广西科学院学报
Journal of Guangxi Academy of Sciences
2006,22(4):298～301
Vol. 22,No. 4 November 2006

内网安全监控系统中插件编程的实现*

Implementation of Plug-ins Programming in the Intranet Security Monitoring System

黄国石,李陶深**,严 毅,蔡世平,曾 亮,黄顶源
HUANG Guo-shi, LI Tao-shen, YAN Yi, CAI Shi-ping, ZENG Liang, HUANG Ding-yuan

(广西大学计算机与电子信息学院,广西南宁 530004)
(School of Computer, Electronics and Information, Guangxi University, Nanning, Guangxi, 530004, China)

摘要:介绍在内网安全监控系统中监控部件的插件编程方法,并阐述具体的实现技术然后通过应用实例说明这些方法和编程技术在内网安全监控系统地研制与开发中是有效的。

关键词:内网 安全 DLL Delphi 插件
中图法分类号:TP393.07 **文献标识码**:A **文章编号**:1002-7378(2006)04-0298-04

Abstract: The tips for programming the plug-ins in the security monitoring system of intranet are introduced. These tips are explained in an application.

Key words: intranet, security, Dll, Delphi, plug-in

内网安全监控系统是通过直观有效的安全策略设置,对内网中的各种行为进行监控和管理,通过完善的日志记录各种敏感行为的痕迹,为管理和审核提供途径。内网安全监控系统包括网络监控、外设监控、系统监控三大部分,实现了20个监控部件,各部件之间不存在直接联系,均以插件形式存在。每个插件都是由客户端主程序来加载并以多线程方式运行。这样就使得系统的可拆分成为可能,因为在以后的使用中,人们可以随意增加、修改或者去除单个监控部件,极大地方便了各个项目开发人员的协同开发。本文介绍内网安全监控系统中监控部件的插件编程方法和技术要点,并以客户端主程序的设计与开发作为实例,说明插件的编程实现。

1 插件编程的方法和技术要点

利用DLL动态链接库的形式实现插件结构编程。插件结构编程需要一个插件容器(也就是我们的客户端Client)来控制各DLL的运行情况,将划分好的每个子系统安排到一个DLL库文件中。对每个DLL程序需要为容器预留接口函数,一般接口函数包括:启动调用DLL库的函数、关闭DLL库的函数。通过接口函数,插件容器可以向DLL模块传递参数实现动态控制。

1.1 程序接口设计及插件调用方式

插件容器Client使用一个独立的工程创建,Client主窗口的作用相当于MDI程序中的MDI容器窗体,Client将显式调用DLL中的接口函数。每个插件程序独立使用各自的工程,与普通工程不同的是,DLL工程创建的是DLL Wizard,相应编译生成的文件是以DLL为后缀。在插件编程中,我们统一预留三个接口函数,它们分别是:(1)SynHandle。该函数将应用程序的句柄、主窗体句柄、DLL句柄传递给插件DLL,DLL程序将动态创建DLL的实例。应用程序句柄用于在需要用到窗体的DLL程序中,让在DLL中创建的窗体挂接到应用程序中;主窗体句柄用于DLL插件程序向客户端发送监控日志;DLL句柄用于在需要的DLL插件程序中中止自身插件线程。初次调用一个DLL插件时使用此函

收稿日期:2006-07-17
作者简介:李陶深(1957-),男,广西邕宁人,教授,主要从事网络安全、网络路由、分布式数据库方面的研究。
*本文得到广西留学回国人员科学基金项目(桂科回0342001)和广西电子信息应用项目(桂电办2004-17号)联合资助。
**通讯联系人。

数初始化。(2)Start。该函数将以参数形式向 DLL 插件程序传递策略。该函数可能被多次调用,以实现 DLL 插件程序在运行时动态改变监控测路。(3)Stop。该函数用于指示插件程序准备停止工作,处理释放资源等工作。

DLL 窗体程序无法直接执行,需要有一个插件容器来调用。因此,我们需要先实现客户端程序 Client,然后将 Client.exe 保存在一个固定的目录中。对每个 DLL 工程做如下设置:(1)打开 DLL 工程;(2)选择菜单 Run Parameters;(3)在弹出的窗口中浏览到我们的容器 Client.exe。这样在调试 DLL 程序时将会自动调用 Client 程序,利用 Client 中预留的调用接口调试 DLL 程序。

在 Delphi 中需要为接口函数声明函数类型,然后实例化函数类型的实例。该实例实际是一个指向函数的指针,通过指针可以访问到函数并传递参数、获取返回值。在单元文件 Data_Unit.pas 的 Interface 部分加入函数类的申明:

```
SynHandleProc=Procedure(FrmHandle:
THandle;DLLHandle:THandle;AppHandle:
THandle);stdcall;
StartFunc=Function(strPolicy:PChar):
Boolean;stdcall;
StopProc=Procedure();stdcall;
```

Stdcall 是参数调用方式,必须和插件中的接口函数的参数调用方式一致;为避免不必要的麻烦,应放弃使用 Delphi 的字符串类型 string,而使用标准的 C 格式字符串(也称零结束字符串)。C 格式字符串在 Delphi 中的类型为 PChar,可以用来作为要传送的监控策略的参数类型。

1.2 插件程序的实现

DLL 程序的设计方式和普通 WINAPP 没有很大的区别,只是所有的窗口都是作为一种特殊的"资源"保存在 DLL 库中,需要手动调用,而不像 WINAPP 中会有工程自动创建。声明接口函数的方法很简单,具体步骤有:(1)在 Unit 的 Implementation 部分中声明函数;(2)在函数声明语句的尾部加上 stdcall 标记;(3)在工程代码(Project View Source)的 begin 语句之前,用 exports 语句声明函数接口:

```
Exports
  SynHandle Name 'SynHandle',
  Start Name 'Start',
  Stop Name 'Stop';
```

1.3 接口函数的引入

调用 DLL 库中的函数有显式和隐式两种方式,显式调用比较灵活,因此本系统使用显示调用。在 Delphi 中需要为接口函数申明函数类型,然后实例化函数类型的实例,该实例实际是一个指向函数的指针,通过指针可以访问到函数并传递参数、获取返回值。在单元文件的 Interface 部分加入函数类的申明:

```
SynHandleProc=Procedure(FrmHandle:
THandle;DLLHandle:THandle;AppHandle:
THandle);stdcall;
StartFunc=Function(strPolicy:PChar):
Boolean;stdcall;
StopProc=Procedure();stdcall;
```

1.4 载入 DLL 库文件

通过调用 API 函数 LoadLibrary,可以将 DLL 库载入到内存中。LoadLibrary 的参数是 DLL 文件的地址路径,如果载入成功会返回一个 CARDINAL 类型的变量作为 DLL 库的句柄;如果目标文件不存在或其他原因导致载入 DLL 文件失败会返回一个 0。主要代码如下:

```
Plugins[PluginID].DLLHandle:=
GetModuleHandle(PChar(Plugins[PluginID].
ExePath));
If Plugins[PluginID].DLLHandle = 0 Then
  Plugins[PluginID].DLLHandle :=
  LoadLibrary(PChar(Plugins[PluginID].
ExePath));
```

1.5 实例化接口函数

获得接口函数指针的函数为 GetProcAddress(库文件句柄,函数名称),如果找到函数则会返回该函数的指针,如果失败则返回 NIL。

使用上文定义的函数类型定义函数指针变量,然后使用@操作符获得函数地址,这样就可以使用指针变量访问函数。主要代码如下:

```
Plugins[PluginID].SynHandle:=GetProcAddress(
  Plugins[PluginID].DLLHandle,'SynHandle');
Plugins[PluginID].Start := GetProcAddress(
  Plugins[PluginID].DLLHandle,'Start');
Plugins[PluginID].Stop := GetProcAddress(
  Plugins[PluginID].DLLHandle,'Stop');
```

2 内网安全监控系统中插件编程的实现

2.1 插件的工作流程

每个插件将以插件名.DLL的形式放置在本目录或者Plugin目录下。其工作流程如下：

(1)客户端主程序Client启动后，将会处理身份验证、监控策略同步、监控日志上传、自身保护等一系列工作。在完成了这些初始化工作后，Client将在本目录及Plugin子目录搜索所有DLL文件，并使用其文件名作为插件名，为插件建立一个记录对象并添加到插件列表中。

(2)在插件线程内部，将加载插件DLL，并获得SynHandle、Start、Stop这三个接口函数的地址。完全按照接口规范开发的插件将会被成功加载，并取得三个接口函数的地址。

(3)加载插件成功后，将首先调用SynHandle函数把Client主窗体句柄、DLL句柄、应用程序句柄作为参数送给插件。如果插件需要建立窗体，就将在这个时候建立。

(4)调用Start接口函数时，要把监控策略作为参数送给插件，插件内部分析策略并即时更新监控的方式。可多次调用该函数以实现动态改变监控方式。

(5)至此，插件开始工作。如果在工作过程中产生日志信息，将通过消息发送给主程序的监控日志模块进行处理。

(6)客户端需要停止工作的时候，可以调用插件的Stop方式，通知插件处理善后工作，随即终止插件线程。

2.2 插件的结构化管理

为了方便管理插件，我们设计插件结构，每个插件对应一个结构体，所有插件一起组成结构体数组。结构体的声明如下。

```
PPlugin = ^ TPlugin;
TPlugin = Record
  Index: Integer; // 编号
  FrmHandle: THandle; // 主窗体句柄
  DLLHandle: THandle; // DLL 句柄
  AppHandle: THandle; // 应用程序句柄
  Name: String; // 插件名字
  ExePath: String; // 插件路径
  Policy: String; // 插件策略
  SynHandle: SynHandleProc;//插件的 SynHandle 接口
  Start: StartFunc; // 插件的 Start 接口
  Stop: StopProc; // 插件的 Stop 接口
  Thread: TPluginThread; // 插件的线程对象
End;
```

2.3 以多线程方式工作

在一个独立的单元文件PluginThread_Unit.pas中，声明一个插件线程类：

```
TPluginThread = Class(TThread)
  Private
    { Private declarations }
    PluginID: Integer;
    DbMsg: String;
  Protected
    Procedure Execute; Override;
    Procedure Start();
    Procedure Stop();
    Procedure DbOut();
  Public
    Constructor Create(pid: Integer);
    Destructor Destroy; Override;
  End;
```

该类将会为每一个插件实例化一个线程对象，该插件的所有工作将在其独立的线程内完成。

2.4 接收插件返回日志的方法

调用接口函数SynHandle的时候，给插件送的3个参数其中一个就是主窗体句柄，插件程序可以向该窗口句柄发送WM_COPYDATA消息，主窗体监控日志模块将会进行处理。Delphi版本的发送日志例子程序如下。

```
Procedure SendLog(strLog: String);
Var
  tCD : TCOPYDATASTRUCT;
Begin
  Try
    strLog:=PluginName+'$"'+
FormatDateTime ('yyyy- mm-dd hh:mm:ss
zzz', Now()) + '"' + strLog;
    FrmTest.mmo1.Lines.Add('发送日志:'
+ strLog);
    tCD.cbData := Length(strLog) + 1;
    GetMem(tCD.lpData, tCD.cbData);
    StrCopy(tCD.lpData, PChar(strLog));
      SendMessage (g_hFrm, WM_
```

```
    COPYDATA, g _ hDll, Cardinal(@tCD));
    Except
      FrmTest. mmo1. Lines. Add('发送日志失败');
    End;
  End;
```

客户端主窗体接收消息的代码如下:

```
{所有监控模块,使用 WM _ COPYDATA 消息向本程序发送日志数据}
Procedure On _ WM _ COPYDATA _ MessageArrival (Var copyData: TWmCopyData);
Message WM _ COPYDATA;
{接收监控模块传递过来的监控日志}
Procedure TFrmMain. On _ WM _ COPYDATA _ MessageArrival (Var copyData: TWmCopyData);
Var
  tmpLog          :String;
  tmpPluginName   :String;
Begin
    tmpLog := StrPas (copyData. CopyDataStruct^. lpData);
  tmpPluginName := Copy(tmpLog, 1, Pos('$', tmpLog) - 1);
  mmo1. Lines. Add(tmpLog);
  WriteLog(tmpPluginName, tmpLog);
End;
```

3 应用实例

客户端启动而且经过身份验证后获得了自己的主机 ID 这一身份标识(见图 1)。

点击加载插件,程序将搜索本目录的插件(见图 2)。如果 SynHandle、Start、Stop 三个函数全部为 0,表明还没有加载。

以 CdRom(光驱监控)插件和 Process(进程监控)插件作为例子,启动这两个插件。如图 3 所示。可以看到,CdRom(光驱监控)插件和 Process(进程监控)插件的 SynHandle、Start、Stop 三个函数已经全部获得具体数值,表明成功加载了插件 DLL 并获得了接口函数地址,它们都已经分别在系统为它们开辟的线程中工作。

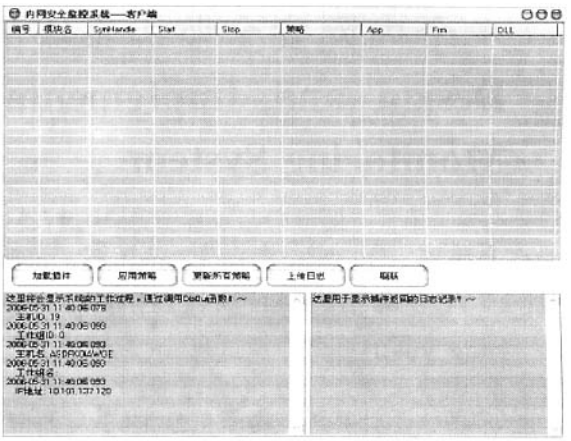

图 1 系统客户端的主界面

图 2 加载插件

图 3 启动插件

(责任编辑:邓大玉 凌汉恩)