远程线程注入技术在监控系统中的应用

王 峥,娄渊胜
(河海大学 计算机及信息工程学院,江苏 南京 210098)

摘 要:监控系统要求具有实时性和隐藏性,远程线程注入技术能实现在Windows系统下进程的隐藏。将监控程序编译成动态链接库(DLL)文件,采用远程线程注入技术注入到系统进程运行,能有效地提高监控系统的安全性能。本文介绍了远程线程注入技术的原理,分析了基于远程线程注入的监控系统的关键技术和实现方法,通过设置定时器的方法解决了系统实时性需求,通过给出的远程线程注入技术解决了隐藏性需求。最后分析采用两级监控和应对安全检测技术来提高监控系统的安全性。

关键词:远程线程注入;监控系统;进程隐藏;动态链接库
中图分类号:TP277 **文献标识码**:A **文章编号**:1673-629X(2010)03-0207-04

Application of Remote-Thread Injection Technique on Monitor System

WANG Zheng, LOU Yuan-sheng
(College of Computer & Information Engineering, Hohai University, Nanjing 210098, China)

Abstract: The monitor system demands a characteristic of real time and hiding. The process can be hidden by remote-thread injection technique in Windows system. The monitor program is compiled to DLL and injected into system process to run. In this way, the safety of monitor system can be enhanced effectively. The theory of remote-thread injection technique is presented in this paper. The main technique and implementation method of monitor system based on remote-thread injection technique is analyzed, by setting timer to solve the demands of system's real time, through the remote-thread injection technique to solve the demands of hiding. At last, the two-stage monitor technique and the responding to safety detection technique are discussed to improving the safety of monitor system.

Key words: remote-thread injection; monitor system; hidden process; DLL

0 引 言

监控系统必须要保证系统的正常运行,并最大程度地减少系统资源负担,同时要防止被用户结束,较为有效的方法就是把进程隐藏。进程隐藏的实现技术主要包括修改动态链接库、应用程序的接口挂钩、远程线程注入等方式[1]。应用程序的接口挂钩技术是通过API HOOK编程方法来截获系统遍历进程函数并替换来达到任意进程的隐藏[2]。修改动态链接库技术,由于DLL文件不能独立运行,因此和远程线程注入技术结合运用。远程线程注入技术是指在指定远程进程中创建线程,通过注入到指定进程的内存地址空间。通过远程线程注入技术将所需的线程代码注入的目标进程中运行,不创建独立的进程,很难被发现,但常规的远程线程注入技术难以避过安全检测技术的检测[3]。远程线程注入被广泛地运用到电脑病毒传播中,然而也由于其不易被使用者删除的特点可以协助管理人员安装监控、计费等系统管理程序而不被使用者破坏,达到安全管理的目的[4]。

文中讨论了基于远程线程注入的监控系统实现的关键技术和实现方法,分析了在DLL中定时监控功能的实现,提出采用两级监控的方法来提高监控系统的功能。

1 远程线程注入技术

1.1 远程线程技术原理

动态连接库(Dynamic Link Library,DLL)是一个可以被其他应用程序共享的程序模块,其中封装了一些过程、函数和资源[5,6]。一般情况下,DLL文件不能直接运行,只能由执行文件来调用。把需要的功能编

收稿日期:2009-06-22;**修回日期**:2009-09-11
基金项目:国家高技术(863)计划项目(2007AA01Z179)
作者简介:王 峥(1982-),女,河北行唐人,硕士研究生,研究方向为应用集成;娄渊胜,博士,副教授,研究方向为软件体系结构、分布式应用集成。

写成函数装封到 DLL 文件中,然后再另外写一个执行文件来启动它。远程线程注入技术通过远程插入线程调用动态连接库文件实现。一般情况下,每个进程都有自己的私有空间,理论上,别的进程不允许对这个私有空间进行操作,但可以利用一些方法进入这个空间并进行操作,将编写的代码写入正在运行的进程中,即采用远程线程注入方法。

1.2 实现远程线程技术所用 API 函数

线程注入过程的原理是采用 Win32 API 函数 WriteProcessMemory 和 CreateRemoteThread,将 DLL 的代码直接复制到远程进程空间,用 CreateRemoteThread 执行它。

远程线程注入是通过 CreateRemoteThread 函数建立一个远程线程,调用 LoadLibrary 函数来加载指定的 DLL。CreateRemoteThread 函数的用法如下:

```
function CreateRemoteThread(
    hProcess: THandle;
    //远程进程的句柄
    lpThreadAttributes: Pointer;
    //线程安全描述字
    dwStackSize: DWORD;
    //线程栈大小,以字节表示
    lpStartAddress: TFNThreadStartRoutine;
    //一个 TFNThreadStartRoutine 类型的指针,指向在远程进程中执行的函数地址
    lpParameter: Pointer; //传入参数的指针
    dwCreationFlags: DWORD;
    //创建线程的其它标志
    var lpThreadId: DWORD
    //线程身份标志,如果为 0,则不返回
): THandle;
```

在 Windows 系统下,每个进程都拥有自己的地址空间,各个进程之间都是相互独立的。需要在远程进程的内存空间里申请一块内存空间,写入需要注入的 DLL 的路径。需要用到的 API 函数有:

(1)OpenProcess():打开目标进程,得到目标进程的操作权限。

(2)VirtualAllocEx():用于在目标进程内存空间中申请内存空间以写入 DLL 的文件名。

(3)WriteProcessMemory():往申请到的空间中写入 DLL 的文件名。

1.3 远程线程注入技术的实现流程

远程线程注入技术的主要步骤如下:

(1)调整权限,使程序可以访问其他进程的内存空间(EnableDebugPriv)。

(2)得到远程进程的 HANDLE(OpenProcess)。

(3)在远程进程中为要注入的数据分配内存(VirtualAllocEx)。

(4)将注入 DLL 复制到本进程,实现函数 DLL 装载过程(MapInjectFile)。

(5)把 DLL 资源复制到分配的内存中(WriteProcessMemory)。

(6)用 CreateRemoteThread 启动远程的线程[7]。

2 基于远程线程注入的监控系统

2.1 监控系统的特性和构成

监控系统种类和应用很广,如上网计费、机房管理等。不同的领域功能不同,但一般要求具有实时性和隐藏性(健壮性)两个特点。实时性可由定时器实现,通过设置定时器的时间间隔就能满足一般性的要求。而隐藏性则要求系统隐蔽,不容易被发现,即使被发现也要能防止被结束[8,9]。若将监控程序编写成 DLL,通过远程线程注入技术,在注入的系统进程中运行,能够满足上述要求。

基于远程线程注入的监控系统由启动程序和监控程序两个部分构成。启动程序可以随开机自动运行,并负责完成将监控程序 DLL 模块注入到目标进程中。注入完成后,监控线程开始在目标进程中执行,实现监控功能。监控线程和目标进程具有相同的生命期。监控系统运行的过程如图 1 所示。注入的执行过程很快,启动程序在开机的瞬间就运行结束。

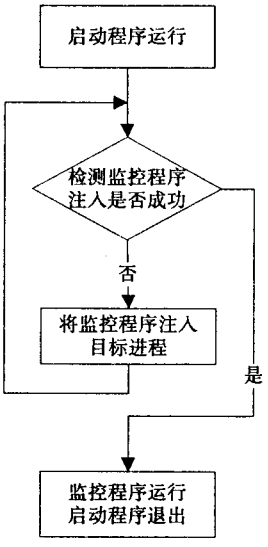

图 1 机房监控系统运行过程

2.2 定时监控功能的实现

每一 DLL 必须有一个入口点,DLLMain 函数是 DLL 模块的默认入口点。DLLMain 负责初始化和结束工作。DLLMain 函数在将 DLL 模块加载到进程时、DLL 模块与进程分离时被调用。Delphi 语言中,DLL

入口由 DLLProc 传入。

在 DLL 文件中编写一个线程执行体函数 ThreadProc 实现监控的功能,比如监控某个程序是否在运行,如果没有则重新启动这个程序。定时功能可调用 API 函数 SetTimer 实现,比用组件精度高。在 DLL 注入到目标进程后,将监控程序创建一个线程,启动过程如图 2 所示。创建和结束线程分别用到 CreateThread 函数和 TerminateThread 函数。

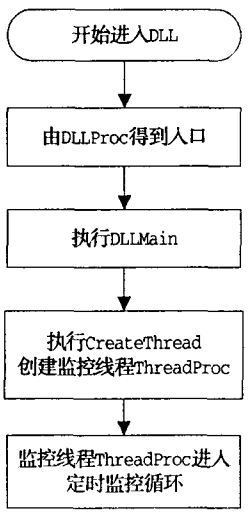

图 2　监控线程启动过程

CreateThread 的函数定义:

```
function CreateThread(
    lpThreadAttributes: Pointer;
    dwStackSize: DWORD;
    lpStartAddress: TFNThreadStartRoutine;
    lpParameter: Pointer;
    dwCreationFlags: DWORD;
    var lpThreadId: DWORD
): THandle; stdcall;
```

其中 lpStartAddress, lpParameter, lpThreadId 三个参数是必需的。lpStartAddress 参数指向的是线程执行体 ThreadProc 的开始地址; lpParameter 指针类型,线程的传入参数,给线程执行体 ThreadProc 传递数据; lpThreadId 返回创建线程 ID,这是控制线程必需的。实现监控功能的 DLL 文件的主要代码如下:

```
procedure ThreadProc(p : pointer); stdcall;
//线程执行体,实现监控功能
begin
SetTimer(0, 1, 10000, nil);
//设置定时器的时间间隔为 10000 毫秒
while GetMessage(Msg,0,0,0) do
//从消息队列得到消息
begin
if Msg.message= WM_TIMER then
// WM_TIMER 定时器消息
try
if not FindProcess('watch.exe') then
winexec(pchar('watch.exe'),SW_Show);
//此处为定时处理的内容,功能是检测'watch.exe 进程是否存在,如果不存在启动
except;
end
else
begin
TranslateMessage(Msg);
//用于将 virtul-key 消息翻译为字符消息
DispatchMessage(Msg);
//派送消息到窗口过程
end;
end;
end;
procedure DllMain(dwReason : DWORD);
//DLL 入口程序
var
dwThreadID : DWORD;
begin
case dwReason of
DLL_PROCESS_ATTACH : //进程被调用
begin
hThreadHandle := CreateThread(nil, 0, @ThreadProc, nil,
0, dwThreadID);
// 将 ThreadProc 创建一个线程
end;
DLL-PROCESS-DETACH : //进程被停止
begin
SendMessage(hWnd, WM-THREADEXIT, 0, 0);
//发送线程结束消息
if (hThreadHandle <> 0) then
begin
TerminateThread(hThreadHandle, 0);
//撤销该线程
end;
end;
DLL_THREAD_ATTACH : //线程被调用
begin
end;
DLL_THREAD_DETACH : //线程被停止
begin
end;
end;
end;
begin
DLLProc := @DLLMain;//DLL 入口
DLLMain(DLL_PROCESS_ATTACH);
```

//调用 DLLMain,参数为 DLL_PROCESS_ATTACH
end.

2.3 相关问题分析

相关问题分析如下:

(1)两级监控。

注入的目标进程一般选择系统进程如 csrss、system、Winlogon 等,这样安全性更高,即便注入的 DLL 模块被发现,系统进程也不允许被关闭。但有时要选择 explorer 进程,explorer 是一个重要用户进程,用于管理 Windows 图形壳,包括开始菜单、任务栏、桌面和文件管理。删除该进程会导致 Windows 图形界面无法使用。注入 explorer 进程的优点是监控程序可以启动具有图形界面的程序;缺点是 explorer 进程被结束后,可以重新在任务管理器中运行 explorer,系统恢复正常,但是先前注入的监控程序就不存在了。解决的办法可以采用两级监控:第一级监控程序模块注入到系统进程,负责监控第二级监控程序模块是否已经注入到 explorer 进程,若没有则将其注入到 explorer 进程;第二级监控程序负责监控具体任务。采用两级监控可以使监控系统更安全、功能更强。

(2)应对安全检测技术。

远程线程注入过程需要在目标进程中创建远程线程,该行为可以被某些安全检测软件检测到。若不能通过许可,注入就被禁止。利用异步过程调用(Asyncroneus Procedure Call,APC)机制实现远程线程注入,该技术未对内核对象等系统敏感信息进行修改(未挂钩系统服务函数、修改服务函数执行路径),因此其能有效对抗系统一致性检测、EPA 执行路径分析、挂钩检测等检测技术。相比于常规的远程线程注入,该技术未采用跨进程创建线程的常规方案,从而能有效地避开安全测试软件的常规行为检测(针对跨进程创建线程的可疑行为的检测)。

3 结束语

将监控程序编译成 DLL 文件,通过远程线程注入技术将其注入到系统进程,可以有效地对监控系统起到隐藏和保护的作用。文中研究方法应用到了学校机房管理中,用来监控机房管理客户端程序的运行。通过注册表自动启动完成注入,配合硬盘保护卡使用,取得了良好的效果。

参考文献:

[1] 王建华,张焕生,侯丽坤. Windows 核心编程[M]. 北京:机械工业出版社,2001.

[2] 肖道举,左 佳,陈晓苏. 进程隐藏的相关问题研究[J]. 微处理机,2008(2):78-80.

[3] 何 志,范明钰,罗彬杰. 基于远程线程注入的进程隐藏技术研究[J]. 计算机应用,2008,28(6):92-94.

[4] 李元良,黄 强. 远程线程注入技术的实现[J]. 矿业研究与开发,2006,26(3):77-78.

[5] 曹 蕾,李光明,傅 蓉,等. Delphi 7 程序设计与上机指导[M]. 北京:冶金工业出版社,2003.

[6] 龙启明,刘 斌,程 捷. Delphi 高级编程范例[M]. 北京:清华大学出版社,2003.

[7] 崔 甲,李毅超,梁 晓,等. 基于 Windows 平台的键盘记录技术的研究[J]. 网络安全技术与应用,2007(9):79-81.

[8] Cicotti P, Taufer M, Chien A A. DGMonitor: A Performance Monitoring Tool for Sandbox-Based Desktop Grid Platforms [J]. The Journal of Supercomputing, 2005, 34:113-133.

[9] Rashidzadeh R, Ahmadi M, Miller W C. On-chip measurement of waveforms in mixed-signal circuits using a segmented subsampling technique[J]. Analog Integr Circ Sig Process, 2007, 50:105-113.

(上接第 206 页)

参考文献:

[1] Bajcinca N. Design of robust PID controllers using decoupling at singular frequencies[J]. Automatica, 2006, 42: 1943-1949.

[2] Saeki M. Properties of Stabilizing PID Gain Set in Parameter Space[J]. IEEE Trans on Automatic Control, 2007, 52(9): 1710-1715.

[3] Ho M T, Datta A, Bhattacharyya S P. A linear Programming Characterization of All Stabilizing PID Controllers[C]//Proc of American Control Conf. Albuquerque: [s. n.], 1997: 3922-3928.

[4] Keel L H, Bhattacharyya S P. PID Controller Synthesis Free of Analytical Models[C]//Proceedings of the 16th IFAC World Congress. Prague, Czech Republic: [s. n.], 2005.

[5] Silva G J, Datta A, Bhattacharyya S P. New Results on the Synthesis of PID Controllers[J]. IEEE Trans on Automatic Control, 2002, 47(2): 241-252.

[6] Soylemeza M T, Munrob N, Baki H. Fast calculation of stabilizing PID controllers[J]. Automatica, 2003, 39: 121-126.

[7] Bajcinca N, Hulin T. Menge aller robust stabilisierenden PID-Regler: Methodik und Software (Teil I)[J]. Methoden, 2005, 53(11): 556-564.

[8] Ho Ming-Tzu, Datta A, Bhattacharyya S P. Generalizations of the Hermite-Biehler theorem[J]. Linear Algebra and its Applications, 1999, 302-303: 135-153.

[9] 吴 麒. 自动控制原理[M]. 北京:清华大学出版社,1996.