包捕获方法以及多线程程序编写方法。

三、 实验具体过程

(一) 实验整体思路

实验整体思路如下图所示：

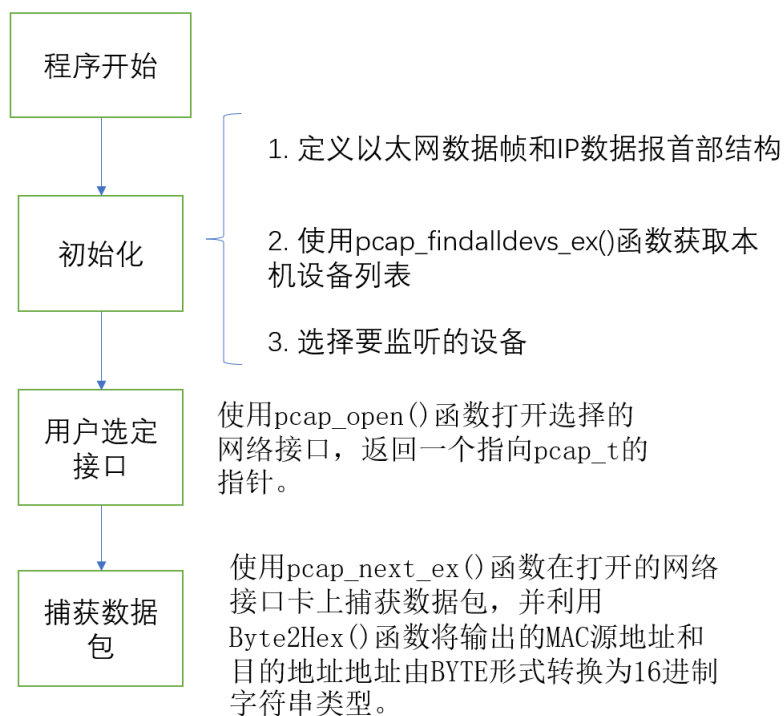


图 3: 实验思路

(二) 程序中用到的主要数据结构

- FrameHeader_t: 帧首部, 包含 48 位的源 MAC 地址、目的 MAC 地址和帧类型。
- IPHeader_t: IP 首部, 包含校验和、源 IP 地址、目的 IP 地址等等。
- Data_T: 数据包, 包含帧首部和 IP 首部。
- pcap_if: 网络接口卡中的链表, 包括网络接口卡的名字, 指向该网卡描述的字符串等等。

(三) 程序中用到的主要函数

- CAPLIST(): 自定义函数, 利用 pcap_findalldevs_ex(PCAP_SRC_IF_STRING, NULL, &alldevs, errbuf) 获取本机的接口设备列表。返回 alldevs 参数指向获取的网络接口列表的第一个元素。
- 自定义函数 CHOOSEDEV(int M, pcap_if_t* D, pcap_if_t* alldevs): 监听选定的网络接口卡, 并利用 pcap_open(D->name, 100, PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL, errbuf) 函数打开网络接口。
- 自定义函数 Byte2Hex(unsigned char bArray[], int bArray_len) 将地址由 BYTE 形式转换为 16 进制字符串类型。
- 自定义函数 Capture(), 利用 pcap_next_ex(chosed_dev, &pkt_header, &pkt_data) 函数在打开的网络接口卡上捕获网络数据包。

(四) 程序中用到的主要代码部分解释

1. 用 pcap_findalldevs_ex() 函数获取网络接口设备列表。获取成功, 则继续向下打印, 获取失败, 则打印 errbuf 里的错误信息, 函数返回。如果 pcap_findalldevs_ex() 函数调用成功,

alldevs 参数指向获取的网络接口列表的第一个元素，通过指针和循环，遍历所有的网络接口并输出相应的信息，包括设备名字和设备描述。

```

1  if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, //获取本机的接口设备
2      NULL, //无需认证
3      &alldevs, //指向设备列表首部
4      errbuf //出错信息保存缓冲区
5  ) != 0)
6  {
7      //出错信息处理,标准错误输出
8      cout << stderr << "error in pcap_findalldevs_ex" << errbuf << endl;
9  }
10 //!=1时,显示接口列表
11
12 for (d = alldevs; d != NULL; d = d->next) {
13     cout << n << ". " << d->name;
14     if (d->description)
15         cout << ":" << d->description << endl;
16     else
17         cout << "NO DESCRIPTION FOUND;" << endl;
18     n++;
19 }

```

2. 用户指定选择捕获数据包的设备，调用函数 pcap_open() 函数打开选择的设备的网卡，如果打开失败，打印 errbuf 里的错误信息，调用 pcap_freealldevs() 释放该设备列表，函数返回。

```

1  choosed_dev = pcap_open(D->name, 100, PCAP_OPENFLAG_PROMISCUOUS, 1000, NULL,
2      errbuf);
3  if (choosed_dev == NULL) {
4      cout << stderr << "error in pcap_open" << errbuf << endl;
5      pcap_freealldevs(alldevs);
6      return ;
7  }

```

3. 通过 Capurer() 函数在打开的网络接口卡上捕获网络数据包，循环调用 pcap_next_ex() 函数，如果该函数返回值 =0，则无法捕获到数据包，但不是错误，需要继续捕获；如果返回值 =1，则成功捕获到了一个数据包，输出数据包的相应信息：源 MAC 地址、目的 MAC 地址、类型；如果返回值 =-1，调用过程中发生了错误，需要打印错误信息。此过程我们需要重复多次因为捕获数据包的正确率不是百分百的。

```

1  int m = pcap_next_ex(choosed_dev, &pkt_header, &pkt_data);
2  while (m!= -1) {
3      //cout << pcap_next_ex(choosed_dev, &pkt_header, &pkt_data) << endl;
4      if (m == 0) //pkt_data指向捕获到的网络数据包
5          continue;
6      else {
7          //将捕获到的数据报信息进行输出;
8          Data_t* pack; //包含帧首部和IP首部的数据包
9          pack = (Data_t*)pkt_data;
10         cout << "捕获到的第" << i << "个数据包: ";

```

```

11         cout << "源MAC地址: " << *(Byte2Hex(pack->FrameHeader.SrcMAC,6))
12             << endl;
13         cout << "目的MAC地址: " << *(Byte2Hex(pack->FrameHeader.DesMAC,6))
14             << endl;
15         cout << "类型: " << pack->FrameHeader.TrameType << endl;
16         i++;
17     }
18
19     if (i == 10)
20         break;
21 }
22
23 if ((pcap_next_ex(choosed_dev, &pkt_header, &pkt_data)) == -1)
24     cout << "error in pcap_next_ex" << endl;

```

4. 为了更方便的判断获取的数据包的正确性, 利用函数 Byte2Hex() 将地址由 BYTE 形式转换为 16 进制字符串类型。

```

1 string* Byte2Hex(unsigned char bArray[], int bArray_len)
2 {
3     string* strHex = new string();
4     int nIndex = 0;
5     for (int i = 0; i < bArray_len; i++)
6     {
7         char hex1;
8         char hex2;
9         int value= bArray [i];
10        int S = value / 16;
11        int Y = value % 16;
12        if (S >= 0 && S <= 9)
13            hex1 = (char)(48 + S);
14        else
15            hex1 = (char)(55 + S);
16        if (Y >= 0 && Y <= 9)
17            hex2 = (char)(48 + Y);
18        else
19            hex2 = (char)(55 + Y);
20        if (i != bArray_len - 1) {
21            *strHex = *strHex + hex1 + hex2 + "-";
22        }
23        else
24            *strHex = *strHex + hex1 + hex2;
25    }
26
27    return strHex;
28 }

```

5. 最终在主函数中, 需要调用 pcap_freealldevs(alldevs) 释放所有的设备。

四、 实验程序代码

实验程序的全部代码如下所示：

```
1 #include "pcap.h"
2 #include <iostream>
3 #include <WinSock2.h>
4 #include <process.h>
5 #include <bitset>
6 #define WIN32
7 #define HAVE_REMOTE
8 #define BYTE unsigned char
9 using namespace std;
10 #pragma comment(lib, "wpcap.lib")
11 #pragma comment(lib, "ws2_32.lib")
12 #pragma warning(disable:4996)
13
14 pcap_t* choosed_dev; //全局变量
15 //IP数据报的提取
16 #pragma pack(1)
17 typedef struct FrameHeader_t { //帧首部
18     BYTE DesMAC[6]; //目的地址
19     BYTE SrcMAC[6]; //源地址
20     WORD TrameType; //帧类型
21
22 }FrameHeader_t;
23
24 typedef struct IPHeader_t { // IP首部
25     BYTE Ver_HLen;
26     BYTE TOS;
27     WORD TotalLen;
28     WORD ID;
29     WORD Flag_Segment;
30     BYTE TTL;
31     BYTE Protocol;
32     WORD Checksum;
33     ULONG SrcIP;
34     ULONG DstIP;
35 }IPHeader_t;
36
37 typedef struct Data_t { //包含帧首部和IP首部的数据包
38     FrameHeader_t FrameHeader;
39     IPHeader_t IPHeader;
40 }Data_t;
41
42 #pragma pack()
43
44 //获取设备列表
45 pcap_if_t* CAPLIST() {
```



```

46     pcap_if_t    *alldevs;    //指向设备链表首部的指针
47     pcap_if_t    *d;
48     pcap_addr_t  *a;
49     int           n=1;        //通过n来选择后续想要监听的设备
50     char          errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
51
52     //获取本机的设备列表
53     if (pcap_findalldevs_ex(PCAP_SRC_IF_STRING, //获取本机的接口设备
54         NULL, //无需认证
55         &alldevs, //指向设备列表首部
56         errbuf //出错信息保存缓冲区
57     ) !=0)
58     {
59         //出错信息处理,标准错误输出
60         cout << stderr << "error in pcap_findalldevs_ex" << errbuf << endl;
61     }
62     //!=1时, 显示接口列表
63
64     for (d = alldevs; d != NULL; d = d->next) {
65         cout <<n<<". "<< d->name;
66         if (d->description)
67             cout << ":" << d->description << endl;
68         else
69             cout << "NO DESCRIPTION FOUND;" << endl;
70         n++;
71     }
72     //需要释放设备列表吗??
73     //pcap_freealldevs(alldevs);
74     return alldevs; //pcap_findalldevs_ex函数调用成功后, alldevs参数指向获
        取的网络接口列表的第一个元素
75 }
76 //监听选定的网络接口卡——打开网络接口
77 void CHOOSEDEV(int M, pcap_if_t* D, pcap_if_t* alldevs) {
78     //指针指向要监听的设备
79     int i = 1;
80     char          errbuf[PCAP_ERRBUF_SIZE]; //错误信息缓冲区
81     while (i < M )
82     {
83         D = D->next;
84         i++;
85     }
86
87     //打开选择的网络接口, 返回一个指向pcap_t的指针
88
89     choosed_dev = pcap_open(D->name, 100, PCAP_OPENFLAG_PROMISCUOUS, 1000,
        NULL, errbuf);
90     if (choosed_dev == NULL) {
91         cout << stderr << "error in pcap_open" << errbuf << endl;

```

```
92     pcap_freealldevs(alldevs);
93     return ;
94 }
95 return;
96
97
98 }
99 //将地址由BYTE形式转换为16进制字符串类型
100 string* Byte2Hex(unsigned char bArray[], int bArray_len)
101 {
102     string* strHex = new string();
103     int nIndex = 0;
104     for (int i = 0; i < bArray_len; i++)
105     {
106         char hex1;
107         char hex2;
108         int value= bArray [i];
109         int S = value / 16;
110         int Y = value % 16;
111         if (S >= 0 && S <= 9)
112             hex1 = (char)(48 + S);
113         else
114             hex1 = (char)(55 + S);
115         if (Y >= 0 && Y <= 9)
116             hex2 = (char)(48 + Y);
117         else
118             hex2 = (char)(55 + Y);
119         if (i != bArray_len - 1) {
120             *strHex = *strHex + hex1 + hex2 + "-";
121         }
122         else
123             *strHex = *strHex + hex1 + hex2;
124     }
125
126     return strHex;
127 }
128 //再打开的网络接口卡上捕获网络数据包
129 void Capture() {
130
131     int i = 1;
132     struct pcap_pkthdr* pkt_header;
133     const u_char* pkt_data;
134     //循环调用pcap_next_ex()捕获数据报
135     int m = pcap_next_ex(choosed_dev, &pkt_header, &pkt_data);
136     while (m!= -1) {
137         //cout << pcap_next_ex(choosed_dev, &pkt_header, &pkt_data) << endl;
138         if (m == 0) //pkt_data指向捕获到的网络数据包
139             continue;
```

```
140     else {
141         //将捕获到的数据报信息进行输出;
142         Data_t* pack; //包含帧首部和IP首部的数据包
143         pack = (Data_t*)pkt_data;
144         cout << "捕获到的第" << i << "个数据包: ";
145         cout << "源MAC地址: " << *(Byte2Hex(pack->FrameHeader.SrcMAC,6))
146             << endl;
147         cout << "目的MAC地址: " << *(Byte2Hex(pack->FrameHeader.DesMAC,6))
148             << endl;
149         cout << "类型: " << pack->FrameHeader.TrameType << endl;
150         i++;
151     }
152     if (i == 10)
153         break;
154 }
155 if ((pcap_next_ex(choosed_dev, &pkt_header, &pkt_data)) == -1)
156     cout << "error in pcap_next_ex" << endl;
157
158 //_endthread();
159
160 }
161
162
163 int main() {
164
165     //CAPLIST();
166
167
168
169     pcap_if_t *alldevs; //指向设备链表首部的指针
170     alldevs= CAPLIST();
171     //选择要监听的设备
172     cout << "请选择要监听的设备: ";
173     pcap_if_t *D;
174     D = alldevs;
175     int m;
176     cin >> m;
177     CHOOSEDEV(m, D, alldevs);
178
179     Capture();
180
181     pcap_freealldevs(alldevs);
182
183
184
185 }
```

五、 实验程序运行结果

运行结果截图如下：

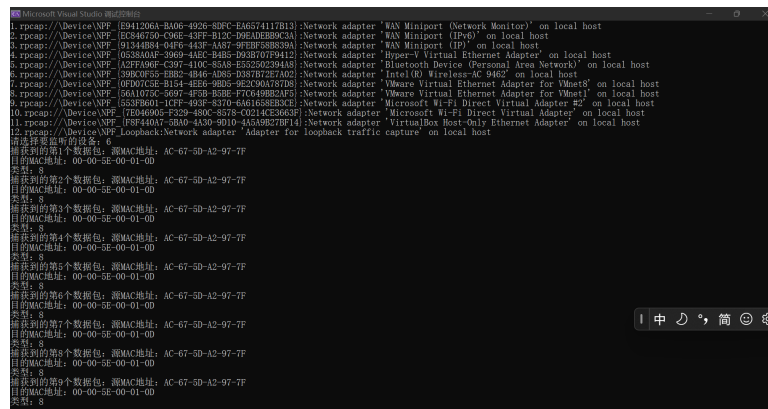


图 4: Caption

与命令行获取的信息对比：

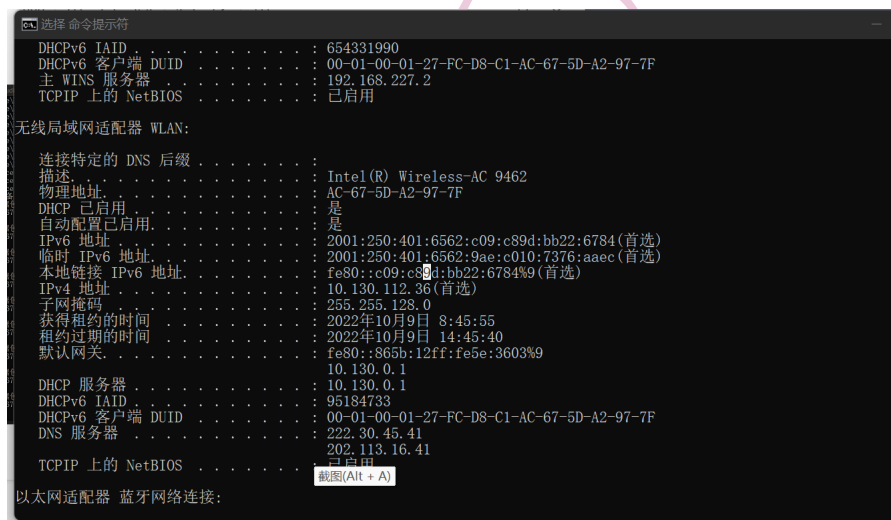


图 5: Caption

可以发现：选择监听“Intel(R) Wireless-AC 9462”程序得到的源 MAC 地址与本机获取的物理地址对应相同，可以证明捕获到的数据包是正确的。但是结果的正确性具有随机性。

六、 总结

本实验利用 Npcap 提供的功能获取了网络接口设备列表和各个接口的详细信息，并且能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。通过本次实验，对 Npcap 关于捕获数据包的功能具有了一定的了解，学会了获取设备列表，打开网络接口以及捕获数据包的方法，受益匪浅！

参考文献

NKU