Windows 平台下软件自身防护关键技术①

李诗松[1]，陈　伟[2]，陈　运[1]

[1](95942 部队，武汉 430013)
[2](95944 部队，武汉 430013)

摘　要：提高 Windows 平台下软件自身防护包括加强对敏感文件和自身进程的安全保护两个方面。介绍了目前常用到的三种文件保护方案，以及进程保护的技术，并对他们存在的优缺点进行了分析。

关键词：文件过滤驱动；HOOK 技术；进程保护

Protection Technology for Software Under the Windows Platform

LI Shi-Song[1], CHEN Wei[2], CHEN Yun[1]

[1](95942 Troops People's Liberation Army, Wuhan 430013, China)
[2](95944 Troops People's Liberation Army, Wuhan 430013, China)

Abstract: To improve the ability of protection technology for software include two aspect: enhance the peotection for sensity file and process itself. This paper presents three scheme for file protect in common use, as well as the technology for peocess protect, and then analyses their advantages and disadvantages.

Key words: file filter driver; hook technology; process protect

随着信息科学研究的不断深入，信息攻防手段也不断发展。同时，攻击者实施攻击的目标逐渐从单纯的炫耀技术走向了以获取经济利益和情报为最终目标，信息泄露给用户带来的危害比以往更大。为此，提高软件特别是一些监控系统操作的关键进程自身的生存能力也越来越受到广大研究人员的重视。此时，监控系统自身生存能力就显得尤为重要。为了实现安全防护的目的，至少需要做好两个方面的防护：一是对日志文件的防护，日志文件是安全审计和追查可能攻击者的重要信息，保护好日志文件应该做到防止他人对日志文件的非法查看、复制、删除等操作；二是对监控系统守护进程的防护，监控系统的守护进程不能因有意或无意关闭，造成信息的失控，即提高主进程的存活能力。

本文就目前使用最普遍的 Windows 操作系统平台下软件自身安全防护的关键技术进行了讨论。文中第一节讨论了 Windows 平台下主要的文件保护方案，并对各自的优缺点作了简单分析。第二节对软件的进程防护技术进行了讨论。

1　Windows敏感文件保护方案分析

Windows 平台可分为 Windows 内核层、用户层，在内核与用户之间通过系统服务接口相连结。

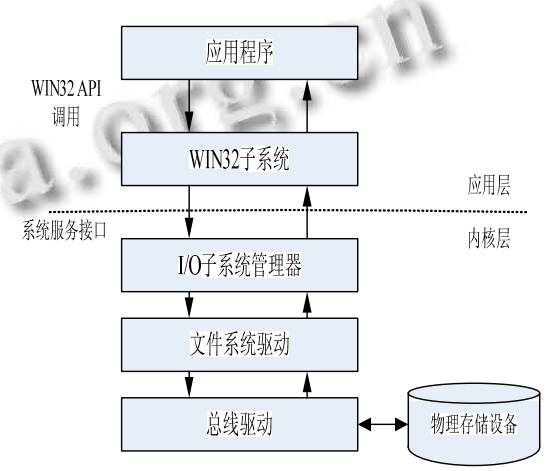

图 1　Widows NT 文件系统工作原理

在 Windows 内核层通过编写文件过滤驱动程序，增加一层文件访问控制内核过滤驱动程序实现文件的访问控制。基于文件过滤驱动的文件访问控制具有较

① 收稿时间:2011-08-04;收到修改稿时间:2011-10-06

强的访问控制功能，由于其访问控制能力是一种 Windows 内核态中间层驱动，不需要改变下层驱动或用户程序而增加新的功能，具有效率高、可靠性强、可扩充等特点，成为现阶段信息安全技术研究的热点。在 Windows API 接口层上，应用 HOOK(钩子)技术，通过截获 Windows 提供的文件操作消息，监视文件访问，缺点是效率低、稳定性和一致性差，不适合于大型系统的开发[1]。另外，还有一种技术是通过利用 Windows 自身文件访问机制，限制 Windows 平台下其他进程对文件保护对象的访问。

1.1 基于文件过滤驱动的文件访问控制方案

针对 Windows 文件系统而言，文件过滤驱动既可以位于文件系统驱动之上，也可位于文件系统驱动和存储设备驱动之间[2]。I/O 管理器接收上层应用程序的 I/O 请求，将其转换成成 IRP，并传递给下层驱动[3]。

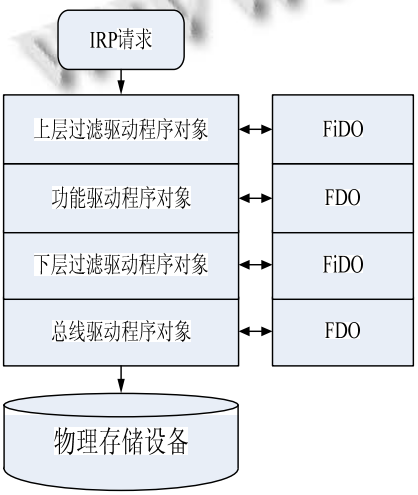

图 1 设备对象驱动程序

Windows2000/xp 依然采用的是 NT 内核，它的 I/O 子系统是由 I/O 管理器，可扩展的一组驱动程序和其它一些执行体服务组成。I/O 管理器采用的是分层驱动程序模型，如图 2 所示。设备对象(FDO，FiDO)、驱动程序对象都是系统为便于分层管理而创建的数据结构。一个驱动程序可以根据需要创建多个设备对象。每个 I/O 请求，由 I/O 管理器出发，依次从相应的设备栈的顶部向下传递。每传递一层，系统就调用与当前设备对象关联的驱动程序例程来对请求进行处理。

Windows 2000 的这种分层的驱动程序模型，允许一个驱动程序构造一个匿名的设备对象，并把它附着在另一个设备对象上[4]。I/O 管理器在传递 IRP 到目标设备对象之前，如果有附着的匿名设备对象，它就把 IRP 首先传递给此匿名设备对象，经过对象的过滤设备驱动程序处理之后才发给真正的目标设备对象。而这个附着的匿名设备对象就是我们说的 FiDO(过滤设备对象)。与它关联的驱动程序就称为过滤驱动程序。因此，我们可以在文件系统设备对象的上面设置一个过滤驱动程序，这个过滤驱动程序可以首先获得 I/O 的 IRP 请求，从而对原始的请求进行预处理、修改和监控，达到对文件保护的目的。

一次完整的文件打开操作的完成过程如图 3 所示：由应用程序通过 Windows API 接口通过 Win32 子系统向内核 I/O 系统服务提交文件打开操作请求，I/O 系统服务调用文件对象管理器查找指定文件对象的相关符号链接和位置信息，同时调用安全监视器检查文件的访问权限。如果以上操作正确，由 I/O 管理器为本次访问分配 IRPs，经文件驱动和设备驱动实现请求的操作。最后将请求的结果通过 I/O 管理器返回给上层协议。

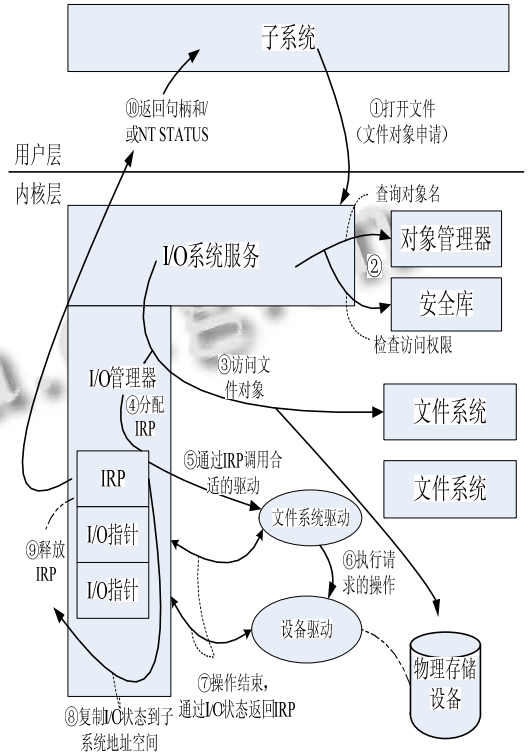

图 2 打开一个文件对象一般处理过程

1.2 基于 HOOK 的文件保护方案

Windows 操作系统是建立在事件驱动机制之上的，系统各部分通过传递消息而相互沟通。钩子是

Windows 操作系统中非常重要的一种系统接口，用它可以轻松截获并处理在其他应用程序之间传递的消息，并由此可以完成一些普通应用程序难以实现的特殊功能。

钩子的本质是一段用以处理系统消息的程序，通过系统调用，将其挂入到系统。任何一个钩子都由系统来维护一个指针列表（钩子链表），其指针指向钩子的各个处理函数。最近安装的钩子放在链的开始，最早安装的钩子则放在最后，当钩子监视的消息出现时，操作系统调用链表开始处的第一个钩子处理函数进行处理，即最后加入的钩子优先获得控制权。其中钩子处理函数必须是一个回调函数（callback function），而且不能定义为类成员函数，必须定义为普通的 C 函数。在使用钩子时可以根据其监视范围的不同将其分为全局钩子和线程钩子两大类，其中线程钩子只能监视某个线程，而全局钩子则可对在当前系统下运行的所有线程进行监视。显然，线程钩子可以看作是全局钩子的一个子集，全局钩子虽然功能强大但钩子函数的实现必须封装在动态链接库中，实现起来也比较烦琐，对系统性能的影响也最为明显。

在 Windows 中，也可以在系统 API 层上通过截获系统对文件操作，如文件的打开、删除、复制、重命名等，来监视用户对文件夹的操作。将一个文件夹对象安装多个拷贝钩子处理程序，Shell 会依次调用每个处理程序。只有当每个处理程序都返回 IDYES 时，Shell 才真正执行用户请求的操作。

基于 HOOK 的文件保护方案由于需要对 Windows 的消息截获后进行分析处理，故可以采用全局钩子来实现，但显然这容易导致系统性能的迅速下降，故在使用上应当慎重。

1.3 基于 Windows 文件访问机制的文件保护方案

在 Windows 系统内部，对每一次文件操作也要对其权限进行检查。当一个进程需要对系统中某一文件进行操作时，系统首先检查本次访问与文件自身访问必要性是否相符，然后检查访问进程是否与其他进程正在进行的操作相冲突[5]。

在 VC 环境中，对于文件操作类 CFile 中，在创建文件操作对象时可以定义一个 CFile::shareExclusive 标志，拒绝其他应用程序对本文件的读和写操作，也可以设置 CFile::shareDenyWrite?或 CFile::shareDenyRead 对文件与其他进程的访问策略进行设置。这样，在进程存活的整个时间内，其他任何的进程对本文件的操作均要求与该进程的策略相一致。但这种方案不能防止来自底层的攻击，如直接对硬盘的读写，就可以绕过原定的策略限制。另外，在一些特殊的应用场合，如对特殊的日志记录文件的保护，要求加强进程自身的防护能力。本文第二节就可能的进程保护技术进行了讨论。

2 Windows平台下进程保护的实现

加强进程自身的安全防护，是提高安全监控软件的生存能力的重要途径。本节将就目前常用的进程保护方法进行讨论。

2.1 基于多进程互锁的进程保护方案

在操作系统中，进程是存储器，外设等资源的分配单位，同时也是处理器调度的对象，但为了提高进程内的并发性，操作系统中引入了线程这个概念，这时系统把线程作为处理器调度的对象，一个进程可以同时拥有多个并发的线程。通常情况下的简单程序就只有一个主线程，它是在进程创建时自动生成的。我们可以将想要执行的代码放在主进程里。同时生成两个远端辅助线程，实现对程序的保护功能，防止程序被用户关闭或删除。我们称这种方案为基于多进程互锁的进程保护方案。

主线程需要完成的任务有三个，它们分别是准备工作，创建辅助线程和程序主要功能的实现。准备工作当然是为程序运行过程中所需要的一些部件做好准备，其中包括文件复制和保存，一般情况下是把可执行文件复制到系统目录下。当然为了防止意外删除，我们可以将程序的可执行文件备份，不过要注意修改一些属性（文件类型，大小，日期，属性），这样就不容易被发现与我们的可执行文件有关联了。创建辅助线程包括两个线程，一个驻留在主进程体内，另一个通过创建远程线程驻留到其他正在运行的进程体内。这两个线程的功能就是监视其他进程或线程的运行情况，如果出现异常立即恢复。

驻留主进程内的线程同时观察注册表和远程进程的情况。它实时查询注册表里 HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run 键下相关可执行文件的键值，如果被删除就立即将其恢复。这就是对注册表的实时监视，使得每次开机时都会运行我们的可执行文件。而另一个功能则是

监视驻留在远程进程体内辅助线程的运行情况，如果该线程被关闭，立即通过创建远程线程将被关闭的线程驻留到特定的进程内。如果你知道了辅助线程是驻留在那个进程内的，你就可以将这个远程进程关闭掉。但是我们还是可以将辅助线程驻留到其他的远程进程内。远程进程的选定上，一般选用与系统运行紧密联系的系统级进程。

驻留在远程进程体内的辅助线程则监视主进程的运行情况，如果主进程被关闭，它会确认程序的可执行文件是否也被删除掉了。如果系统目录下的可执行文件不存在了，则用我们备份的文件恢复可执行文件，然后再重新启动程序，这样你在任务管理器里怎么也删除不了主进程。由于我们是创建远程线程，所以必须把线程的代码和线程所需要的参数都复制到远程进程的地址空间里。这是因为在windows2000/xp环境下，访问其他进程地址空间是违规的。我们把代码和参数复制过去后，线程就完全在远程进程的地址空间运行了。我们可以看到，通过两个辅助线程，就很难把主进程关闭或删除掉，达到进程保护的目的。

2.2 进程隐藏

在 Windows 平台的进程隐藏分为伪隐藏和真隐藏两种，其中伪隐藏是指采用 API 拦截技术，通过建立系统钩子，拦截 PSAPI 的 EnumProcessModule 等相关函数来实现对进程和服务的遍历控制，当到进程 ID 为自身 ID 时，直接跳过，从而实现进程的隐藏。而进程的真隐藏是通过将自身程序运行代码注入到系统其他进程空间中，通常为系统进程。实现这种代码注入的方法一般有 DLL 注入方法、WriteProcessMemory 方法等。

我们通常所讲的进程隐藏一般指进程的伪隐藏，实现起来相对较为简单，而且要实现对整个监控系统的进程真隐藏是十分困难的，实现效果也不理想。同时，由于杀毒软件等原因，可能导致 WriteProcess Memory 方法操作失败，造成在某些应用环境下系统启动失败。

3 结语

本文对 Windows 平台下加强监控系统软件安全防护能力的两项关键技术进行了讨论。其中，可行的文件保护技术也还有其他一些实现的途径，如基于文件恢复技术的文件保护方案等。总体而言，将自身安全策略放在更加底层来实现，对实现效果而言会更理想，但同时实现技术上也要求更高，系统出错对操作系统带来的影响也更大。随着信息安全监控能力的不断增强和发展，对于软件自身安全防护能力的研究将成为其未来研究的重要方面。

参考文献

1 王川.基于过滤驱动的文件保护系统.网络安全技术与应用,2011,(1).

2 Oney W. Programming Microsoft Windows Driver Model 2nd ed. Microsoft Press,2003.

3 Whitepapers. I/O File System Filter Driver For Windows NT.http://www.cswl.com/whiteppr/white/Guaranteed.html,2005–02–23.

4 Baker R, Lozano J. Windows 2000 设备驱动程序设计指南 第 2 版.施诺,等译.北京:机械工业出版社,2001.

5 王雷,庄毅,潘龙平.基于强制访问控制的文件安全监控系统的设计与实现.计算机应用,2006,(12).