doi :10.3969/j.issn.1671-1122.2009.05.015

Windows 环境木马进程隐藏技术研究

卢立蕾，文伟平

（北京大学软件与微电子学院信息安全系，北京 102600）

摘 要：本文介绍了在Windows环境下特洛伊木马常用的进程隐藏技术，结合实际，详细分析了利用系统服务方式、动态嵌入方式、SSDT Hook和DKOM技术实现进程隐藏的基本原理，对如何防御和检测木马具有一定的参考意义。

关键词：木马；进程；隐藏；系统服务；动态嵌入；SSDT Hook；DKOM

中图分类号：TP309.5 文献标识码：A

Research of Process Concealment Technology Used by Trojan Horse in Windows Environment

LU Li-lei, WEN Wei-ping

(Department of Information Security, SSM, Peking University, Beijing 102600, *China)*

Abstract: The paper introduces the common process concealment technologies used by the Trojan horse in the Windows environment. The technologies such as System Service, Dynamic Embedding, SSDT Hook and DKOM method are analyzed in detail on the basis of practice. This will give us some reference about how to defend and check Trojan horse.

Key words: Process; Concealment; Trojan horse

0 引言

随着计算机技术的迅速发展和网络应用的日益普及，恶意代码出现的频率越来越高。目前，隐藏功能已成为很多恶意代码的一个重要特征，分析恶意代码使用的隐藏手段和相关的技术，可以让我们更好地提高防范意识，并据此采取相应的防御和检测措施。

特洛伊木马（简称木马）是具有较强隐藏功能的一类恶意代码。它通常伪装成合法程序或隐藏在合法程序中，通过执行恶意代码，为入侵者提供非授权访问系统的后门[1]。本文首先简单介绍了木马实现的常用技术，接着重点分析了Windows 环境下木马实现进程隐藏经常使用的技术。

1 木马实现常用技术

1.1 注册表修改

很多木马在实现时都会修改注册表，修改的目的通常是：借助于注册表来实现启动或隐藏。

在注册表的多个项目中都包含启动项。启动项的数值可以设置为用户指定的、伴随系统启动或服务启动而自动运行的程序的绝对路径。因此，将启动项的数值设置为木马程序的路径，就可以实现木马的启动。但是，使用注册表编辑器能够很容易地将注册表项的数值删除，为此，很多木马在使用时都加入了时间控制程序段，以监视注册表中相应的数据是否存在，一旦发现被删除则会立即重新写入。

1.2 反向连接

反向连接就是被攻击者主动连接攻击者的过程，通常是运行木马的服务端进程主动连接客户端控制进程并进行通信的过程，也称为反弹技术[2] [3]。其基本原理是利用防火墙对由内到外的连接疏于防范的弱点来实现由内到外的主动连接。

NameLess 木马实现时也用到了反向连接，它使用嗅探原理来取得控制端的 IP 地址，然后实现反向连接。

1.3 端口复用

端口复用是指在一个端口上建立了多个连接，而不是在一个端口上面开放了多个服务。一种常用的端口复用技术，就是利用系统实际存在的系统的合法端口进行通讯和控制，如 21、23、80 等，使一个端口除了完成正常的功能外，还可用于木马通信，而不是使用一个新开的端口号[2]。这样的好处就是非常隐蔽，不用自己开端口也不会暴露自己的访问，因为通讯本身就是系统的正常访问。

1.4 文件隐藏

文件隐藏可以通过修改系统呈现给用户的文件列表，使用户或检测软件无法发现事实上存在的文件，从而达到隐藏目的[4]。目前比较常见的文件隐藏技术通常使用 Rootkit 来实现，这和后面要介绍的使用 Rootkit 实现的进程隐藏的基本原理类似，这儿不做介绍。

以上木马实现技术通常不是孤立存在的，如端口复用和反向连接技术通常关联在一起。另外，比较成功的木马其实现也往往同时使用多种技术。当然，木马实现的常用技术除了上面列举的几种之外，还包括非常重要的进程隐藏技术，下面重点就本地系统 Windows 环境下木马常用的进程隐藏技术加以讨论。

2 木马进程隐藏常用技术

2.1 概述

隐藏是木马的一个首要特征。从木马的隐藏方式来看，通常可分为本地隐藏、通信隐藏和协作隐藏。本地隐藏是指为防止本地用户或系统管理人员发现而采取的隐藏手段，主要包括文件隐藏、进程隐藏、网络连接隐藏、内核模块隐藏等[5]。

进程隐藏是木马常用的一种隐藏手段。进程隐藏，就是通过某种手段，使用户不能发现当前运行着的某个特定进程。从隐藏的程度来看，进程隐藏又分为两种：假隐藏和真隐藏。假隐藏，是指某个程序运行时，与它对应的进程仍然存在，只不过是消失在任务管理器的进程列表里；而真隐藏，则是让程序彻底消失，不再以一个进程或者服务的方式运行[6]。

下面介绍的常用进程隐藏技术包括系统服务方式、动态嵌入方式和Rootkit方式。

2.2 系统服务方式

在较早的Windows 9x系统 中，利用RegisterServiceProcess方法，可以把任何程序注册为一个系统服务，它只是以服务的方式在后台工作，而不出现在任务管理器的列表中[6]。因此在Windows 9x系统中，使用这种方式可以很方便地实现进程隐藏。事实上，这种隐藏只是一种“假隐藏”，因为注册为系统服务的程序在运行时还有独立的进程存在，只是不在任务管理器列表中显示而已。对WINNT类的操作系统，如Windows NT/2000/XP等，这种方法已不再有效[7]。

通过替换系统服务进程同样可以实现木马进程的隐藏，这时木马是以合法服务的形式运行，从而很好地实现了自身进程的隐藏。NameLess木马就是一个很好的例子。NameLess木马的实质是编写了一个Windows形式的服务，该服务通过系统提供的Rundll32.exe程序来进行安装，安装时替换原有的系统服务，通过修改现有服务组里的服务、把它的ServiceDll指向自己来实现的。

2.3 动态嵌入方式

动态链接库(Dynamic Link Library，简称DLL) 是Windows操作系统的基础，Windows API中的所有函数都包含在DLL中。DLL不能独立运行，一般都是由进程加载并调用[8]。把DLL加载到进程中将具有很好的隐蔽性：(1) DLL被映射到宿主进程的地址空间中，它能够共享宿主进程的资源，并根据宿主进程在目标主机的级别非法访问相应的系统资源；(2) DLL没有独立的进程地址空间，能够达到隐蔽自身的目的。因此，如果我们编写了一个DLL程序，并且把它加载到别的进程(如explorer.exe)，那么在任务管理器或检测软件中只会显示别的进程名(explorer.exe)，而不会显示我们编写的DLL，这样也就实现了DLL的隐藏，从而达到侵入目标系统的目的。

在Windows操作系统中，每个进程都有自己的内存空间，通常情况下不允许其他进程对某个进程的内存地址空间进行操作，但是使用动态嵌入方式可以进入并操作另一个进程的内存地址空间，把代码嵌入到该进程中。动态嵌入技术有多种，最常见的有窗口Hook、挂接API以及远程线程插入等[8]。大多数DLL木马都采用远程线程插入技术把自己挂在一个正常的系统进程中。

远程线程插入[9]技术指某一程序在运行时不创建自己的进程，而是通过创建远程线程的方式，将自身作为一个独立的线程运行另一个进程的内存地址空间中[10]。很多木马使用这种方式实现进程隐藏。创建远程线程的函数有7个参数：

```
HANDLE CreateRemoteThread(
        HANDLE hProcess,
        LPSECURITY_ATTRIBUTES lpThreadAttributes,
        SIZE_T dwStackSize,
        LPTHREAD_START_ROUTINE lpStartAddress,
        LPVOID lpParameter,
        DWORD dwCreationFlags,
        LPDWORD lpThreadId
);
```

第1个参数是要注入线程的那个宿主进程的句柄。为了得到宿主进程的句柄，需要使用宿主进程的PID来调用OpenProcess函数。宿主进程的PID可以使用Windows中的任务管理器Taskmgr.exe来查找，也可以编程查找。第2个参数和第7个参数设置为NULL，第3个参数和第6个参数设置为0。

剩下的第4和第5个参数，相对麻烦些。第4个参数为宿主进程中的LoadLibrary的地址。这个地址必须存在于宿主进程中，这只有当导出LoadLibrary的kernel32.dll被装入宿主进程时才起作用。为了得到LoadLibrary的地址，可以调用GetProcAddress函数：

```
GetProcAddress(GetModuleHandle(TEXT("Kernel32")),
"LoadLibraryA")。
```

以上调用会获得正在注入的进程中LoadLibrary的地址，这是假定kernel32.dll在目标进程中有相同的基地址的前提下，实际情况也往往如此。因为将DLL装入内存时，重新定位DLL会耗费系统时间，而微软尽量避免这种额外的开销。GetProcAddress返回类型为THREAD_START_ROUTINE，因此可以把CreateRemoteThread函数的第4个参数设置为GetProcAddress。

第5个参数是要传给LoadLibrary的参数的内存地址，是要注入的DLL的全路径。Windows操作系统提供了解决此问题的两个函数。调用VirtualAllocEx，能够分配宿主进程中的内存。接着使用从VirtualAllocEx函数得到的地址来调用WriteProcessMemory，从而得到在宿主进程中调用LoadLibrary时要用到的DLL的路径名，按照前面的方式，就可以实现远程线程插入了。

将一个DLL或线程注入到另一个目标进程是访问目标进程地址空间的常用方法。

2.4 Rootkit方式

Intel CPU有4个特权级别:Ring 0,Ring 1,Ring 2,Ring 3。Windows只使用了其中的Ring 0和 Ring 3两个级别。操作系

统分为内核和外壳两部分：内核运行在Ring 0级，通常称为核心态（或内核态），用于实现最底层的管理功能，在内核态可以访问系统数据和硬件，包括处理机调度、内存管理、设备管理、文件管理等;外壳运行在Ring 3级，通常称为用户态，是基于内核提供的交互功能而存在的界面，它负责指令传递和解释。通常情况下，用户态的应用程序没有权限访问核心态的地址空间。

Rootkit是攻击者用来隐藏自己的踪迹和保留root访问权限的工具，它能使攻击者一直保持对目标机器的访问，以实施对目标计算机的控制[1]。从Rootkit运行的环境来看，可将其分为用户级Rootkit和内核级Rootkit。

用户态下，应用程序会调用Win32子系统动态库（包括Kernel32.dll,User32.dll,Gdi32.dll等)提供的Win32 API函数，它们是 Windows 提供给应用程序与操作系统的接口，运行在Ring 3级。用户级Rootkit通常就是通过拦截Win32 API，建立系统钩子，插入自己的代码，从而控制检测工具对进程或服务的遍历调用，实现隐藏功能[10]。

内核级Rootkit是指利用驱动程序技术或其它相关技术进入Windows操作系统内核，通过对Windows操作系统内核相关的数据结构或对象进行篡改，以实现隐藏功能。由于Rootkit运行在Ring 0级别，甚至进入内核空间，因而可以对内核指令进行修改，而用户级检测却无法发现内核操作被拦截[10]。下面介绍两种使用Rootkit技术来实现进程隐藏的方法。

2.4.1 SSDT Hook

反汇编taskmgr.exe，可以发现taskmgr.exe是通过调用内核中的服务函数NtQuerySystemInformation枚举进程的，因此可以对NtQuerySystemInformation设置钩子函数来修改它的行为，实现隐藏进程的目的。SSDT Hook就是基于这样的思路。

Windows操作系统以Win32 API的形式给用户提供了EnumProcess、CreateToolhelp32Snapshot等函数来遍历当前系统中的所有进程，这些函数位于Win32子系统动态库Kernel32.dll中。它们在执行时都会调用Ntdll.dll导出的、内核能够理解的API函数ZwQuerySystemInformation，该函数实际上只是简单的通过中断进入内核态，接着由系统服务调度程序查询SSDT（System Service Dispatch Table，系统服务调度表）找到要执行的服务函数的地址。服务函数就是位于ntoskrnl.exe底层内核中的NtQuerySystemInformation。操作系统在执行上述操作时的处理流程如图1所示[10]。

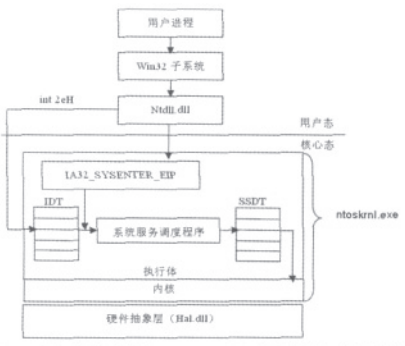

图1 用户程序调用Win32API时系统的处理流程

从图1可以看出，由Ntdll.dll到核心态有两种途径：一种是通过中断调用int 2eH陷入内核，查找IDT（中断描述符表），这是Windows 2000使用的方式；另一种是通过SYSENTER指令完成与中断相同的功能，这是Windows XP使用的方式，接下来两者都调用系统服务调度程序，由系统服务调度程序查找SSDT，确定要调用的系统服务函数的地址，然后从执行体转到底层内核，执行相应的系统服务函数。

需要说明的是，在ntoskrnl.exe中存在两个SSDT，分别由指针KeServiceDescriptorTablet和KeServiceDescriptorTabletShadow导出，前者的访问接口由Ntdll.dll提供，可供上一层Win32子系统动态库Kernel32.dll和advapi32.dll中导出的Win32 API间接调用；后者的访问接口由User32.dll和Gdi32.dll提供，系统服务的功能在Win32.sys中实现[10]。图1中的SSDT属于第一种情况。

由于操作系统在SSDT中存放了所有系统服务函数的入口地址，系统服务调度程序可以通过查找SSDT来获得相应的系统服务函数NtQuerySystemInformation的地址。这种机制给木马带来了可乘之机，在这个过程中，内核级Rootkit可以将SSDT中的系统服务函数NtQuerySystemInformation的入口地址替换为自己的Hook函数的地址。这样，当操作系统调用系统服务函数时，实际调用的就是Hook函数[10]。Hook函数首先调用原来的系统服务函数NtQuerySystemInformation来获得进程的相关信息，再对结果进行相应的处理：查找需要隐藏的进程信息，将其过滤，最后再把处理后的结果返回给上层函数。这样上层的API函数就无法通过枚举进程来获得系统底层真正的进程信息，从而达到了隐藏进程的目的。这是使用SSDT Hook方法隐藏进程的基本原理。

由于系统服务函数处于内核态，在用户程序中无法直接调用，因此需要借助驱动程序技术或其它的相关技术来实现调用。另外，一些高版本的Windows操作系统如Windows XP和 Windows 2003使用了特定的内存写保护，只能对SSDT进行读操作，不能直接对其改写，为此，可以将SSDT 所在的内存区域映射为一个MDL（Memory Descriptor List，内存描述符表），通过更改这个MDL 的flags，就可以解除写保护[9]。

2.4.2 DKOM（ Direct Kernel Object Manipulation，直接内核对象操作）

使用DKOM方法进行进程隐藏。在Windows操作系统中，系统会为每一个活动进程创建一个进程对象EPROCESS，为进程中的每一个线程创建一个线程对象ETHREAD。在EPROCESS进程结构中有个双向链表LIST_ENTRY，LIST_ENTRY结构中有FLINK和BLINK两个成员指针，分别指向当前进程的前驱进程和后继进程[9]。如果要隐藏当前进程，只需把当前进程的前驱进程的BLINK修改为当前进程的BLINK，再把当前进程的后继进程的FLINK修改为当前进程的FLINK。如图2所示[9]，虚线表示的就是把

（下转46页）

击者通过修改注册表的方式将网络监控程序所属SPI卸载。其总体流程图如图4所示。

4 试验测试结果

该网络监控程序设计方案在国家保密局涉密网络监控系统中得到实施。在实际中，针对该程序各个性能的测试结果如下。

4.1 网络监控系统自启动

操作系统启动之后，不做任何操作，插入外网网线，使得系统能够连接外网，将当前用户信息提取发送到管理员邮箱之后，系统自动强行关机。证明成功实现了网络监控程序系统自启动功能。

4.2 进程自我隐藏

系统启动之后，通过进程查看等各种方法，无法观察到监控进程的存在，同时真正用户体验也无法感受到该监控程序的存在，证明成功实现了自我隐藏功能。

4.3 Rootkit自我保护

系统启动之后，原先位于程序安装目录的关键动态链接库消失不见，系统注册表也无法查询到本系统的网络监控SPI，证明成功实现了Rootkit自我保护功能。

4.4 程序防止强制关闭

根据调试，发现程序被加载到svchost.exe<network>进程中，强行将该进程删除，系统提示核心进程遭到破坏，系统启动强行关机程序。证明成功实现了防止攻击者强行关闭功能。

5 结论

通过上面的分析和测试，我们可以看到，将基于Winsock2 SPI框架的网络监控程序能够成功将自身监控线程同系统关键进程进行绑定，这种程序能够在不被用户察觉的基础上，实现进程的自我隐藏，并且防止攻击者强行关闭。在同Rootkit等其他技术进行结合之后，能够进一步提高程序的自我保护和自我隐藏性能，防止用户从硬盘上直接删除程序关键文件或者通过注册表卸载网络监控程序所属SPI。因此，基于SPI技术的网络监控程序具有较高的自我保护性能和自我隐藏性能。（责编 张岩）

参考文献：
[1] 何波，董世都，涂飞，程勇军. 涉密网安全保密整体解决方案[J]. 微计算机信息，2006年，第22卷，第9-3期:116-118.
[2] 蒲天银. 安全隔离网闸技术发展探讨[J]. 计算机时代，2006年，第6期: 18-19.
[3] 董惠勤，陆魁军. 跨安全网闸的内外网数据库同步的实现. 科技通报，2007年，23卷(2期): 266-270.
[4] 吴晓昶，李名世. 办公业务网信息监控系统设计[J]. 厦门大学学报，2004年，增刊: 332-335.
[5] 李焕洲，张健，陈麟. 涉密网资源监控体系的研究与实现[J]. 计算机应用，2006年，26卷(5期): 1090-1092.
[6] 甘利杰，丁明勇，杨永斌. 基于Winsock SPI技术的包过滤研究[J]. 计算机科学，2007年，08期: 112-114.
[7] 田磊，李祥和，辛志东，潘军. 基于Winsock2 SPI技术的木马植入新方案[J]. 计算机工程，2006，7期: 166-168.
[8] Jones A. Network Programming for Microsoft Windows (SecondEdition) [M]. Microsoft Press, 2002.
[9] Greg Hoglund, James Butler, Rootkit-subverting the windows kernel, Addison Wesley Professional, July 22, 2005.

作者简介：张亚航(1985-)，男，硕士研究生，主要研究方向：网络安全；文伟平(1976-)，男，副教授，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。

（上接37页）

当前进程从链表中删除以后的结果。以上就是使用DKOM方法实现进程隐藏的基本思路，即平常所说的“摘链法”。

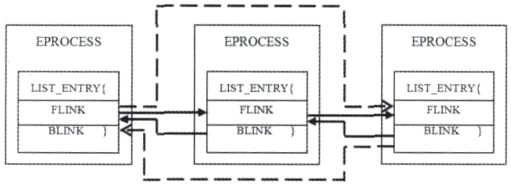

图2 DKOM方法删除当前进程示意图

3 结论

从上面可以看出，木马实现常用的技术很多，木马用于进程隐藏的技术手段也是多种多样的：从应用层到操作系统内核，由表及里，越来越深入。对于各种进程隐藏技术，我们可以在研究和分析其基本原理与方法的基础上，一方面，考虑把它们应用到我们实际的系统管理工作中去，如某些安全控制程序；另一方面，可以为开发防病毒和木马软件打下良好的基础，从而使我们更好地检测和防御木马，确保我们系统的安全。（责编 张岩）

作者简介：卢立蕾(1971－)，女，高级讲师，硕士研究生，主要研究方向：系统与网络安全；文伟平(1976－)，男，副教授，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。

参考文献：
[1] 孙淑华，马恒太，张楠，卿斯汉. 内核级木马隐藏技术研究与实践[J]. 微电子学与计算机，2004，21(3): 76-80 .
[2] 许国顺. 木马攻击与防范技术研究[D]. 四川大学：工程硕士学位论文，2006，03 .
[3] 贾建忠，姜锐. 新型木马技术剖析及发展预测[J]. 网络安全技术与应用，2007，5: 43-45.
[4] 刘牧星. 木马攻击与隐蔽技术研究[D]. 天津大学：硕士学位论文，2006.
[5] 文伟平. 恶意代码机理与防范技术研究[D]. 中国科学院研究生院(软件研究所)：博士论文，2005.
[6] 康治平，向宏. 特洛伊木马隐藏技术研究与实践[J]. 计算机工程与应用，2006，09: 103-105，119.
[7] 彭迎春，谭汉松. 基于DLL 的特洛伊木马隐藏技术研究[J]. 信息技术，2005年，第12期: 41-43.
[8] 于继江. 动态嵌入式DLL木马实现方法[J]. 电脑知识与技术，2005年，21期: 47-49.
[9] Greg Hoglund，James Butler.Rootkits:Subverting the Windows Kernel. July 22，2005.
[10] 杨彦，黄皓.Windows Rootkit隐藏技术研究.计算机工程，2008，6，Vol.34，No.12:152-156.