## 网络攻防实验报告

姓名：\*\*\* 学号：2021E\*\*\*

### 实验名称：嗅探器设计与实现

### 实验内容：

设计并实现一个网络嗅探器，要求能对数据包进行抓取过滤和分析。

### 实验环境：

开发环境：Visual Studio2022

架构类型：MFC

外部库：Winpcap

### 嗅探器设计：

#### 4.1、 获取网卡数据

直接使用winpcap的pcap\_findalldevs\_ex函数获取网卡数据

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

return FALSE;

for (d = alldevs; d; d = d->next) {

m\_list1.InsertItem(0, (CString)d->name);

m\_list1.SetItemText(0, 1, (CString)d->description);

}

#### **4.2、数据抓取**

if ((pCap = pcap\_open\_live(pDlg->m\_pDevice->name, 65536, PCAP\_OPENFLAG\_PROMISCUOUS, 1000, pDlg->strErrorBuf)) == NULL)

{

return -1;

}//抓取数据并进行判断

if (pDlg->m\_pDevice->addresses != NULL)

/\* 获得接口第一个地址的掩码 \*/

netmask = ((struct sockaddr\_in\*)(pDlg->m\_pDevice->addresses->netmask))->sin\_addr.S\_un.S\_addr;

else

/\* 如果接口没有地址，那么我们假设一个C类的掩码 \*/

netmask = 0xffffff;

//编译过滤器

if (pcap\_compile(pCap, &fcode, pDlg->m\_filtername, 1, netmask) < 0)

{

AfxMessageBox(\_T("请设置过滤规则"));

return -1;

}

//设置过滤器

if (pcap\_setfilter(pCap, &fcode) < 0)

return -1;

CFileFind file;

char thistime[30];

struct tm\* ltime;

memset(pDlg->filepath, 0, 512);

memset(pDlg->filename, 0, 64);

//创建目录用于保存抓取的数据

if (!file.FindFile(\_T("SavedData")))

{

CreateDirectory(\_T("SavedData"), NULL);

}

//对抓取的数据进行处理

time\_t nowtime;

time(&nowtime);

ltime = localtime(&nowtime);

strftime(thistime, sizeof(thistime), "%Y%m%d %H%M%S", ltime);

strcpy(pDlg->filepath, "SavedData\\");

strcat(pDlg->filename, thistime);

strcat(pDlg->filename, ".lix");

strcat(pDlg->filepath, pDlg->filename);

pDlg->dumpfile = pcap\_dump\_open(pCap, pDlg->filepath);

if (pDlg->dumpfile == NULL)

{

AfxMessageBox(\_T("文件创建错误！"));

return -1;

}

while ((res = pcap\_next\_ex(pCap, &pkt\_header, &pkt\_data)) >= 0)

{

if (res == 0)

continue;

if (!pDlg->m\_bFlag)

break;

CSnifferDlg\* pDlg = (CSnifferDlg\*)AfxGetApp()->GetMainWnd();

pDlg->ShowPacketList(pkt\_header, pkt\_data);

//进行文件写入

if (pDlg->dumpfile != NULL)

{

pcap\_dump((unsigned char\*)pDlg->dumpfile, pkt\_header, pkt\_data);

}

pDlg = NULL;

}

#### **4.3、端口处理**

在进行端口处理时嗅探器主要选择了对常用的一些应用程序包括微信、QQ、百度网盘、火狐、虚拟机等进行数据处理，由于在Windows的设计中为了保证计算机的安全性，对于应用程序的端口分类都是动态分配的，嗅探器在对数据进行分类判断程序时是通过端口号来判断应用程序的，因此当程序更换主机后，可能需要重新修改端口号。

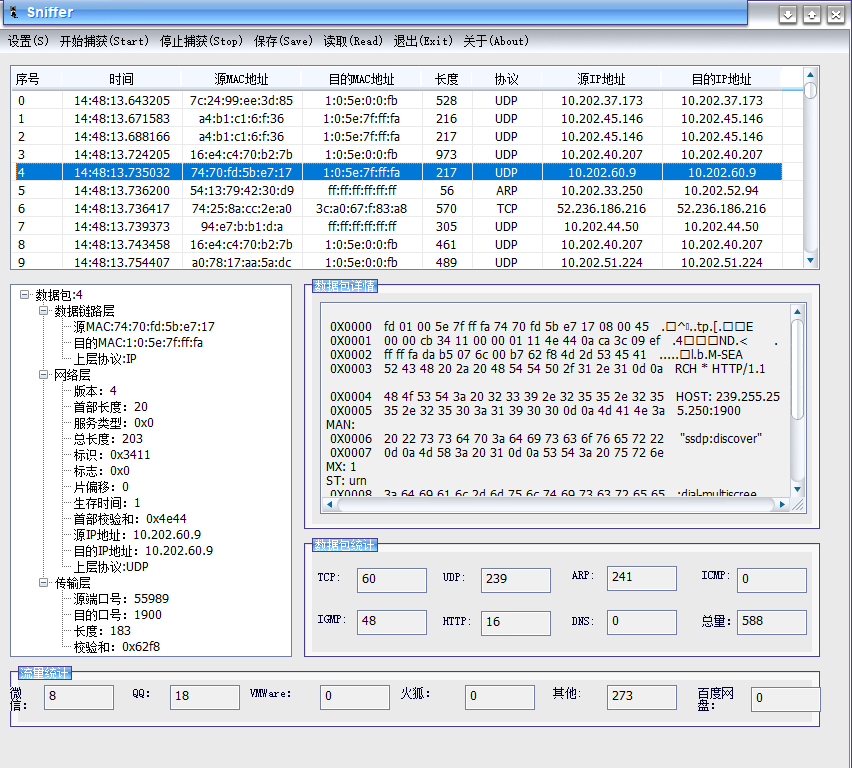
如下面代码为对火狐端口号中的tcp数据进行监听和计算代码：

if (ntohs(th->sport) == 0xC511 || ntohs(th->dport) == 0xC511 //50449 450

|| ntohs(th->sport) == 0xC512 || ntohs(th->dport) == 0xC512)

m\_firfoxCount++;

### 嗅探器界面：



### 实验体会和改进空间：

在编写嗅探器的同时学习并了解了如何配置和使用Winpcap库，增加了对网上抓取数据的理解。在实现嗅探器的过程中也遇到过各种困难，通过不断查找资料去解决问题才终于完成了这个作业，然而，由于时间有限，代码仍然存在着很多的改进空间。

例如，在对应用程序进行分类时，项目是直接查看电脑的进程PID和对应端口号，通过端口号来判断数据为哪个应用程序发出，通过查看任务管理和任务监视器可以知道，系统应该是能动态获取应用程序的监听端口号的，如果能实现这一功能，程序的迁移性将会得到更大提升。