网管信息窃听方法及C++实现

孙金萍，　雷英杰，　赵学军
(空军工程大学 导弹学院，陕西 三原　713800)

摘　要：网管信息十分复杂，其范围也十分广泛，这势必容易造成漏洞，威胁到网管信息的安全，通过研究多种网络交换机的内部网管信息结构，从安全威胁方面入手，着重介绍在Visual C++6.0下用MFC Socket类编写程序，实现网管信息窃听的一种方法。

关键词：计算机网络；信息窃听；C++；Socket编程
中图分类号：TP274　　**文献标识码**：A　　**文章编号**：1009-3516(2001)01-0066-04

在网络软件产品开发过程中，往往需要研究多种以太网交换机(又称智能集线器)的内部网管信息结构，为此，必须编写出一个“窃听”程序，把网管程序和交换机的通讯内容记录下来加以分析。本文介绍在Visual C++6.0下用MFC Socket类编写程序，实现上述目的的一种方法。

1　基本原理

目前标准的网络管理程序与支持网管的网络设备之间大多采用标准的SNMP进行通讯。SNMP是一种高层协议，建立于UDP/IP之上。通讯双方按照SNMP格式来传递各种网管信息和控制信息，进行事件实时报告或报警，从而使网络管理员能方便及时地控制网络当前的运行情况[1]。网管信息的范围十分广泛，如网络流量，连接状态等，因被管设备的不同而不同，各厂家也依照有关的国际标准自定义自家产品的网管信息[2]。网管信息集中定义于管理信息库(MIB)中，整个体系是一个可扩展的树状结构。一条条的网管信息被懈装在SNMP协议包内，再往下传给传输层，转成UDP包，然后通过Socket机制发送出去[3~6]。

本方法的基本原理是：将程序插在网管程序和被管设备之间运行，与网管程序通讯时充当被管设备，与被管设备通讯时充当网管程序，“上传下达”，同步记录，以达“窃听”之目的。

2　类设计

2.1　界面设计

1)用Project Wizard产生一个支持Socket而基于对话框的MFC应用程序，其类名为CChatDlg。

2)用资源编辑器在这个对话框上加上一个Listen命令按钮。

接收到的所有信息将在Visual C++集成环境的Output窗口中用TRACE语句输出。在本程序中，Client指网管程序，Server指交换机。

2.2　类 CClientSocket

调用Class Wizard工具生成类CClientSocket，此类衍生自CSocket类，其作用是被用于接收来自网管程序的UDP数据。在此类中定义如下类别成员和虚函数：

```
public:  BOOL m_bFirst;          //自定义的类别成员;
         CChatDlg * pDlg;        //自定义的类别成员
public:  virtual void OnReceive(int nErrorCode);
```

收稿日期：2000-06-05
作者简介：孙金萍(1975-)，女，陕西闫良人，硕士生，主要从事计算机网络及信息安全技术研究.

2.3 类 CServerSocket

调用 ClassWizard 工具生成类 CServerSocket,此类衍生自 CSocket 类,其作用是被用于接收来自交换机的 UDP 数据。在此类中定义如下类别成员和虚函数:

```
public:  CChatDlg * pDlg;
public:  virtual void OnReceive(int nErrorCode);
```

3 处理函数设计

3.1 按钮 Listen 的处理函数 CChatDlg::OnListen

在 CChatDlg 类中的对象按钮 Listen 的处理函数 CChatDlg::OnListen 代码如下:

```
Void CChatDlg::OnListen( )
{  pClientSocket=new CClientSocket(this);
  if(pClientSocket! =NULL){
        if(! pClientSocket->Create(SNMP_SOCKET_PORT,SOCK_DGRAM))
              AfxMessageBox("Can not create ClientSocket!");
        else
              ::EnableWindow(GetDlgItem(IDC_LISTEN)->m_hWnd,FALSE);
  }else  AfxMessageBox("Can not new ClientSocket!");
}
```

注意:SNMP_SOCKET_PORT 是自定义宏,可设为 161。

3.2 虚函数 CClientSocket::OnReceive

```
    void CClientSocket::OnReceive(int nErrorCode)
{    CSocket::OnReceive(nErrorCode);
    unsigned char tmp[MAXTMPSIZE];      //MAXTMPSIZE 是自定义宏,可为 1024;
    int i,RecNum;
    UINT ClientPort;
    CString ClientAddress;
    if(m_bFirst){   m_bFirst=false;
          RecNum=ReceiveFrom(tmp,MAXTMPSIZE,ClientAddress,ClientPort);
          if(RecNum>0){
                TRACE("Received from client,%d bytes:\n",RecNum);
                for(i=0;i<RecNum;i++)  if(i%10==0)TRACE("\n");
                    TRACE("%5d,",tmp[i]);}
                TRACE("\n\n");
                pDlg->CreateServerSocket(ClientAddress,ClientPort);
                pDlg->Send(true,tmp,RecNum);
             }else  AfxMessageBox("Error:fail to Receive from client the first time!");
        }else{  RecNum=Receive(tmp,MAXTMPSIZE);
             if(RecNum>0){  TRACE("Received from client,%d bytes:\n",RecNum);
                  for(i=0;i<RecNum;i++)
                      {if(i%10==0)TRACE("\n");
                      TRACE("%5d,",tmp[i]);}
                  TRACE("\n\n");
                  pDlg->Send(true,tmp,RecNum);
             }else  AfxMessageBox("Error:fail to Receive from client!");
        }
        if(RecNum<=0){  AfxMessageBox("Error:fail to Receive from client!");
```

```
            return;
    } }
```

这段程序的大意是:如果本程序首次收到来自网管程序的 UDP 包,则要记录下它的 Socket 端口号和 IP 地址。这样做的原因是:网管通讯开始时一般是由网管程序首先发出 SNMP 请求包,所以要先响应网管程序;另一目的是由此获得事先未知的网管程序侦听的 Socket 端口号和 IP 地址,然后让 CChatDlg 由此创建 CServerScoket。随后调用 CChatDlg::Send 函数将收到的 UDP 包转发给交换机,并在 Output 窗口中显示出收到的数据。

3.3 函数 CCatDlg::CreateServerSocket

函数 CCatDlg::CreateServerSocket 的作用是在 Client 和 Server 之间创建 Socket,其代码如下:

```
void CChatDlg::Create Serve Socket(CString address,UINT port)
{   m _ ClientAddress=address;
    m _ ClientPort=port;
    pServerSocket=new CServerSocket(this);
    if(pServerSocket! =NULL)
        if(! pServerSocket->Create(m _ ClientPort,SOCK _ DGRAM))
            AfxMessageBox("Can not create ServerSocket!");
    else  AfxMessageBox("Can not new ServerSocket!");
}
```

3.4 函数 CChatDlg::Send

函数 CChatDlg::Send 的作用是在 Client 和 Server 之间"上传下达",其代码如下:

```
void cchatDlg::Send(BOOL ToServer,unsigned char * buf,int buf _ len)
  {if(ToServer){  if(pServerSocket! =NULL){
            if(pServerSocket->SendTo(buf _ len,SNMP _ SOCKET _ PORT,
                  m _ ServerAddress)==SOCKET _ ERROR)
                  AfxMessageBox("Error·fail to send data to server!");
        }
    }else  if(pClientSocket! =NULL)
          if(pClientSocket->SendTo(buf,buf _ len,m _ Clientport,
              m _ ClientAddress)==SOCKET _ ERROR)
              AfxMessabeBox("Error·fail to send data to client!");
}
```

这里,m _ ServerAddress 是交换机的 IP 地址,要事先在 CChatDlg::OnInitDialog 函数或其它地方设定。

3.5 虚函数 CServerSocket::OnReceive

要处理接收到的来自于交换机的 UDP 包,将其中的数据在 Ouput 窗口中显示出来,然后调用 CChatDlg::Send 函数将其载发给网管程序。这在虚函数 CServerSocket::OnReceive 中实现:

```
void CServerSocket::OnReceive(int nErrorCode)
  {CSocket::OnReceive(nErrorCode);
    unsigned char tmp[MAXTMPSIZE];
    int RecNum,i;
    RecNum=Receive(tmp,MAXTMPSIZE);
    if(RecNum>0){
        TRACE("Received from server,%u bytes:\n",RecNum);
        for(i=0;i<RecNum;i++)
            {if(i%10==0)TRACE("\n");
            TRACE("%5d,",tmp[i]);}
    TRACE("\n\n");
```

```
    pDlg->Send(false,tmp,RecNum);
    }else{
        i=GetLastError( );
        TRACE("RecNum=%d,GetLastError( )=%d\n,"RecNum,i);
        AfxMessageBox("Error·fail to Receive from server!");
    }
}
```

4 结论

分别在两台机器上装上本程序和网管程序,将它们连上交换机,先运行本程序,点 Listen 按钮,然后运行网管程序。一般的网管程序运行时,需要设置被管设备的 IP 地址,此时,要将其设为本程序所在机器的 IP 地址,使网管程序将所有的 SNMP 包发给本程序。之后两程序应能正确运行,在 Output 窗口可以看到数据源源不断地显示出来,这是对网管过程的真实记录!当数据量足够后,结束本程序,可以看到网管程序界面上显示出"设备已断开连接!"的提示信息。然后可以将 Output 窗口中的数据拷贝到文本文件中,按照 SNMP 的格式和编码规则进行详细分析,网管协议及有关信息就可由此破译出来。

以上程序在 Visual C++6.0 下编译通过并运行成功,实践效果很好。

参考文献:

[1] Antun Caric, Kristian Toivo. New Generation Network Architecture and Software Design[J]. IEEE Communications, 2000, 38(2): 108 - 114

[2] Daniel O. Awduche. MPLS and Traffic Engineering in IP Network [J]. IEEE Commuhnications, 1999, 37(12): 42 - 47.

[3] Stever Holzner. Visual C++6: In Record Time[M]. California: SYBEX Inc, 1998.

[4] 吴晓文,吴诗奇,李乐民. 无线 ATM 能鹤网多址访问协议与信道动态分配算法研究[J]. 电子学报, 2000, 28(2): 101 - 104.

[5] Andrew S. Tanenbaum. Computer Network(3rd ed)[M]. New Jersey: Prentice-Hall International, Inc, 1997.

[6] Andrew S. Tanenbaum. Distributed Operating Systems[M]. New Jersey: Prentice-Hall International, Inc, 1998.

A method and C++ implementation of listening in network information

SUN Jin-ping, LEI Ying-jie, ZNAO Xue-jun

(Missile Institute, AFEU., Sanyuan 713800, China)

Abstract: The information regarding computer network management is very complicated and its range is quite wide, too. In this case, it must be easy to cause loopholes and the security of the information regarding computer network management is certain to be imperilled. In order to develop network software products, it is usually necessary to investigate the internal data structures of network protocols within several switch boards. A method to perform listening in network and keeping useful information, and how to complete the program using MFC Socket under Visual C++ 6.0 environment is presented in this paper.

Key words: computer network; listening in information; C++; Socket program