

天津大学

硕士学位论文

基于行为的入侵防护技术研究

姓名：余保明

申请学位级别：硕士

专业：计算机应用技术

指导教师：张钢

20070601

中文摘要

网络安全问题已成为信息时代人类共同面临的挑战，国内的网络安全问题也日益突出。计算机恶意代码使用的技术也在不断深化，有些恶意代码可以终止、修改或挂起杀毒软件、防火墙等安全软件的进程，使系统的保护机制失效。因此，如何防止病毒和木马的运行、如何检测系统进程和可信进程不受其它进程的入侵攻击是一项重要的研究课题。

本文的主要研究结果是：论文讨论了国内外入侵检测发展现状，深入剖析了几个典型病毒和木马程序的运行机理，在此基础上提出了一个基于行为的入侵防护系统模型，该模型完全符合 PPDR 的动态安全防护要求。同时描述了该模型的结构以及在策略、检测、响应和保护四个环节的工作原理，分析了该模型的优缺点；其次在分析进程的各种入侵攻击行为的基础上，深入研究了实现入侵防护模型的关键技术——通过采用内核修补技术，实现了与操作系统的无缝连接，在内核态中完成了对进程入侵攻击行为的拦截；入侵防护系统担负着控制进程创建和保护系统进程和可信进程不受其它进程攻击的使命，因此，本文对如何保证入侵防护系统自身的安全也进行了研究。

关键词：入侵防护 进程防护 恶意代码 病毒

ABSTRACT


The network security has become a challenge to all human beings in the information age. The technology of using computer malicious code is deepening, some malicious codes can even terminate, alter and hang the process of the antivirus software, firewall and invalidate its guard mechanism. So how to prevent the operation of virus and Trojan horse, and how to detect the system process and reliability process free from the invasion of other processes is an important research subject.

In this paper, we have done some research about the following subjects: First, we discussed the present situation of invasion detection both home and abroad, then we analyzed the operation mechanism of some typical viruses and Trojan horse programs deeply and raised a behavior-based invasion guard system model. This model coheres with the dynamic security guard demands of PPDR. We described the model's structure and its working principles of strategy, detection, reaction and guard, then we analyzed the model's advantages and disadvantages; meanwhile, in the paper we studied the key technology of realizing the invasion guard system deeply based on the analysis of invasion behaviors. We accomplished the connection between our system and the operating system, and we also accomplished the interception of invasion behaviors. The invasion guard system underook the creation of control process and the mission of guard process and reliability process free from the invasion of other processes. We have also done some research on the invasion guard system's self security.

KEY WORDS: Invasion Guard Process Protection Malicious Codes Virus

独创性声明


本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 天津大学 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

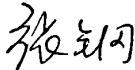
学位论文作者签名： 签字日期：2007 年 6 月 14 日

学位论文版权使用授权书

本学位论文作者完全了解 天津大学 有关保留、使用学位论文的规定。特授权 天津大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

(保密的学位论文在解密后适用本授权说明)

学位论文作者签名：

导师签名：

签字日期：2007 年 6 月 14 日

签字日期：2007 年 6 月 14 日

第一章 绪论

1.1 引言

目前,网络系统的广泛应用已为企业的管理、运营和统计等带来了前所未有的高效和快捷,但同时计算机网络的安全隐患亦日益突出。

国际咨询服务机构 Frost and Sullivan 近日宣称,逾七成亚洲企业遭遇过网络安全入侵,与日俱增的网络攻击行为迫使计算机用户为了防止病毒、木马和恶意代码对主机系统的破坏以及为了保护自己的敏感信息不被窃取,一般都安装杀毒软件;如果主机接入因特网,为了防止黑客攻击还要安装个人防火墙。即使这样,仍有数不胜数的主机因遭到病毒、木马和恶意代码的入侵和破坏而造成了重大的经济损失。这是因为传统的杀毒软件存在局限性,它们都是通过模式匹配的方式来查找已知的病毒、木马和恶意代码,但对于新出现的病毒和已知病毒的变种不能够进行有效地检测;传统的个人防火墙只能进行包过滤,对于病毒的侵袭也是束手无策。

国内普遍使用的反病毒软件主要有瑞星、江民、赛门铁克、卡巴斯基等公司的产品。其最新版杀毒软件中都实现了对计算机的实时监控,如果安装了这些反病毒软件,就可以对文件、邮件、内存、引导区、网页、注册表等进行实时监控,这在一定程度上降低了系统遭受未知病毒入侵的风险性,有效地保护计算机系统的安全,但是,这些反病毒软件都没有做好入侵前的充分防护。现在,许多病毒通过利用远程插入线程技术、全局挂钩技术将恶意代码注射到其它进程中运行,有的木马甚至采用内核驱动技术来隐藏文件名、注册表信息、进程名、端口名等。这给查毒和杀毒带来了相当的难度。

入侵防护系统所遵循的思想就是:“把充分的防护措施做到恶意代码入侵之前”。入侵防护所要实现的目标是保护 Windows 系统进程和用户可信进程免受其它进程、服务、驱动程序、以及系统上的其它形式的可执行代码的攻击;未被用户许可的程序不让运行。

面对复杂的网络环境,依靠传统单一防病毒、防火墙等产品,不足以完全制服广泛传播的网络威胁。顺应新时期安全需求,只有将防病毒、防火墙、入侵检测、漏洞扫描等多项安全技术紧密集成在一起,形成整体的安全解决方案才能真正抵御威胁入侵^[1]。因此,研制和开发基于行为的入侵防护系统是非常有必要的,它是对目前流行的反病毒软件的有效补充。反病毒软件的计算机实时监控、个人

防火墙和入侵防护系统组成了一张主机安全防护网,可以对计算机系统特别是主机系统实施有效的保护。

1.2 国内外研究现状

目前,恶意代码的技术发展更加迅速。以木马为例,第四代木马与以前的木马相比,在进程隐藏方面做了很大的改动,它利用远程插入线程技术、嵌入 DLL 线程或者挂接 PSAPI 实现木马程序的隐藏,达到了良好的隐藏效果^[2]。目前木马已经发展到了第五代,ROOTKIT 是第五代木马的典型代表,它通过更改内核系统结构或者更改内核执行路径来隐藏文件名、进程名、端口号、注册表信息等,这给计算机安全技术的研究提出了新的课题。

这种严峻现实,促使许多安全软件公司和研究人员致力于各种动态防护技术的研究。目前,在国内,主要有中科网威^[3]、启明星辰、清华紫光比威、三零盛安、东软软件、中联绿盟、复旦光华、上海金诺、瑞星、金山、联想等安全届知名厂商有自己的 IDS 产品。国内的产品多数集中在低端、采用模式匹配的方法,与国外相比还有一段差距。在 Windows 操作系统下,利用更改系统函数执行路径的内核修补技术来对进程进行全面防护的学术文章和商业软件比较少,比较有名的软件是有澳大利亚的 Diamond Computer Systems Pty. Ltd 公司研制的 Process Guard^[4],这样的商业软件国内目前还没有发现。

1.3 论文安排和研究结果

这一节简要介绍一下论文的内容安排和一些主要研究成果。

1.3.1 论文的安排

第二章:在对典型病毒剖析的基础上,总结归纳了病毒和木马的运行机理。

第三章:根据 PPDR 模型的动态防护思想,设计了一个基于行为的入侵防护体系 (Invasion Guard Based On Behaviors,简称 IGBOB)模型,并分析了该模型的结构及工作原理。

第四章:在分析各种进程攻击行为基础上研究了几种内核修补技术和方法。

第五章:研究了控制进程创建、控制进程终止、控制访问物理内存、控制安装驱动等的方法和实现技术。对如何防护 IGBOB 自身安全提出了实现方法。设计了一个映像加载规则库,并说明其工作原理。

第六章:结束语。

1.3.2 主要研究成果

以下是本文取得的一些成果：

1、 深入剖析典型病毒、木马的基础上，总结了病毒、木马的植入技术和运行机理，为建立防护模型提供了依据。

2、 提出了一个完全符合 PPDR 的基于行为的入侵防护系统模型，描述了该模型的结构以及在策略、检测、响应和保护四个环节的工作原理，并分析了该模型的优缺点。该模型可以有效阻止恶意代码的传播，并可保护系统进程和可信进程不受其它进程的攻击。

3、 对实现模型的关键技术进行了详细研究。在深入分析进程的各种攻击行为的基础上，通过采用内核修补技术，实现了与操作系统的无缝连接，在内核态中完成了对进程攻击行为的拦截。

4、 入侵防护系统担负着控制进程创建和保护系统进程和可信进程不受其它进程攻击的使命，因此，本文也对如何保证入侵防护系统自身的安全进行了研究。

第二章 计算机病毒和木马概述

从第一个计算机病毒出现到现在,已经有整整半个世纪了。病毒的发展日新月异,令查杀的困难大大增加,造成的损失也越来越大。计算机病毒这个幽灵,从计算机诞生的那一刻起就注定要如影相随的。只要还有用心险恶的人存在,那么病毒就不会消亡。病毒之战,会在今后的日子里越演越烈……

2.1 计算机病毒和木马的定义

病毒——1994年2月18日,我国正式颁布实施了《中华人民共和国计算机信息系统安全保护条例》。在该条例的第二十八条中明确指出:“计算机病毒,是指编制或者在计算机程序中插入的破坏计算机功能或者毁坏数据,影响计算机使用,并能自我复制的一组计算机指令或者程序代码^[5]。”

木马——来自“特洛伊木马”,是指一种与远程计算机之间建立起连接,使远程计算机能够通过网络控制本地计算机的恶意程序。它的运行遵照 TCP/IP 协议,由于它像间谍一样潜入用户的电脑,为其他人的攻击打开后门,与战争中的“木马”战术十分相似,因而得名木马程序^[6]。

蠕虫——蠕虫是计算机病毒的一种,是指利用网络缺陷进行繁殖的病毒程序,其原始特征之一是通过网络协议漏洞进行网络传播^[7]。

脚本病毒——利用脚本来进行破坏的病毒,其特征为本身是一个 ASCII 码或加密的 ASCII 码文本文件,由特定的脚本解释器执行。主要利用脚本解释器的检查漏洞和用户登录身份的不当对系统设置进行恶意配置或恶意调用系统命令造成危害。

但目前,由于病毒,木马,蠕虫,脚本病毒这四类程序在不断杂交中衍生,已经形成了“你中有我,我中有你”的多态特性。它们将网络蠕虫、计算机病毒、木马程序合为一体,开创了网络病毒传播的新路^[7]。为了行文方便,以下将病毒、木马、蠕虫、脚本病毒统称为“病毒”,但其实这四类程序的感染机制和编写方式是完全不同的。

2.2 典型病毒剖析

病毒入侵计算机后,会想方设法获取系统的运行权,不然的话,病毒将失去

活力，变成硬盘上的一个垃圾文件。下面，我们对几个典型病毒进行分析，看它们是如何获取系统的运行权。

2.2.1 “红色代码”病毒

“红色代码”病毒采用了一种叫做“缓存区溢出”的黑客技术，利用网络上使用微软 IIS 系统的服务器来进行病毒的传播。这个蠕虫病毒使用服务器的 80 端口，而这个端口正是 Web 服务器与浏览器进行信息交流的渠道^[6]。“红色代码”病毒能够破坏 Windows 2000 服务器安全体系，更改系统设置，修改 Windows 文件并放置特洛伊木马程序，最终导致被感染的计算机系统后门大开。

“红色代码”病毒入侵系统后，将 Cmd.exe 复制到 C:盘及 D:盘的目录中，其文件名为 root.exe: \inetpub\scripts\program files\common files\system\msadc 然后，病毒开始扫描网络，寻找其它可被攻击的系统，这一过程在英文 Windows 2000 系统中将持续 24 小时，而在运行中文 Windows 2000 系统中将持续 48 小时。接着，病毒程序在 C:盘和 D:盘的根目录下生成一个大小为 8192 字节的 Explorer.exe 木马程序，然后重启系统，执行木马程序。

这个木马程序首先运行 Windows 的 Explorer.exe 程序，然后通过修改系统注册表的以下键值，使 Windows 丧失对系统文件的保护能力：

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\

Windows NT\CurrentVersion\Winlogon\SFCDisable=0xFFFFFFFF9D，该键值的默认值为 0。

修改之后，木马程序通过修改系统注册表以下键值，创建两个虚拟 IIS 目录 C 和 D，分别映射到系统的 C:盘和 D:盘。

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\Virtual Roots

这些虚拟目录被赋予读写及执行的权限，这样木马程序通过 IIS 向所有黑客提供了对被感染服务器 C:盘和 D:盘的完全控制能力。

蠕虫将“explorer.exe”木马放在“C:\”和“D:\”的根目录下面，这是想利用微软安全公告 MS00-052 中所描述的漏洞，windows 系统在执行可执行程序时，会先搜索系统盘根目录下面有没有同名的程序，如果有，就先执行该程序^[8]。因此，如果攻击者将“exploer.exe”木马放在系统盘根目录下面，就可能先于真正的“exploer.exe”被执行。当属于管理员组的用户交互地登录进入系统时，木马将被执行。如果您没有安装 SP2 或者 MS00-052 中的补丁，您就可能执行这个木马程序；否则，您不会执行这个木马。

2.2.2 Win32.Troj.ADNavihelper(广告木马)

该病毒属于广告木马，一般会捆绑在正常软件中。当用户运行捆绑有该病毒的文件时，会释放出该文件。病毒通过浏览器帮助对象(Browser Help Object, 简称 BHO)，注入到 IE 浏览器中，然后通过 SQL 查询，强行打开指定网页。这样，浏览器不时地打开广告窗口，如：“女生宿舍”或“青涩宝贝”等，给用户使用浏览器带来不便^[9]。

“广告木马”入侵系统后

1、将自身存放在如下路径中：

`%system%\Navihelper.dll`

2、添加如下注册表键值

`HKEY_CURRENT_USER\AppID\{13FACA62-5FC4-4817-9175-9C8D00975916}`

`HKEY_CURRENT_USER\AppID\NaviHelper.DLL`

`HKEY_CURRENT_USER\NaviHelper.NaviHelperObj.1`

`HKEY_CURRENT_USER\NaviHelper.NaviHelperObj`

`HKEY_CURRENT_USER\CLSID\{3E422F49-1566-40D3-B43D-077EF739AC32}`

`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects\{3E422F49-1566-40D3-B43D-077EF739AC32}`

这些键值用于注册 BHO，使得每次开启 IE 浏览器时都会加载该广告程序。

3、从网上下载

`http://bar.×××××8.com/host.dat` 到本地的 `%system%\host.dat`

4、host.dat 文件为 SQL 数据库，包含了全部要显示的网页地址，包括以下网址：

`http://www.qul23.com/aoyu1.html`

`http://baby.aoe88.com/ad.html`

2.2.3 冲击波病毒

计算机被“冲击波”病毒感染后，会产生下列现象：系统资源被大量占用，有时会弹出 RPC 服务终止的对话框，并且系统反复重启，不能收发邮件，不能正常复制文件，无法正常浏览网页，DNS 和 IIS 服务遭到非法拒绝等。下面是弹出 RPC 服务终止的对话框的现象：

“冲击波”入侵系统后：

1、将自身复制到 window 目录下，并命名为 `msblast.exe`。

2、病毒运行时会在系统中建立一个名为“BILLY”的互斥量，目的是保证在内存中只有一份病毒体，避免被用户发现。

3、病毒运行时会在内存中创建一个名为 `msblast.exe` 的进程，该进程就是活

的病毒体。

4、病毒会修改注册表，在 HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run 下添加以下键值："windows auto update"="msblast.exe"，以便每次系统启动时，病毒都会运行。

5、病毒每 20 秒检测一次网络状态，当网络可用时，病毒会在本地的 UDP/69 端口上建立一个 tftp 服务器，并启动一个攻击传播线程，不断地随机生成攻击地址进行攻击，另外该病毒攻击时，会首先搜索子网的 IP 地址，以便就近攻击。

6、当病毒扫描到计算机后，就会向目标计算机的 TCP/135 端口发送攻击数据。

7、当病毒攻击成功后，便打开目标计算机的 TCP/4444 端口作为后门，并绑定 cmd.exe，然后蠕虫会连接到这个端口，发送 tftp 命令，回连到发起进攻的主机，将 msblast.exe 传到目标计算机上并运行。

2.2.4 He4Hook

Windows rootkit 就是 Windows 系统中的 rootkit，是一种程序或程序集，其用途是秘密控制被入侵的计算机的行为^[10]。它经常用来隐藏一个后门程序以及其它类似工具程序，使这些工具程序能在指定计算机中非法存在。当用户查询计算机的当前状况时，它通过隐藏和这些工具相关的所有信息来欺骗用户，使用户相信计算机未受到侵害。按照运行时的环境不同，Windows rootkit 分为两类：内核模式 rootkit 和用户模式 rootkit。内核模式 rootkit 驻留在内核态中，以内核驱动程序的方式存在于操作系统中；用户模式 rootkit 则运行在权限较低的用户态中。

He4Hook 是一个内核模式 Windows rootkit，可运行于 Windows NT4.0、Windows2000 中。在 www.rootkit.com 网站中，有它的完整源代码。He4Hook 在技术上很先进，而且也比较稳定。它的执行文件主要包括两个，He4HookInv.sys 和 He4HookControl.exe。其中，He4HookInv.sys 是内核模式驱动程序，He4HookControl.exe 是用于和 He4HookInv.sys 进行通信的控制台程序。它的主要特点使用 SystemLoadAndCallImage 技术而不是使用服务控制管理器 (SCM) 来装载驱动 He4HookInv.sys，并提供了两种文件系统挂钩方法，一种使用常规的挂钩系统服务地址表技术，另一种则使用很少见的挂钩文件系统驱动程序的技术，它可以隐藏或保护某些目录，受保护的目录在应用程序中无法访问。

2.2.5 冰河木马

冰河由两个程序组成：G_server.exe(服务端程序，即木马)和 G_client.exe(控

制端程序), 它的特性有:

- 1) 自动跟踪目标机屏幕变化, 同时可以完全模拟键盘及鼠标输入。
- 2) 记录各种口令信息, 包括开机口令、屏保口令、各种共享资源口令及绝大多数在对话框中出现过的口令信息, 记录击键输入。
- 3) 获取系统信息, 包括计算机名、注册公司、当前用户、系统路径、操作系统版本、当前显示分辨率、物理及逻辑磁盘信息等多项系统数据。
- 4) 限制系统功能, 包括远程关机、远程重启计算机、锁定鼠标、锁定系统热键及锁定注册表等多项功能限制。
- 5) 远程文件操作, 包括创建、上传、下载、复制、删除文件或目录、文件压缩、快速浏览文本文件、远程打开文件等多项文件操作功能。
- 6) 注册表操作, 包括对主键的浏览、增删、复制、重命名和对键值的读写等所有注册表操作功能。
- 7) 发送信息, 以四种常用图标向被控端发送简短信息。
- 8) 点对点通讯, 以聊天室形式同被控端进行在线交谈。
- 9) 邮件功能, 自动往设定的电子邮箱发送系统信息。

2.3 病毒的加载运行技术

通过对典型病毒的分析, 可以看出, 病毒想尽各种方法使自己能够得到加载运行, 在 Windows NT/2X/XP 中, 病毒的加载运行大体可分为下面六种:

1. 利用系统初始化文件实现病毒的自动启动

例如 win.ini 是保存在系统目录(%system%)下的一个初始化文件。系统在启动时会检索该文件中的相关项, 以便对系统环境进行初始设置。

在该文件中的[windows]数据段中, 有两个数据项“load=”和“run=”, 它们的作用就是在系统启动之后自动地装载和运行相关程序, 如果我们需要在系统启动之后装载并运行一个程序, 只要将该程序的全文件名添加到该数据项的后面, 系统启动后就会自动运行该程序。

2. 利用注册表实现病毒的自动启动

系统注册表保存着系统的软件、硬件及其它与系统配置有关的重要信息, 一台计算机系统的注册表一旦遭到破坏, 整个系统将无法运行。大多数病毒通过修改注册表来达到自动加载的目的^[1]。例如, “冲击波”、“求职信”、“爱虫”等。

① Run 注册键

Run 是自动运行程序最常用的注册键, 位置在:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

② RunOnce 注册键

安装程序通常用 RunOnce 键自动运行程序，它的位置在

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\
RunOnce

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunO
nce

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
CurrentVersion\RunOnceEx

③ RunServicesOnce 注册键

RunServicesOnce 注册键用来启动服务程序，启动时间在用户登录之前，而且先于其他通过注册键启动的程序。RunServicesOnce 注册键的位置在：

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunSe
rvicesOnce

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
CurrentVersion\RunServicesOnce。

④ RunServices 注册键

RunServices 注册键指定的程序紧接 RunServicesOnce 指定的程序之后运行，但两者都在用户登录之前。RunServices 的位置在：

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunSe
rvices

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\
CurrentVersion\RunServices

⑤ Userinit 键值，位置在：

HKEY_LOCAL_MACHINE\
Software\Microsoft\WindowsNT\CurrentVersion\Winlogon

存放在 Userinit 键值中的数据，也能够系统在启动时自动运行。通常该键值的数据为 userinit.exe，但这个键值中存放的数据允许指定用逗号分隔的多个程序名，例如 userinit.exe,OSA.exe。

⑥ Services 注册键

位置在：HKEY_LOCAL_MACHINE\System\Current Control Set\Services，系统启动后将自动加载该子键下的一些服务或驱动程序，这些服务或驱动程序的启动类型均为自动。

3. 利用启动文件夹实现病毒的自动启动

在 Windows 系统中，放入启动文件夹中的快捷方式总是自动启动。

① 当前用户专有的启动文件夹

这是许多应用软件自动启动的常用位置，Windows 自动启动放入该文件夹中的所有快捷方式。用户启动文件夹一般在：X:\Documents and Settings\<用户名字>\「开始」菜单\程序\启动，其中“<用户名字>”是当前登录的用户帐户名称。

② 对所有用户有效的启动文件夹

这是寻找自动启动程序的第二个重要位置，不管用户用什么身份登录系统，放入该文件夹的快捷方式也总是自动启动该文件夹一般在：

X:\Documents and Settings\All Users\「开始」菜单\程序\启动。

注：X 表示系统盘的盘符。

4. 修改文件关联实现病毒的加载运行

通常，对于一些常用的文件（例如，txt 文件），我们只要用鼠标双击文件名就可以打开这个文件。这是因为在系统注册表中，已经把这类文件与一个应用程序关联了起来（例如，notepad.exe 与 txt 文件关联）。有些病毒通过修改文件关联来达到加载的目的，例如冰河木马就是利用文本文件(.txt)的文件关联来加载自己。如果感染了冰河木马，则[HKEY_CLASSES_ROOT\txtfile\shell\open\command]中的键值不是“%SystemRoot%\system32\notepad.exe %1”，而是改为“sysexplr.exe”^[12]。

5. 利用系统漏洞实现病毒的加载运行

如果您的系统没有及时打上补丁的话，病毒就可以利用某些系统漏洞来加载运行。例如微软安全公告 MS00-052 中描述了这样一个系统漏洞：windows 系统在执行可执行程序时，会先搜索系统盘根目录下有没有同名的程序，如果有，就先执行该程序^[8]。“红色代码”病毒就是利用了上述这个漏洞，它把“explorer.exe”木马放在“C:\”和“D:\”的根目录下，这样木马就先于真正的“explorer.exe”被执行。如果没有安装 SP2 或者 MS00-052 中的补丁，就可能执行这个木马程序；否则，不会执行这个木马。

6. 利用其它进程来实现病毒的加载运行

这种病毒一般都是无进程 DLL 木马，它们以动态连接库的形式存在于系统中，并且会利用注册表来启动自己。例如，在注册表：

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows 的键值 AppInit_DLLS 中，可以存放一个或多个 DLL 的名字，根据 MSDN 文档，当一个使用 USER32.DLL 的应用程序被启动时，键值 AppInit_DLLS 中指明的 DLL 也被自动加载^[13]。在上一节中被剖析的 Win32.Troj.ADNavihelper，这个广告就是一个无进程 DLL 木马，它被注册成

BHO，使得每次开启 IE 浏览器时都会加载该广告木马。

从病毒加载运行的方法可以看出，如果我们对注册表的某些重要子键、键值进行实时监控，那么可以防止病毒以服务/驱动程序的形式安装到系统中，可以防止 DLL 木马注射到可信进程的地址空间中，也可以防止病毒修改文件关联等等，从而切断病毒的感染途径；如果程序的运行必须要得到用户认可的话，那么可以剥夺病毒的运行权利。

病毒的植入和对系统的攻击，离不开进程这个载体，因此如果我们控制住进程的行为就可以阻止病毒的入侵：首先程序的运行必须要得到用户的认可，其次根据访问控制规则来判断进程的当前行为是否合法，如果非法，及时阻止进程的当前行为。这就是入侵防护所要实现的目标。

2.4 本章小结

本章首先介绍了病毒、木马、蠕虫的定义，然后对几个典型的病毒如“红色代码”病毒、“冰河”木马等进行了详细的剖析，归纳总结了病毒、木马的植入技术和运行机理。病毒的植入和对系统的攻击，离不开进程这个载体，因此我们提出了通过进程防护来阻止病毒入侵的思想。

第三章 基于行为的入侵防护体系模型

3.1 PPDR安全理论模型

1985年,美国国防部的国家计算机安全中心(NCSC)发布了可信计算机安全评估准则(TCSEC)。这个准则的发布对操作系统、数据库等方面的安全发展起到了很大的推动作用,被称为信息安全的里程碑^[6]。

但是,TCSEC是基于主机/终端环境的静态安全模型建立起来的标准,是在当时的网络发展水平下被提出来的。随着网络的深入发展,这个标准已经不能完全适应当前的技术需要,无法反映分布式、动态变化、发展迅速的Internet安全问题。针对日益严重的网络安全问题和越来越突出的安全需求,“动态安全模型”应运而生。

传统的信息安全技术都集中在系统自身的加固和防护上。比如,采用B级操作系统和数据库,在网络出口配置防火墙,在信息传输和存储方面采用加密技术,使用集中的身份认证产品等等。单纯的防护技术容易导致系统的盲目建设,这种盲目包括两方面:一方面是不了解安全威胁的严峻性,不了解当前的安全现状;另一方面是安全投入过大而又没有真正抓住安全的关键环节,导致不必要的浪费。所以,现在安全领域强调安全防护的体系建设,强调全面的解决方案,强调水桶原理和基于时间的动态的安全防护理论^[14]。

PPDR模型

举例来看,一个水库的大坝到底应当修多高?大坝有没有漏洞?修好的大坝现在是否处在危险的状态?实际上,我们需要相应的检测机制,比如,利用工程探伤技术检查大坝修建和维护是否保证了大坝的安全;观察当前的水位是否超出了警戒水位。这样的检测机制对保证大坝的安全至关重要。

当发现问题之后就需要迅速做响应,比如,立即修补大坝的漏洞并进行加固;如果到达警戒水位,大坝就需要有人24小时监护,还可能需泄洪。这些措施实际上就是一些紧急应对和响应措施。

在信息安全领域,对安全问题的理解也是类似的。相应提出的就是PPDR模型(见图3-1)。PPDR动态安全模型的产生是与20世纪90年代中期开始的防黑客技术共同起步的,在这方面最突出的就是ISS(Internet Security Systems)公司及其创始人Christopher Klaus先生。PPDR模型是动态安全模型(可适应网络安全模型)的代表性模型。

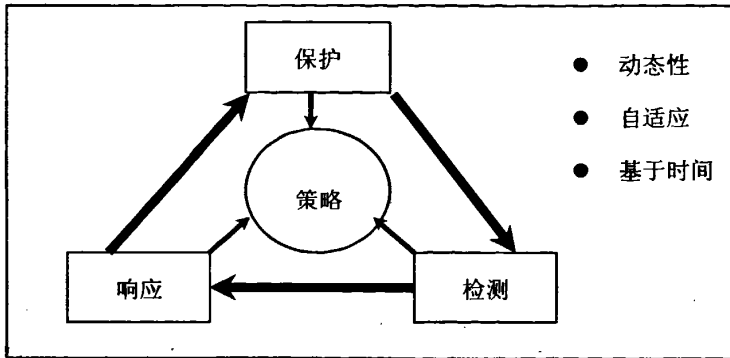


图 3-1 PPDR 安全理论模型

- Policy（安全策略）
- Protection（防护）
- Detection（检测）
- Response（响应）

PPDR 模型由防护、检测和响应组成了一个完整的、动态的安全循环，在安全策略的指导下保证信息系统的安全，PPDR 安全模型的特点就是动态性和基于时间的特性^[15]。

PPDR 模型阐述了这样一个结论：安全的目标实际上就是尽可能地增大保护时间，尽量减少检测时间和响应时间。

- **Policy:** 安全策略是 PPDR 安全模型的核心，所有的防护、检测、响应都是依据安全策略实施的，安全策略为安全管理者提供管理方向和支持手段。
- **Protection:** 保护通常是通过采用一些传统的静态安全技术和方法来实现的，主要有防火墙、加密、认证等方法。
- **Detection:** 在 PPDR 模型中，检测是非常重要的一个环节，检测是动态响应和加强防护的依据，它也是强制落实安全策略的有力工具，通过不断地检测和监控网络和系统，来发现新的威胁和弱点，通过循环反馈来及时作出有效的响应。
- **Response:** 紧急响应应在安全系统中占有最重要的地位，是解决安全潜在最有效的办法。从某种意义上讲，安全问题就是解决响应和异常处理问题。要解决好紧急响应问题，就是制定好紧急响应的方案，做好紧急响应方案中的一切准备工作^[16]。

3.2 基于行为的入侵防护体系模型

随着互联网的普及,安全威胁出现了许多新变化,病毒和攻击不仅比以往更复杂和更快速,而且它们利用新漏洞的时间间隔越来越短,因此,基于特征扫描的防毒软件显然不能满足这种新变化,必须采用 PPDR 安全理论模型来防止病毒的入侵。基于行为的入侵防护体系 (Invasion Guard Base On Behaviors 以下简称 IGBOB) 模型正是 PPDR 安全理论模型在防止病毒入侵方面的具体实现。

3.2.1 模型的结构和组成

IGBOB 就是根据访问规则库来判断进程的行为 (例如进程的建立、进程的修改、进程的终止) 是否允许。IGBOB 模型完全符合标准的 PPDR 模型,即 Policy(策略)、Protection(防护)、Detection(检测)、Response(响应)。防护、检测、响应组成了一个完整的、动态的安全循环,IGBOB 模型如图 3-2 所示:

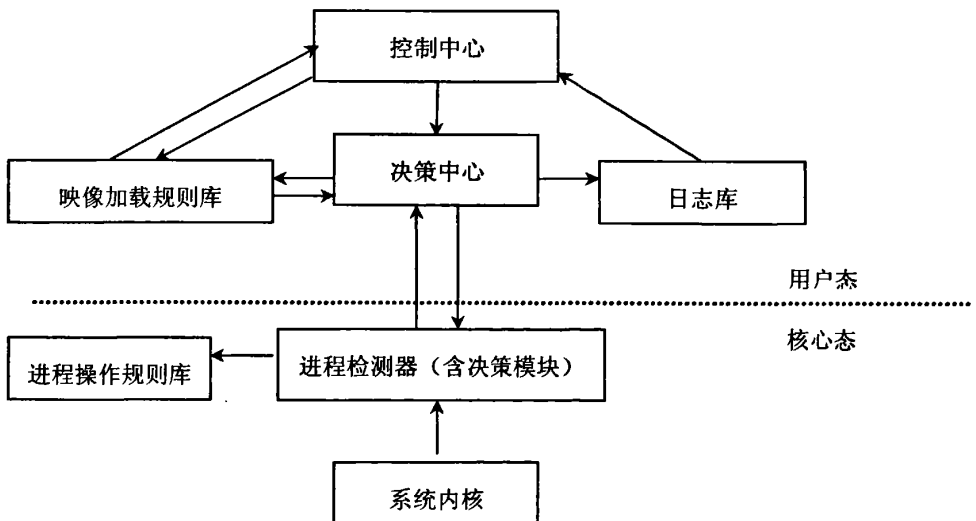


图 3-2 基于行为的入侵防护体系模型

一、策略

访问控制规则是决定进程的行为是否允许、程序能否加载运行的重要依据。IGBOB 的访问控制规则库有两个：**映像加载规则库**和**进程操作规则库**。

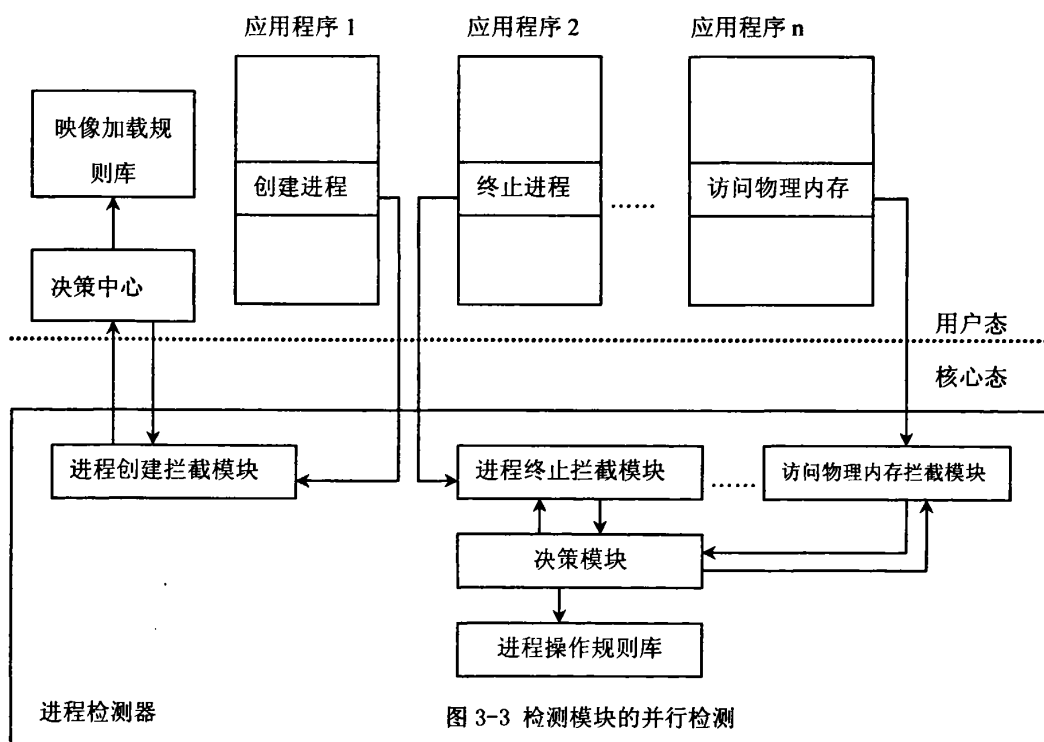
映像加载规则库工作在用户层,它是控制程序能否加载运行的重要依据。其

控制规则的格式用五元组表示为 (O, P1, P2, A, H), O 表示可执行程序的路径及名字; P1 表示 IGBOB 对由 O 加载的进程实施的保护措施(例如,不能对该进程进行终止、修改等操作); P2 表示 IGBOB 给由 O 加载的进程赋予的权限(例如,该进程可以对其它进程进行读、修改等操作); A 表示 IGBOB 将允许还是拒绝 O 的运行; H 表示由 O 计算出的文件校验和。

进程操作规则库工作在核心层,它是控制一个进程的某个操作是否允许的唯一依据。设计进程操作规则库的目的是为了提高决策的速度,因为进程操作规则库是映像加载规则库的一个子集。其控制规则的格式用四元组表示为 (O, ID, P1, P2), O 表示可执行程序的路径及名字, ID 表示进程标识符; P1 表示 IGBOB 对该进程实施的保护措施; P2 表示 IGBOB 赋予该进程的权限。

二、检测

进程检测器工作在核心态,实际上它已嵌入到系统内核中,成为操作系统的一部分。进程检测器拦截与进程有关的操作,例如:建立进程、修改进程、终止进程、直接访问物理内存、安装驱动或服务,并提取与之相关的映像文件名、进程标识符作为决策因子。



进程检测器由若干个检测模块组成,一种进程行为的拦截由一个或多个检测

模块来负责，这样做的目的是为了提高拦截的速度，当有不同的进程行为同时发生时，相应的检测模块可以同时进行拦截。也就是说，检测模块可以并行工作，其流程如图 3.3 所示。

三、响应

1、对创建进程的响应

决策中心根据决策因子来查询映像加载规则库，通过规则的匹配情况来决定做出何种响应，并将响应结果下传给进程检测器。在下列情况下，决策中心所作的决策需要用户的参与（假设映像文件 A 需要加载运行）。

① 控制规则库中找不到 A 的规则

② 控制规则库中有 A 的规则，但 A 从上次运行后 A 被修改过。

当需要用户参与决策时，决策中心询问用户是允许还是拒绝 A 的运行，用户有四种选择：永远允许）、ALLOW ONE（允许运行一次）、BLOCK（永远阻止）、BLOCK ONE（阻止运行一次）。如果用户选择的是 ALLOW 或 BLOCK，那么决策中心还要在映像加载规则库中添加规则或修改规则。

2、对其它进程操作的响应

进程检测器如果拦截到一个进程访问另一个进程或者进程直接访问物理内存等操作，则直接调用核心态中的决策模块来进行决策，决策模块查询进程操作规则库，通过规则的匹配情况来决定是允许还是拒绝该操作。

四、保护

1、未被用户许可的程序不让运行

IGBOB 对计算机中的程序进行控制，只有被用户认可的程序才能执行，这样可以有效阻止蠕虫和木马在后台悄悄运行。

如果决策中心对创建进程的响应为“拒绝”，进程检测器将终止进程的创建。

2、保护 Windows 系统进程和用户可信进程

IGBOB 通过设置保护权限来保护目标进程。例如，如果目标进程被设置成防进程终止，那么即使攻击进程拥有终止其它进程的权限，也无法终止目标进程。这样保护了 Windows 系统进程和用户可信进程免受其它进程，服务，以及系统上的其它形式的可执行代码的攻击。

3、防止 Windows 系统资源被恶意修改

IGBOB 通过设置进程的权限来防止进程直接访问物理内存、安装驱动/服务等、从而防止 Windows 系统资源被恶意修改。

IGBOB 的组成

IGBOB 将由进程检测器、决策中心和控制中心三部分组成。

进程检测器是一个内核驱动程序，它负责拦截进程操作，获取决策的前提条

件，防护 IGBOB 自身安全，维护进程操作规则库，对进程间的操作进行决策和响应等。

决策中心是一个服务，它根据决策的前提条件来查询映像加载规则库，通过规则的匹配情况或用户的抉择来确定进程能否创建，并将决策结果下传给进程检测器。同时，它还负责把允许或拒绝进程操作的情况记录到日志库，以供用户查询和分析。

控制中心是一个 Windows 应用程序，负责映像加载规则库的维护、日志查询、进程检测器的启动和停止等。

IGBOB 所涉及的理论和实现技术适用于 Window NT/2000/XP/2003，不适用于 Window 98/Me 等，因此下面提及的 Windows 是专指 Window NT/2000/XP/2003。

3.2.2 IGBOB模型的优缺点

一、 IGBOB 模型的优点

1、鲁棒性

该模型通过访问控制规则来拒绝未经用户认可的程序运行、保护系统进程和用户可信进程、控制对系统资源的访问，这种措施对未知病毒和已知病毒的变种都具有较高的防护作用，因而模型具有鲁棒性。

2、动态性

访问控制规则是 IGBOB 模型的核心，所有的防护、检测、响应都是依据访问控制规则实施的。访问控制规则不是一成不变的，而是随着时间动态发生变化，当某个程序想要运行，而在规则库中找不到匹配的规则时，模型根据用户的抉择自动生成新的规则添加到规则库中，这样在访问控制规则的指导下保证了主机系统的安全。

3、全面性

IGBOB 模型的进程检测器嵌入到系统内核中，成为操作系统的一部分，而应用程序/服务的运行离不开操作系统的支持（如图 3-4），因此任何进程的攻击行为都在进程检测器的掌控之中。

4、高效性

除了进程创建需要在用户态中决策外，其余进程行为是否允许都是调用核心态中的决策模块来决定的，这样大大减少了决策中心和进程检测器之间的通信量，以及因协调两者之间的同步所花费的时间。另外，检测模块的并行工作机制，提高了拦截和决策进程行为的速度。

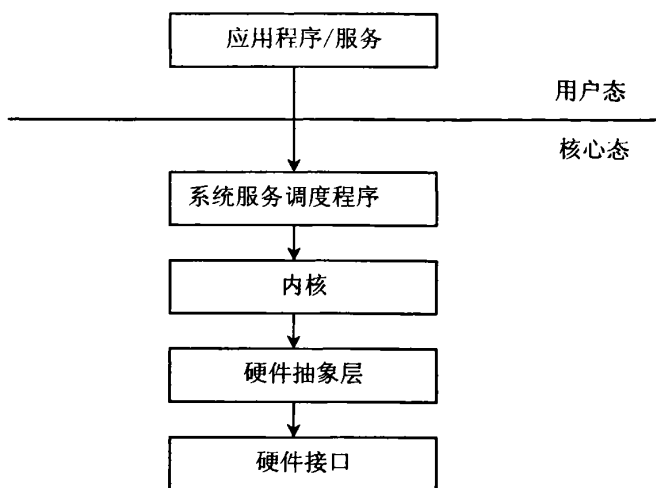


图 3-4 应用程序/服务的执行过程

二、 IGBOB 模型的缺点

1、 应用范围的局限性

IGBOB 所涉及的理论和实现技术适用于 Window NT/2000/XP/2003，不适用于 Window 98/Me，因此用该模型实现的原型系统在应用范围方面具有一定的局限性。

2、 对使用对象有一定要求

IGBOB 原型系统虽能防住未知病毒、木马和其它恶意代码的攻击，但对使用对象有一定要求，当一个新的普通程序、病毒、木马需要加载运行时，完全由使用对象自己判断是允许还是拒绝运行。

3.3 IGBOB模型的工作流程

上一节详细介绍了 IGBOB 模型的总体结构，分析了该模型在策略、检测、响应和保护这四个方面的工作原理以及该模型的优缺点，本节将以控制进程创建、控制进程终止为例，简要介绍该模型的工作流程。

3.3.1 “控制进程创建”的工作流程

“控制进程创建”的工作流程如图 3-5 所示，下面以 winexcel.exe 为例，简

要说明其工作流程：

- 1、当用鼠标双击 excel 文档或者直接加载 winexcel.exe 时，Windows 资源管理器调用 Windows API 函数 CreateProcess 来为 winexcel.exe 创建一个进程。
- 2、CreateProcess 调用内核中相应的系统服务，但该系统服务已被“进程创建拦截模块”所拦截。
- 3、“进程创建拦截模块”获取被创建进程（winexcel.exe）的映像文件名（包括所在路径）、命令行参数等决策因子，并唤醒决策中心进行响应。
- 4、决策中心根据决策因子查询映像加载规则库，如果存在匹配的规则，那么决策结果有该规则确定；如果找不到匹配的规则，那么决策结果由用户确定。“进程创建拦截模块”根据决策中心的决策结果来决定是否允许创建新进程（winexcel.exe）。

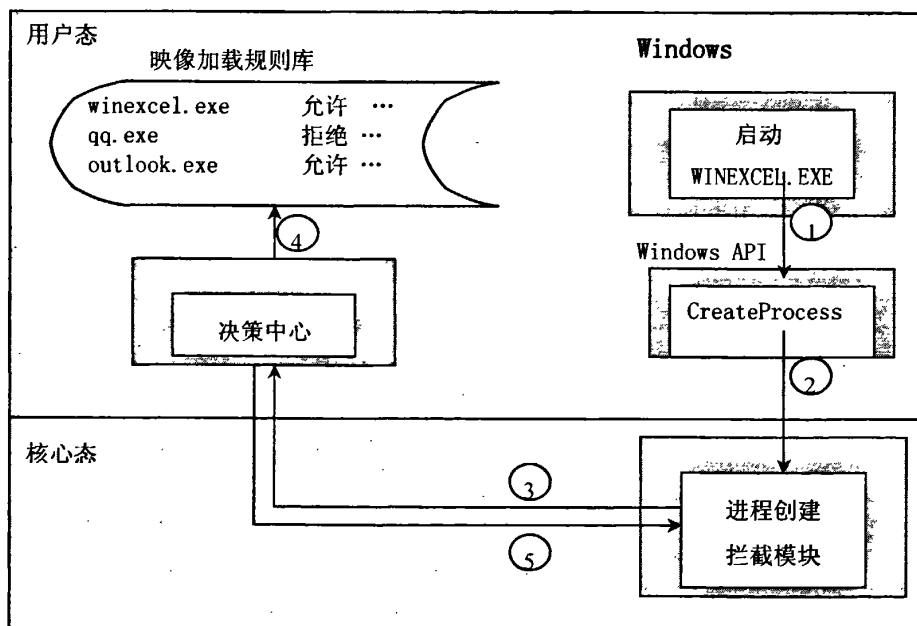


图 3-5 “控制进程创建”的工作流程

3.3.2 “控制进程终止”的工作流程

- 1、“控制进程终止”的工作流程如图 3-6 所示，下面以 mytaskmgr.exe 为例，简要说明其工作流程：
- 2、mytaskmgr.exe 调用 Windows API 函数 TerminateProcess 来终止指定的目标

- 进程。
- 3、TerminateProcess 调用内核中相应的系统服务，但该系统服务已被“进程终止拦截模块”所拦截。
 - 4、“进程终止拦截模块”获取进程 mytaskmgr.exe 的 ID、目标进程的 ID,然后调用内核态中的决策模块进行决策。
 - 5、决策模块首先根据进程 mytaskmgr.exe 的 ID 来查询进程操作规则库，判断它是否有终止其它进程的权利；然后根据目标进程的 ID 来查询进程操作规则库，判断它是否允许其它进程来终止该进程，决策模块根据先后两次查询情况来决定决策结果。
 - 6、“进程终止拦截模块”根据决策结果来决定是否允许 mytaskmgr.exe 终止目标进程。

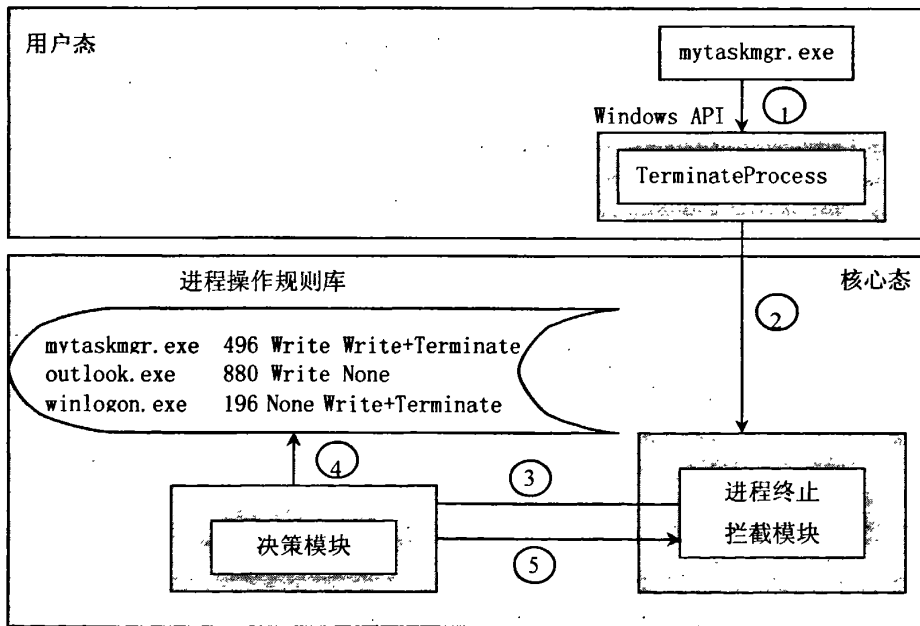


图 3-6 “控制进程终止”的工作流程

3.4 本章总结

本章首先介绍了标准的动态安全防护模型 PPDR，然后提出了一个完全符合 PPDR 的基于行为的入侵防护系统 (IGBOB) 模型，描述了该模型的结构以及在策略、检测、响应和保护四个环节的工作原理，分析了该模型的优缺点，并介绍了

该模型的工作流程。

要想控制住入侵进程的行为，首先必须实现对入侵进程行为的拦截，进程行为的拦截通常采用修改系统函数执行路径的内核修补技术。

第四章 进程行为的拦截技术

实时拦截技术其实并非什么新技术，早在 DOS 编程时代就有之。只不过那时人们没有给这项技术冠以这样专业的名字而已。早期在各大院校机房中普遍使用的硬盘写保护软件正是利用了实时拦截技术。硬盘写保护软件一般会将自身写入硬盘零磁头开始的几个扇区（由 0 磁头 0 柱面 1 扇最开始的 64 个扇区是保留的，DOS 访问不到）并修改原来的主引导记录以便使启动时硬盘写保护程序可以取得控制权。引导时取得控制权的硬盘写保护程序会修改 INT 13H 的中断向量指向自身已驻留于内存中的钩子代码以便随时拦截所有对磁盘的操作^[17]。

进程行为拦截技术采用的是更改系统函数执行路径的内核修补技术。通过对系统内核进行修补，加入自己的特殊代码，主动与操作系统进行无缝连接，从而达到拦截进程行为的目的。

由于进程行为拦截技术涉及到 Windows 操作系统内部知识，因此，下面对必需的相关知识进行简要的介绍。

4.1 保护模式的权限级别

4.1.1 保护模式的权限级别

当前流行的 Windows 操作系统运行在保护模式之下，在保护模式下，所有的应用程序都有权限级别（Privilege Level，简称为 PL），这个权限级别按优先次序分为四等：0，1，2 和 3，其中 3 特权最低，0 特权最高^[18]。特权级环如图 4-1 所示：

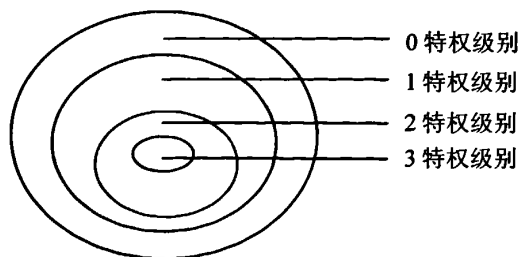


图 4-1 特权级别环

4.1.2 系统服务 (System Service)

系统服务,也叫系统调用、系统函数,是由 Windows 内核提供的一些实现操作系统基本功能的运行于内核态的函数。系统服务只能从内核模式下访问,用户模式应用程序不能直接访问系统服务^[19]。系统服务可通过它的函数名或是代号(称为服务号 ServiceID 或分配号 DispatchID)来访问。代号在 0 到 0xFFFF 之间的系统服务是常规的系统服务,实现代码在 ntoskrnl.exe 文件中,用于实现文件管理、进程管理、线程管理、内存管理、对象管理等操作系统的基本功能。代号在 0x1000 到 0x1FFF 之间的服务是专门用于处理图形与用户接口的相关函数,这些函数可以在用户模式下实现,但为了提高系统在图形处理时的性能,防止出现频繁的状态切换(用户态和内核态之间的转换),微软公司从 Windows NT4.0 之后,将这部分代码也放到了内核态中,具体的实现代码是在 Win32k.sys 文件中。

系统服务在运行期间长驻于系统内存区,不会被换出到磁盘中。在系统内存区中,专门设立了一些指向所有系统服务的函数指针表,以便快速访问它们。这些数据结构的具体情况下面将做详细介绍。

4.1.3 系统服务调度

在 Intel X86 处理器上执行 int 2Eh 指令,会引起系统服务调度,它会使系统陷入系统服务调度程序。被传递的数值参数指明了将要调用的系统服务号,内核使用这个参数查找位于系统服务调度表中的系统服务信息。这个表和中断调度表相似,只是每个入口包含了一个指向系统服务的指针,而不是一个指向中断处理例程的指针。如图 4-2 所示:

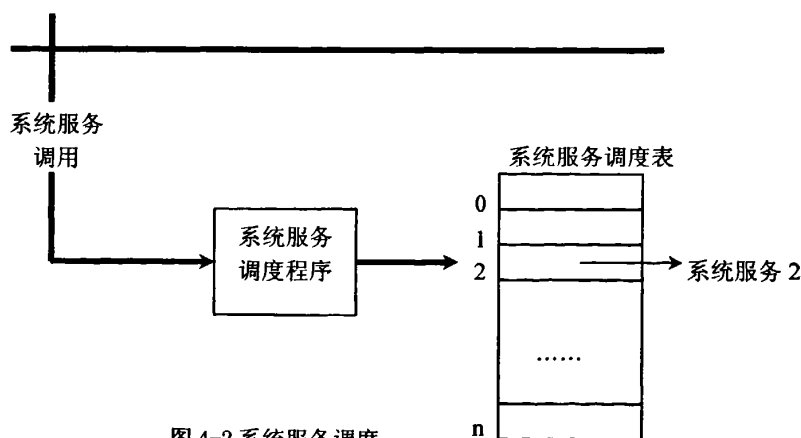


图 4-2 系统服务调度

系统服务调度程序将校验参数，并将调用者的参数从线程的用户堆栈复制到它的核心堆栈中，然后再执行系统服务。如果传递给系统服务的参数指向用户空间的缓冲区，则核心态代码在访问缓冲区之前，会查明这些缓冲区的可访问性^[20]。

用于实现操作系统基本功能的系统服务调度指令存在于系统库 ntdll.dll 中，Win32 子系统动态链接库 (Kernel32.dll、Advapi32.dll 等) 通过调用 ntdll.dll 中的函数来实现用户程序可用的 Win32 API。这里有一个例外是 Win32 USER 及 GDI，出于效率的考虑，其中的系统服务调度指令是在 USER.DLL 及 GDI.DLL 中直接实现的。

系统服务的调度过程入图 4-3 所示。

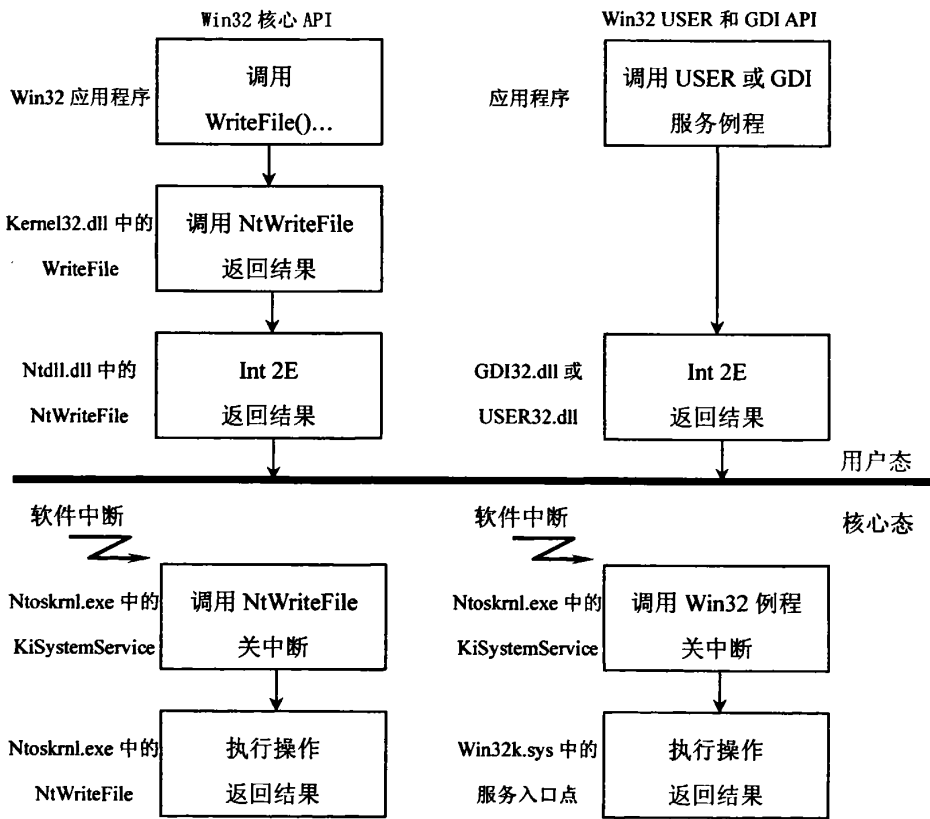


图 4-3 系统服务调度

Kernel32.dll 中的 Win32 WriteFile 函数调用 ntdll.dll 中的 NtWriteFile 函数，后者接着执行相应的指令引起系统服务陷阱，同时传递代表 NtWriteFile 的系统服务号。系统服务调度程序 (Ntoskrnl.exe 中的 KiSystemService) 随后调用实时的 NtWriteFile 来处理 I/O 请求。对于 Win32 USER 和 GDI 函数，系统服务调度程

序调用 Win32 子系统可加载内核模式部分 Win32k.sys 中的函数^[20]。

4.1.4 系统服务描述符表、系统服务表和系统服务地址表

在 Windows NT 系列操作系统中，有两种类型的系统服务，一种实现在 ntoskrnl.exe 中，是常用的系统服务；另一种实现在 win32k.sys 中，是一些与图形显示及用户界面相关的系统服务。这些系统服务在系统执行期间长驻于系统内存区中，并且它们的入口地址保存在两个系统服务地址表 KiServiceTable 和 Win32pServiceTable 中^[21]。而每个系统服务的入口参数所用的总字节数则分别保存在另外两个系统服务参数表 (ArgumentTable) 中 (见图 4-4)。

系统服务地址表和系统服务参数表是一一对应的，每个系统服务表 (System Service Table, 以下简称为 SST) 都指向一个地址表和一个参数表。在 Windows

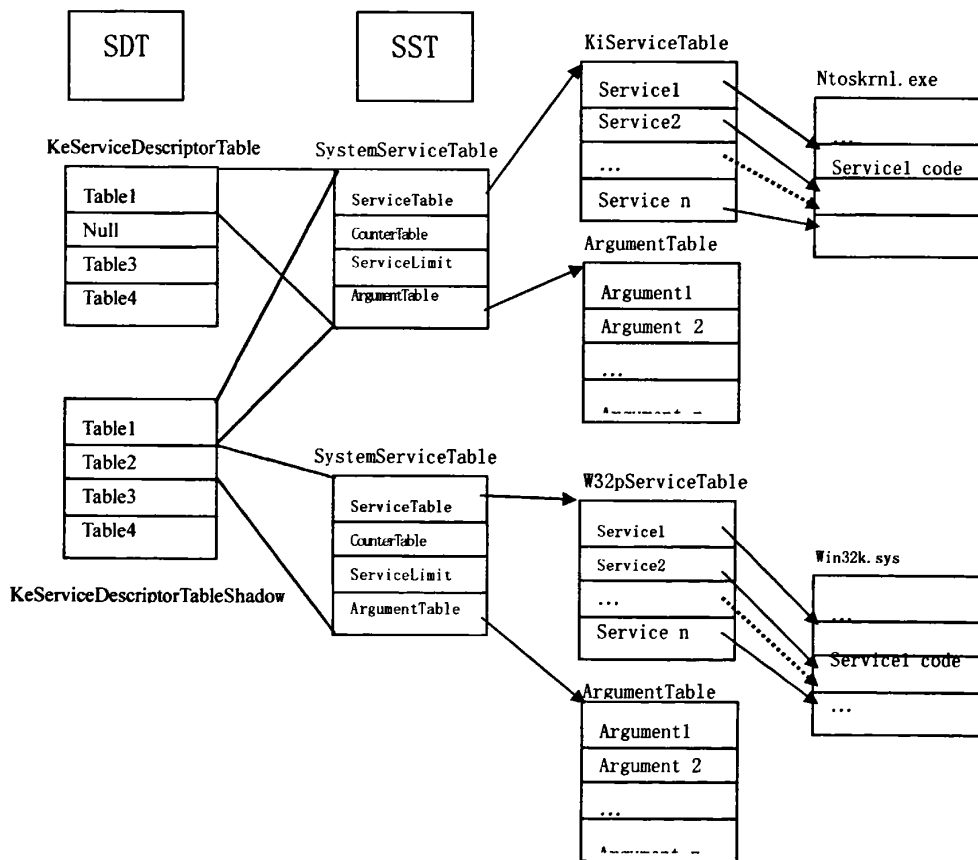


图 4-4 SDT、SST、KiServiceTable 关系图

2000 系统中，只有两个 SST。一个 SST 指向了 KiServiceTable，而另一个 SST 则指向了 Win32pServiceTable。

所有的 SST 都保存在系统服务描述符表 (System Service Descriptor Table, 以下简称 SDT) 中。

图 4-4 表示 SDT、SST、KiServiceTable 这三种表的相互关系:

4.1.5 进程与线程

进程通常被定义为一个正在运行的程序的实例, 它由两部分组成: 进程内核对象和地址空间。内核对象是操作系统用来管理此进程的信息块。地址空间包含所有可执行模块或 DLL 模块的代码和数据, 包含动态分配的内存空间(如线程堆栈和堆分配空间)。进程可拥有一个或多个线程, 由线程负责执行进程地址空间中的代码。一个 Windows 应用程序, 在运行之后, 会在系统之中产生一个进程。Windows 系统会为每个进程都分配一个虚拟的内存空间, 一切相关的程序操作, 都会在这个虚拟的空间中进行。多个进程之间往往通过内存映射文件共享数据 [2]。

线程是进程地址空间中代码的具体执行者。每个线程都有它自己的 CPU 寄存器和堆栈, 用于执行代码和保存数据。进程创建一个线程后会在其内存空间中给该线程分配堆栈。一个进程拥有的多个线程之间可以同步执行多种操作, 一般来讲, 线程之间是相互独立的, 当一个线程发生错误的时候, 并不一定会导致整个进程的崩溃。线程可以访问同进程中其它线程的堆栈。

系统中的所有运行片段都会运行于某一进程的地址空间内。应用程序代码运行于自己的进程空间内。操作系统中完成各种管理和服务的各种模块代码和各种驱动程序运行于相应的系统进程内。

在内核中, 进程和线程结构的关系如下图 4-5 所示:

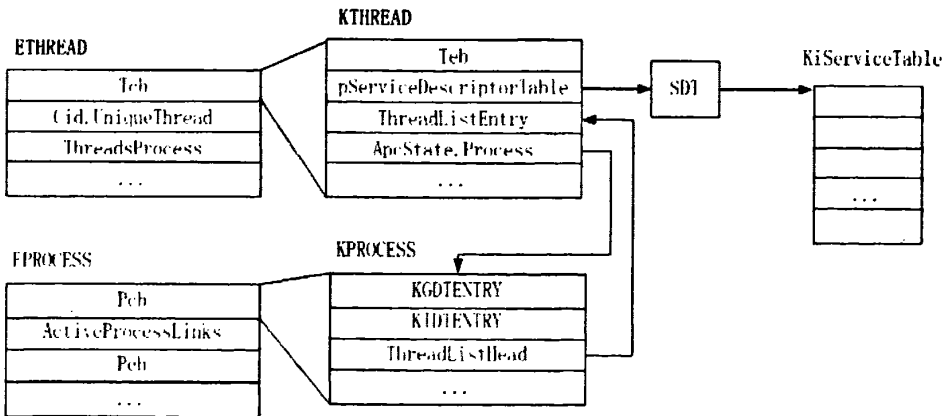


图 4-5 内核中进程、线程结构关系图

其中, KTHREAD 和 ETHREAD 是线程结构, KPROCESS 和 EPROCESS 是进程结构, 每个进程可能包含多个线程, 但每个线程都只能属于某一个进程。在 KTHREAD 中, 有一个项 pServiceDescriptorTable, 它指向了系统服务描述符表 SDT, 通过这个指针, 线程就可以访问到所有的系统服务。

4.1.6 挂钩和钩子函数

挂钩, 也叫挂载、挂接、勾连, 是一种特殊的函数运行机制。挂钩的本质是用一些特殊的函数(钩子函数), 通过某种手段, 替换系统中原有的各种函数。当这些系统中原有的函数被调用时, 钩子函数就会执行^[23]。

钩子函数就是用来替换系统中的现有函数的那些特殊函数, 当这些系统中原有的函数被调用时, 钩子函数被动执行。

通过挂钩, 可以截获系统中发生的各种函数调用, 并在这些函数执行前或执行后进行相应的处理(执行钩子函数)。挂钩的种类很多, 有针对系统消息的挂钩、有针对 Win32 API 的挂钩、还有针对内核级系统函数的挂钩。每种挂钩可以截获相应种类的函数调用, 当用户进程或操作系统执行这些被挂钩的函数调用时, 实际上是在调用钩子函数。此时, 钩子函数有三种可选的处理方法:

- 1、执行一些钩子函数自定义的操作, 然后再调用挂钩前的原始函数;
- 2、先调用挂钩前的原始函数, 然后对该函数的返回结果进行处理;
- 3、强制结束此函数的执行。

4.1.7 用户空间和内核空间

默认情况下, 在 Windows NT 系列操作系统中, 每个进程都可以使用 4G 虚拟内存空间。4G 空间的分配情况如图 4-6 所示, 整个 4G 虚拟内存空间分为两部分, 用户空间和内核空间。用户进程使用的空间叫用户空间, 位于 4G 虚拟内存空间中的前 2G 中, 即 0x00000000-0x7fffffff。内核模块及各种内核数据结构使用的地址空间叫内核空间, 位于 4G 内存空间中的后 2G 中, 即 0x80000000-0xffffffff。

每个进程都独自拥有自己的用户空间, 但是所有进程都共享同一个内核空间。每个进程只可以修改自己用户空间中的内容, 不可直接访问其它进程的用户空间。对某个进

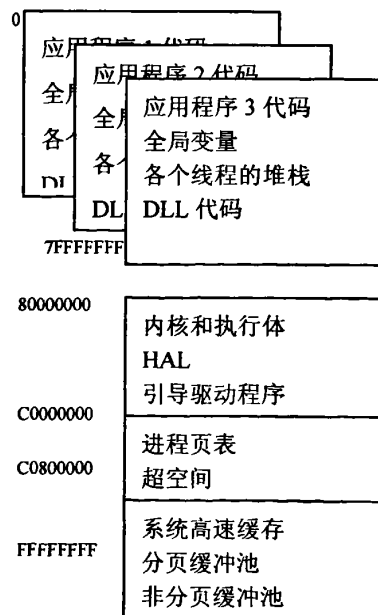


图 4-6 4G 虚拟地址空间布局

程的用户空间的修改只会影响到这个进程，不会影响其它进程。而对内核空间的修改则是全局性的，即只要有一个进程修改了内核空间，所有的进程都会受到影响。

4.2 内核修补技术

进程行为拦截采用的是更改系统函数执行路径的内核修补技术，通过对系统内核进行修补，加入自己的特殊代码，从而达到拦截进程行为目的。

通过 4.1.3 节的介绍，参考图 4-3，可以清楚地看出系统中各种函数的执行路径。只要在这条路径上修改任一个点，就可以改变系统中原有的各种函数的执行路径。改变函数执行路径的方法很多，下面介绍如何在内核模式下更改系统函数执行路径。

4.2.1 挂钩SDT/SST/KiServiceTable

这三种数据结构的具体内容参见 4.1.4，它们之间的关系参见图 4-4。因为这些数据结构都处于系统空间中，对它们的访问只能在内核程序中进行。

1、挂钩系统服务地址表（KiServiceTable）

这种方法修改系统服务地址表，将某些函数地址改为新的函数地址。

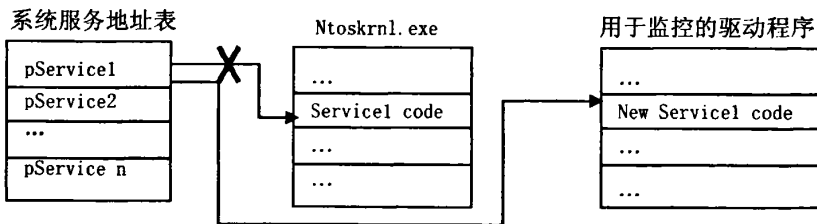


图 4-7 挂钩 KiServiceTable

假设用于监控的内核驱动程序的名字为 `mon.sys`，从图 4-7 可以看出，系统服务地址表 `KiServiceTable` 中存放着系统服务的函数指针，也就是系统函数的入口地址。上图以修改 `Service1` 为例。正常情况下，对 `Service1` 这个系统服务函数访问时，会从系统服务地址表中的对应项中取到的 `Service1` 的函数指针 `pService1`，该指针指向了此系统函数在 `Ntoskrnl.exe` 中的实现代码 `Service1 code`。挂钩 `KiServiceTable` 时，系统服务地址表中的函数指针 `pService1` 被修改为指向 `mon.sys` 中的一段代码 `New Service1 code`。

这样，挂钩 `KiServiceTable` 后，所有对系统函数 `Service1` 的访问都将被转到

mon.sys 中的 New ServiceIcode 处。

由于原理清晰，而且实现起来简洁、高效，许多用于监控的内核驱动程序都使用了这种方法，IGBOB 采用该方法来拦截进程的行为。

2、 挂钩系统服务表 (SST, System Service Table)

此方法修改 SST 中的 ServiceTable 的指针，使其指向一个新的系统服务地址表。详情见图 4-8。

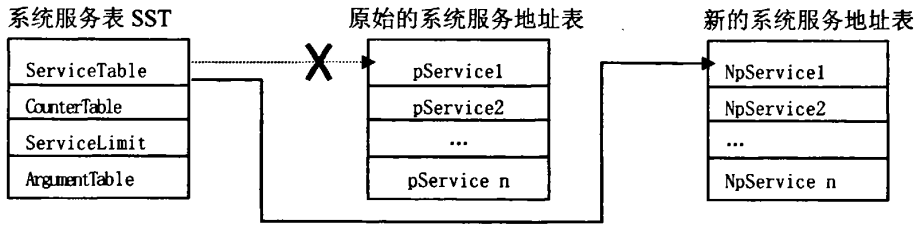


图 4-8 挂钩 SST

既然修改系统服务地址表中的内容可以更改系统函数的执行路径，那么，用一个新的系统服务地址表来代替原始的系统服务地址表，也能达到同样的效果。

这样做需要如下两步：

首先，在内核地址空间中申请一块区域，在该区域中写入新的系统服务地址表。这个新表的尺寸应该与原始的系统服务地址表一样大。然后，将 SST 中的 ServiceTable 指针指向新的系统服务地址表。这样，Windows 操作系统在每次查找系统服务的函数地址时，都会从这个新表中查找。

这种方法要比前挂钩 KiServiceTable 的方法麻烦一些。主要的困难是得在内存区的系统空间中找一块合适区域，以存放新的系统服务地址表。而且，使用这种方法挂钩时，得同时更改保存于 KeServiceDescriptorTableShadow 中的 SST。

3、 挂钩系统服务描述符表 (SDT, Service Descriptor Table)

在每个 KTHREAD 结构中，都有一个指向系统服务描述符表的指针项 pServiceDescriptorTable，运行中的线程通过这个指针来找到 SDT，然后再从 SDT 中找到具体的系统服务函数的地址(参见 4.1.4、4-3)。挂钩 SDT 的方法如图 4-9 所示：

其中，虚线表示指针被改动前所指的的位置。从图中可以看出，各个线程结构 KTHREAD 本来都是指向系统中正常的 KSDT 或 KSDTS，然后通过这两个描述符表找到具体的系统服务表。挂钩 SDT 时，将每个线程结构 KTHREAD 中的 pServiceDescriptorTable 指针指向一个新的 SDT，这个新的 SDT 又指向了一个系统服务地址表 New KiServiceTable，在这种情况下，各个线程在访问系统服务时，就会从 New KiserviceTable 中查找系统服务的函数地址，而不是查找原始的

KiServiceTable 或是 W32pServiceTable。

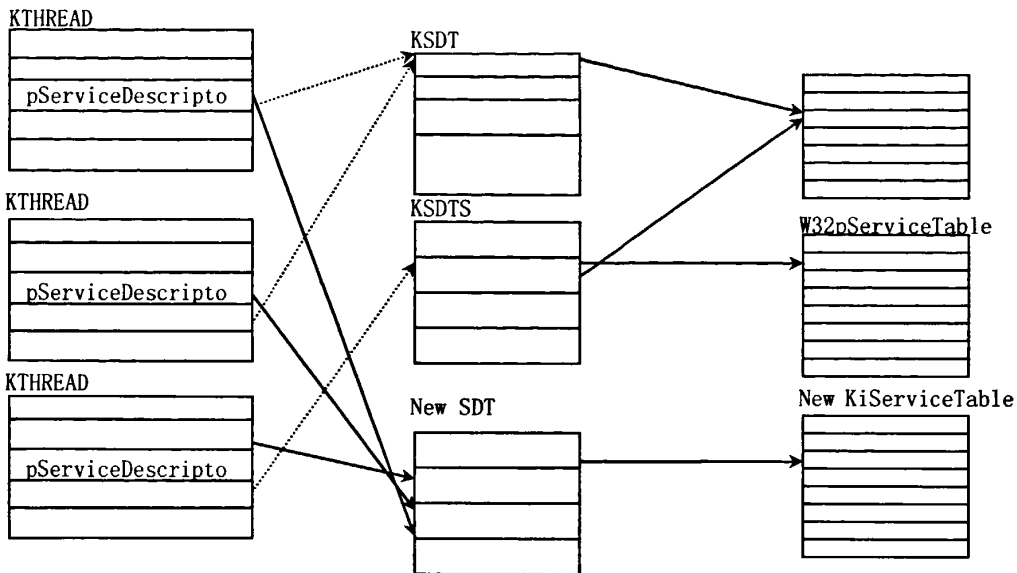


图 4-9 挂钩 SDT

4.2.2 挂钩 2Eh号中断处理程序

所有的中断处理程序的入口点都存放在中断描述符表（Interrupt Descriptor Table, 简称 IDT）中，INT 2Eh 中断也不例外。由于从用户模式发来的系统调用只能从 2Eh 中进入内核模式，所以，挂钩这个关键的中断处理程序，就能截获所有用户程序的系统调用，从而改变这些系统调用的执行路径。

从图 4-10 可知，在挂钩 IDT 时，只需将中断描述符表中的第 2Eh 项的内容改为指向一个新的中断处理函数 New KiSystemService，即可挂钩 2Eh 中断处理程序，控制所有从用户程序发出的系统调用请求。

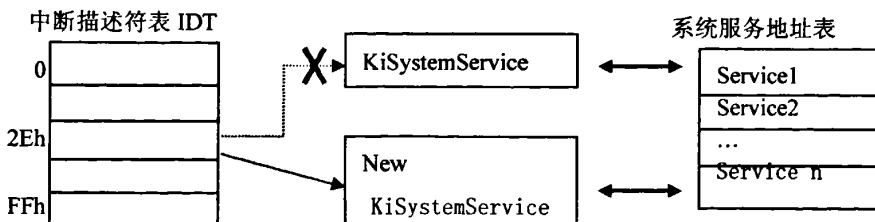


图 4-10 挂钩 IDT

4.3 进程的攻击行为

这里所说的进程攻击行为，不是指可疑进程如何破坏操作系统，而是指该可疑进程对其它进程可能发起的攻击或者获取其它进程的敏感信息。进程的攻击行为有：终止进程、挂起进程、修改进程、内核 rootkit 木马安装、防火墙旁路攻击、口令偷窃器和键盘记录器、物理内存恶意修改、windows 文件保护攻击等等。本小节将对这些攻击进行简单介绍。

1、终止进程

这是可疑进程对目标进程所发起的最常见的攻击方法，可疑进程可以采用多种方法来终止目标进程，其中最常用的是调用 kernel32.dll 中的函数 TerminateProcess。

主要调用的 API 函数：kernel32.dll: OpenProcess、TerminateProcess

user.dll: 未公开的 EndTask、

主要调用的系统服务：ntoskrnl.exe: NtOpenProcess、NtTerminateProcess

2、挂起进程

可疑进程试图挂起目标进程（通常是挂起目标进程中的所有线程），使目标进程成为僵尸，也就是说，该进程虽然存在，但始终处于非活动状态。

主要调用的 API 函数：kernel32.dll: OpenProcess、SuspendThread、

主要调用的系统服务：ntoskrnl.exe: NtOpenProcess、NtSuspendThread

3、进程修改

可疑进程试图修改目标进程或者插入代码到目标进程中，它这样做的目的是改变目标进程的行为或者在目标进程中隐藏自己的代码。例如，可疑进程通过修改防火墙的进程代码，使得所有网络流量都允许进出。

例如，下面是对某个防火墙反汇编后的部分代码：

004016C8	74 21	JE SHORT 004016EB	Jumps to "blocked"
004016CA	48		
004016CB	74 17	JE SHORT 004016E4	Jumps to "permitted"
004016CD	51		
004016CE	4A		
004016CF	68 E0614300		
004016D0	52		
004016D1	56		
004016D2	E8 ACB20100		
004016D3	83C4 10		
004016D4	33C0		
004016D5	5F		
004016D6	5E		
004016D7	5B		
004016D8	C3		
004016D9	68 D4614300	PUSH 004361D4	ASCII "permitted"; Case 4 of switch 004016C5
004016DA	EB 05	JMP SHORT 004016F0	
004016DB	68 CC614300	PUSH 004361CC	ASCII "Blocked"; Case 3 of switch 004016C5
004016DC	4A		
004016DD	68 C8614300		
004016DE	52		
004016DF	56		
004016E0	E8 8AB20100		

例如，发生一个连接 TCP 端口 25 的事件，但规则库中有这样一条规则“阻止所有到 TCP 25 的连接”，那么，该连接肯定会被防火墙阻止，但是可疑进程把最上面一条指令改为：

```
00401603 | EB 1A | JMP SHORT 004016E4 | Always jump to "permitted"
```

仅仅两个字节，就使防火墙形同虚设。

主要调用的 API 函数：kernel32.dll: OpenProcess、WriteProcessMemory、

主要调用的系统服务：ntoskernel.exe: NtOpenProcess、NtWriteVirtualMemory

4、内核 rootkit 木马安装

内核 rootkit 是一种特殊类型的木马，它们都带有一个内核驱动程序，该驱动程序通过修补系统内核实现了隐藏和自我保护。所以，常规的防火墙软件、防病毒软件、端口监控软件、进程监控软件等防护工具难以发现它们。

主要调用的 API 函数：kernel32.dll: RegCreateKeyEx、RegSetValueEx

Ntdll.dll ZwOpenSection、ZwSetSystemInformation

主要调用的系统服务：ntoskernel.exe: NtRegCreateKey、NtRegSetValue

NtOpenSection、NtSetSystemInformation

5、防火墙旁路攻击

可疑进程通过远程线程插入技术和全局挂钩技术，将恶意代码插入到防火墙的可信进程中(例如 Internet 浏览器 IExplorer.exe)，从而绕过防火墙的检测。

主要调用的 API 函数：kernel32.dll: CreateRemoteThread

user32.dll: SetWindowsHookEx

主要调用的系统服务：ntoskernel.exe: NtOpenSection、NtCreateProcess

NtCreateThread

6、直接访问物理内存

以管理员权限运行的程序能访问计算机上的物理内存，Windows 系统使用虚拟内存技术将应用程序隔离，如果一个程序能查看或修改真实物理内存，它就有可能改变系统内存中的任何东西，这是一个很大的安全漏洞，使得系统上任何单一的保护机制都易受攻击。

主要调用的 API 函数：Ntdll.dll ZwOpenSection

主要调用的系统服务：ntoskernel.exe: NtOpenSection

7、口令偷窃器和键盘记录器

通过采用全局挂钩技术，将某个恶意的动态连接库挂钩到所有进程的进程空间中，该动态连接库的功能是：口令偷窃器是一旦用户在对话框中键入密码，就自动进行俘获；键盘记录器是实时记录用户的击键情况，例如，键入的用户名、口令、网址等。口令偷窃器和键盘记录器实际上一种木马，专门窃取用户的敏感

信息。

主要调用的 API 函数: user32.dll: SetWindowsHookEx

主要调用的系统服务: win32k.sys: 具体的系统服务有待研究

8. windows 文件保护攻击

在正常情况下, 系统进程 Winlogon 发现了有对系统文件进行操作, 便调用 sfc.dll 中的输出函数进行检查。在 sfc.dll 中导出了一个没有名字的未公开的序号为 2 的函数, 调用该函数可以终止 Windows 的文件保护, 可疑进程通过在 Winlogon 进程中建立远程线程, 该线程的起始地址就是 sfc.dll 中序号为 2 的输出函数的地址, 这样该线程执行后, Windows 的文件保护被恶意终止, 可疑进程就可以对系统文件进行修改。

主要调用的 API 函数: kernel32.dll: CreateRemoteThread

主要调用的系统服务: ntoskrnl.exe: NtCreateThread

攻击行为多种多样, 攻击的手段和方法也是层出不穷, 但是从技术的角度来看, 可疑进程最终都需要调用某个或者某几个系统服务来攻击目标进程。因此, 在内核态中, 通过控制系统服务的调用就可以阻止进程的攻击行为。

4.4 进程行为的拦截技术

进程行为拦截采用的是挂钩系统服务地址表的内核修补技术, 通过对系统内核进行修补, 加入自己的特殊代码, 从而达到拦截进程行为的目的。

4.4.1 获取系统服务号

用挂钩系统服务地址表来修补系统内核, 该方法原理清晰, 而且实现起来简洁、高效, 关键在于系统服务号的获取。假设想通过用挂钩系统服务地址表的方法, 将函数 NewService 替换系统服务 RealService, 首先必须要知道 RealService 所对应的系统服务号。在 ntoskrnl.exe 中函数名的格式有两种形式: Zwxxxx 与 Ntxxxx, Ntxxxx 是该系统服务的真正实现, 而 Zwxxxx 形式的入口包含了一份从 ntdll.dll 中的拷贝^[25]。Zwxxxx 在 ntoskrnl.exe 和 ntdll.dll 中的通用代码为: (以 ZwWriteFile 为例)

ZwWriteFile:

Mov eax, 0Eh; 0Eh 为系统服务 NtWriteFile 的系统服务号

Lea edx,[esp+04]; [esp+4]指向调用程序传递给系统服务的参数列表

Int 2Eh; 执行系统服务调度程序

Ret 2Ch; 从堆栈中弹出参数, 返回调用者。

从上面的 Zwxxxx 系统服务通用代码可以看出，只要知道 Zwxxxx 的函数地址，就可以取得系统服务号 ServiceId，它们的关系如下：

```
ServiceId=*(UNLONG*)((UCHAR*)Zwxxxx+1);
```

但是，ntoskrnl.exe 并没有导出所有的 Zwxxxx，例如，ZwCreateThread、ZwOpenProcess 等没有导出，虽然可以通过对 ntoskrnl.exe 反汇编得到相应的系统服务号，但是，系统服务号不是固定不变的，它作为内核编译的一部分自动产生的。可以通过分析 ntdll.dll 的 PE 格式（Portable Executable File Format）（可移植可执行文件格式）来获取这些特殊系统服务的系统服务号^[26]。

4.4.2 挂钩系统服务地址表

进程的攻击行为不仅有很多，而且可以采用多种方法来执行同一种攻击行为，因此，在内核态中，我们需要挂钩多个系统服务来达到拦截进程行为的目的。那么，如何挂钩系统服务呢？其方法如图 4-11 所示：

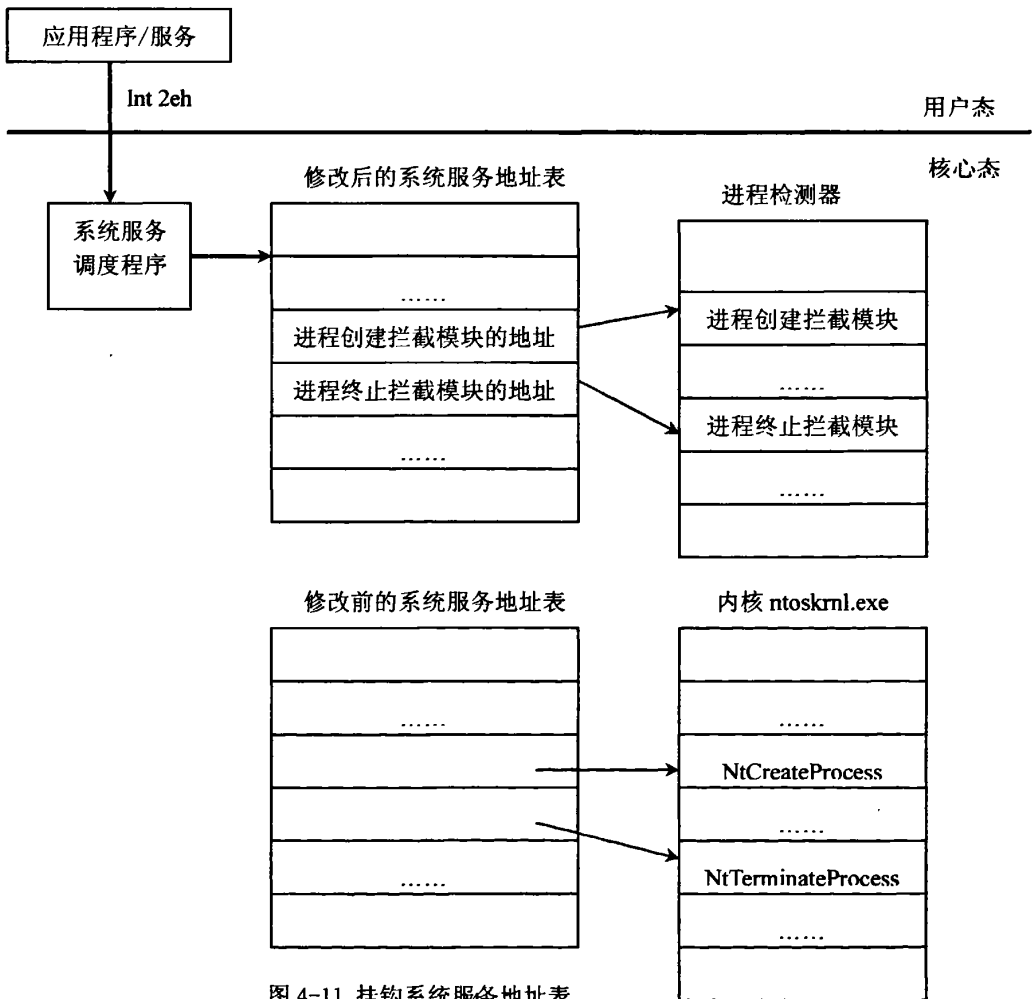


图 4-11 挂钩系统服务地址表

我们以控制进程创建为例，说明挂钩的方法（假设控制进程创建需要挂钩系统服务 NtCreateProcess）。

- 1、获取系统服务 NtCreateProcess 的系统服务号。
- 2、通过 NtCreateProcess 的系统服务号，在系统服务地址表找到 NtCreateProcess 的系统服务地址，保存该地址。
- 3、用进程检测器中的“进程创建拦截模块”的地址替换系统服务地址表中 NtCreateProcess 的系统服务地址。

示例主要代码如下：

KeServiceDescriptorTable 对应于一个数据结构，定义如下：

```
typedef struct SystemServiceDescriptorTable
{
    UINT *ServiceTableBase;
    UINT *ServiceCounterTableBase;
    UINT NumberOfService;
    UCHAR *ParameterTableBase;
}SystemServiceDescriptorTable,*PSystemServiceDescriptorTable;
修改系统服务调用，保存原始的入口地址，修改为我们自定义的程序入口地址，
如 ZwQuerySystemInformation:
OldZwQuerySystemInformation =
(ZWQUERYSYSTEMINFORMATION)(SYSCALL(ZwQuerySystemInformation));
_asm cli
(ZWQUERYSYSTEMINFORMATION)(SYSCALL(ZwQuerySystemInformation))
= NewZwQuerySystemInformation;
_asm sti
```

4.5 本章总结

本章首先介绍了 Windows 内核方面的有关概念，然后对目前常用的几种内核修补技术进行了归纳总结。第三节简要分析了各种进程攻击行为，通过分析，可以知道只要控制系统服务的调用就可以阻止进程的攻击行为，最后一节介绍了通过挂钩系统服务地址表来拦截进程行为的方法。

第五章 进程监测器和决策中心的技术实现

进程监测器负责进程操作的实时监控，它被设计成一个内核驱动程序，通过挂钩系统服务地址表（KiServiceTable）在内核中加入了自己的特殊代码，从而达到拦截进程行为的目的。

5.1 进程监测器的总体结构

进程检测器的总体结构如图 5-1 所示：

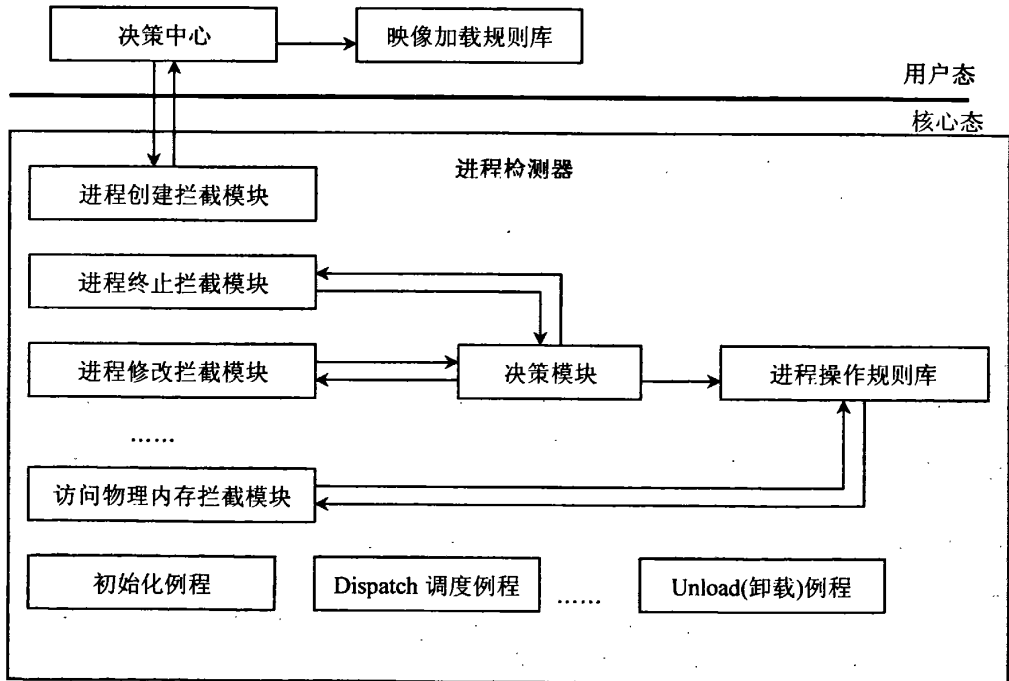


图 5-1 进程检测器的总体结构

进程检测器是一个内核模式的驱动程序，因此，它包含驱动程序所需要的各种例程，如初始化例程、Dispatch 调度例程等等。初始化例程负责创建驱动程序操作的每个设备的设备对象，需要的话还可以在设备名称和设备对用户态可见名称间创建符号链接。同时它还把驱动程序各例程入口点填入驱动程序对象相应的域中^[27]。Dispatch 调度例程是设备驱动程序提供的主要函数，可完成打开，关闭，

读取, 写入以及设备, 文件系统或网络支持的任何其他功能。当被调用去执行一个 I/O 操作时, I/O 管理器产生一个 IRP (I/O 请求包), 并且通过某个驱动程序的调度例程调用驱动程序^[20]。

除了驱动程序的各种例程外, 进程检测器主要由各种拦截模块、决策模块和进程操作规则库 (Process Operate Rule Database, 以下简称 PORD) 所组成。PORD 是控制一个进程对系统资源的操作或者对另一个进程的操作是否允许的唯一依据。设计 PORD 的目的是为了提高决策的速度, 因为系统中的进程不会太多, 所以在系统内核中保存这样一个规则库, 不仅不会占用很多的内存空间, 而且可以大大减少决策中心和进程检测器之间的通信量, 以及因协调两者之间的同步所花费的时间, PORD 的控制规则项的格式用四元组表示为 (O, ID, P1, P2), 其中 O、ID、P1、P2 的含义分别如下:

O —— 可执行程序的路径及名字。

ID—— 进程标识符。

P1——IGBOB 对该进程实施的保护措施 (例如, 不能对该进程进行终止、修改等操作)。

P2——IGBOB 赋予该进程的权限 (例如, 该进程可以对其它进程进行终止操作)。

PORD 实际上是进程加载规则库的一个子集, PORD 中规则的添加和删除均有进程检测器来完成。当进程检测器添加 O 的规则项时, 要求决策中心根据 O 查询映像加载规则库, 将 O 的 P1、P2 下传给进程检测器, 这样进程检测器就将 (O, ID, P1, P2) 添加到 PORD 中。

5.2 控制进程的创建

在介绍如何控制进程的创建以前, 有必要先了解一下进程和线程创建的主要阶段。

5.2.1 进程创建的主要阶段

当应用程序调用 Win32 进程创建的函数之一, 诸如 CreateProcess、CreateProcessAsUser 或 CreateProcessWithLogon 函数时, 将创建一个 Win32 进程。创建进程是在操作系统的三个部分的共同参与下分成六个阶段完成的。这三部分分别是: Win32 客户端的 Kernel32.dll 库、Windows 执行程序 and Win32 子进程 (Csrss)。下面是使用 Win32 CreateProcess 创建进程的主要阶段的总结^[20]。

1) 打开将在进程中被执行的映像文件 (.EXE)

CreateProcess 的第一步是找到合适的 Win32 映像, 这个 Win32 映像将

运行由调用程序指定的可执行程序，并调用内部系统函数 ZwCreateSection 创建一个区域对象以便稍后把它映射到新进程的地址空间中。如果没有指定映像名称，那么命令行的开始标记（被定义为命令行字符串以空格或制表符结尾的开始部分，是一个有效的文件规格说明）被用作映像文件名称。

2) 创建 Windows 执行程序进程对象。

这一步通过调用内部系统函数 NtCreateProcess 来创建运行映像的 Windows 执行程序进程对象。创建执行程序进程对象包括：设置 EPROCESS 块、创建初始进程地址空间、创建内核进程块、决定进程地址空间的设置、设置 PEB、完成执行程序进程对象的设置。

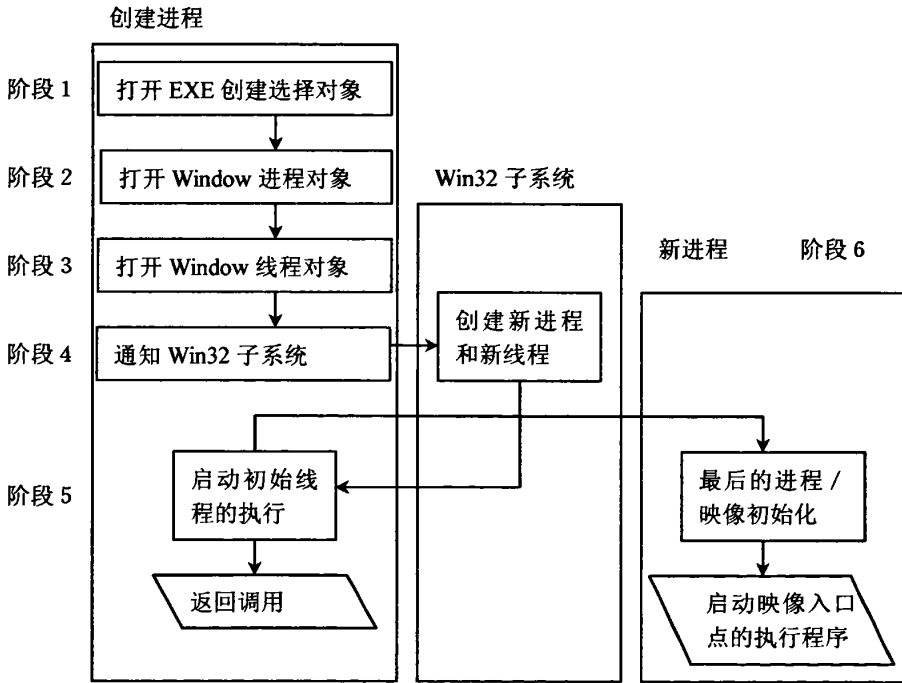


图 5-2 进程创建的主要阶段

3) 创建初始线程(堆栈、环境和 Windows 执行程序线程对象)。

通过前面两步，完成了 Windows 执行程序进程对象的设置。然而，它现在还没有线程，因此还不能做任何事情。在可以创建线程前，必须首先设置堆栈和环境，然后调用 NtCreateThread 创建线程。这个线程是在挂起状态

下创建的，它要等到进程被完全初始化后才能继续运行。

- 4) 向 Win32 子系统通知新进程，以便它可以设置新的进程和线程。

在所有必要的执行程序进程和线程对象创建以后，Kernel32.dll 将消息发送到 Win32 子系统，这样 Win32 子系统就能对新的进程和线程进行设置。

- 5) 启动初始线程的执行（除非 CREATE-SUSPENDED 标记被指定）。

到目前为止，已经确定了进程环境，并且分配了它的线程所用的资源，进程有了一个线程，并且 Win32 子系统知道这个新进程。除非调用程序指定(CREATE_SUSPENDED)标志，初始线程现在被恢复执行，这样它就可以开始运行并完成在新进程环境中的进程初始化工作的剩余部分。

- 6) 在新进程和线程的环境中，完成地址空间的初始化（例如加载所需的 DLL）并开始程序的执行。

进程创建的六个阶段如图 5-2 所示：

5.2.2 挂钩 NtCreateThread 控制进程的创建

在创建进程的六个阶段中，所调的系统服务有：

阶段 1：NtCreateFile、NtCreateSection

阶段 2：NtCreateProcess

阶段 3：NtCreateThread

阶段 4：NtRequestWaitReplyPort

阶段 5：NtResumeThread

阶段 6：

上述列出的系统服务都是创建进程过程中需要调用的，因此从理论上讲，无论挂钩哪一个系统服务都可以达到控制进程创建的目的。但是，NtCreateFile、NtRequestWaitReplyPort、NtResumeThread 无法区分该服务是创建进程过程调用的还是因为执行其它操作而调用的，而挂钩 NtCreateProcess 又无法获得被创建进程的映像信息，例如映像文件名、命令行参数等，因此，如果想控制进程的创建，一般挂钩系统服务 NtCreateSection 和 NtCreateThread。下面的表 5.1 比较这两种方法的优劣。

当一个从未运行的或者上次运行后已修改过的的映像文件需要运行时，IGBOB 完全依赖用户所作出的抉择：是允许还是拒绝该映像文件的运行。因此，IGBOB 需要提供尽可能多的信息给用户，为用户的抉择提供充分的依据，因此，IGBOB 将通过挂钩系统服务 NtCreateThread 来控制进程的创建。

表 5.1 挂钩系统服务 NtCreateSection 和 NtCreateThread 的比较

系统服务名	对系统的影响	获取系统服务号	获取映像参数	新的系统服务
NtCreateSection	大	简单	映像文件名、父进程的映像信息	处理简单
NtCreateThread	小	复杂	被创建进程和父进程的映像信息	处理复杂

5.3 控制进程的终止

有一些病毒为了运行时不被发觉,会主动关闭实时监控软件和防火墙。例如,木马“广外女生”在服务端被执行后,会自动检查进程中是否含有“金山毒霸”、“防火墙”、“iparmor”、“tcmmonitor”、“实时监控”、“lockdown”、“kill”、“天网”等字样,如果发现就将该进程终止,也就是说使实时监控软件和防火墙不起作用^[6]。一个进程(攻击进程)要想终止另一个进程(目标进程)的运行,方法有很多,总结如下:(下面提及的函数均由 Kernel32.dll 导出)

1、攻击进程通过调用 TerminateProcess 来终止目标进程。

2、攻击进程通过创建远程线程来终止目标进程。

攻击进程创建一个远程线程,远程线程的起始地址指向 ExitProcess。一旦该线程得到运行,目标进程被自动终止。

3、攻击进程通过调用 SuspendThread 来终止目标进程。

攻击进程首先枚举目标进程中的所有线程,然后调用函数 SuspendThread 挂起目标进程中的所有线程,从而目标进程成为僵尸。

4、攻击进程通过调用 TerminateThread 来终止目标进程。

攻击进程首先枚举目标进程中的所有线程,然后调用函数 TerminateThread 终止目标进程中的所有线程。

5、攻击进程通过调用 DebugActiveProcess 来终止目标进程。

攻击进程通过调用 DebugActiveProcess 作为一个调试器附着到目标进程上,这样攻击进程就被允许终止目标进程了^[28]。

终止进程的方法虽然很多,但最终都必须调用相应的系统服务来实现,因此,只要控制住与进程终止有关的系统服务,就可以控制进程终止,其方法如下:

IGBOB 通过设置保护权限来保护目标进程。例如,如果目标进程被设置成防进程终止,那么即使攻击进程拥有终止其它进程的权限,也无法终止目标进程。这样保护了 Windows 系统进程和用户可信进程免受其它进程,服务,以及系统上的其它形式的可执行代码的攻击。

如果攻击进程想终止某个目标进程的运行,它首先必须打开目标进程。因此,进程检测器通过挂钩系统服务 NtOpenProcess 来达到防止其它进程对目标进程的攻击。

控制一个进程对另一个进程的的其它操作,例如、对某个进程进行写、挂起等操作,其控制原理和控制进程的终止的原理基本相同,这里不在叙述。

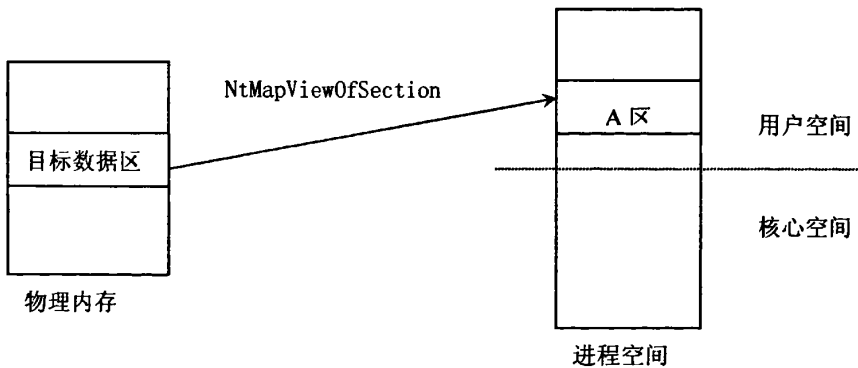
控制对物理内存的直接访问

以管理员权限运行的程序能访问计算机上的物理内存,Windows 系统使用虚拟内存技术将应用程序隔离,如果一个程序能查看或修改真实物理内存,它就有可能改变系统内存中的任何东西,这是一个很大的安全漏洞,使得系统上任何单一的保护机制都易受攻击。如果我们通过限制程序访问内存,可以防止所有的物理内存攻击,如果一些程序确实需要访问物理内存(少数安全程序和游戏有此需要),可以使用控制规则进行指定,这样即可以防住此安全危险,又能使程序正常运行。

5.3.1 访问物理内存的方法

一个工作在在用户态下以管理员权限运行的程序可以通过直接向 \device\physicalmemory 中写入数据, \Device\PhysicalMemory 这个对象用于对系统物理内存的访问^[29]。这种方法的基本思想如下如图 5-3 所示:

首先用 ZwOpenSection()函数获得内核对象\device\physicalmemory 的句柄,然后用 ZwMapViewOfSection()函数将物理内存中的目标数据区映射到用户地址空间中,称为 A 区。这样,用户模式的应用程序访问 A 区就象访问物理内存中的目标数据区一样。



5-3 访问物理内存

5.3.2 控制对物理内存的访问

从访问物理内存的方法可以看出，它首先必须要调用系统服务 `NtOpenSection` 来打开内核对象 `\device\physicalmemory`，因此，进程检测器通过挂钩系统服务 `NtOpenSection` 来达到防止进程对物理内存的直接访问。

控制访问物理内存的过程为：首先判断对象的名字是否为“`\device\physicalmemory`”，如果是，则调用核心态中的决策模块，决策模块判断当前进程有无访问物理内存的权利，如果有，则调用原来的系统服务 `NtOpenSection`。

5.4 控制安装驱动程序

有的时候，阻止安装驱动程序是很有必要的。例如阻止 `rootkit` 的安装，`rootkit` 可以通过安装驱动程序达到了隐身的目的，使你根本察觉不到它的存在，连防病毒软件也对它无能为力。因此通过阻止安装驱动程序来防止 `rootkit` 的入侵是一种比较好的选择。

要想控制安装驱动程序，必须要知道有哪些方法可以安装驱动程序，下面分别介绍这些方法以及针对这些方法如何进行控制。

一、修改注册表

注册表键值 `HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services` 是服务控制管理器（`Service Control Manager`，以下简称 `SCM`）的数据库，在该键值下定义了每个驱动/服务的描述信息，这些信息包括驱动/服务的映像文件位置、可选的显示名称、启动类型、描述当系统启动时如果发生错误如何反应的错误代码等等。如果在该注册表键下面写入某个驱动/服务的注册信息，并且注册为自动启动的话，那么，重新启动机器后，该驱动/服务将会自动启动。

针对这种方法，我们可以通过挂钩注册表系统服务 `NtSetValueKey`、`NtCreateKey` 来阻止修改 `SCM` 的数据库，从而控制驱动程序的安装。

二、使用 `SCM` 函数装入

该方法是加载驱动/服务最常用的方法。Windows 提供了一个 `Win32` 接口，可以在系统运行期间装入、卸载驱动/服务。这是由 `SCM` 实现的。

在装入或访问任何服务之前，必须先获得一个 `SCM` 的句柄，这需要调用 `OpenSCManager()`。这个句柄叫管理器句柄，它在所有的 `CreateService()` 和 `OpenService()` 函数中都要用到。同样，这两个函数也会返回句柄，叫做服务句柄。这些句柄在针对具体的某个服务的调用时要用到，如 `ControlService()`、`DeleteService()`、`StartService()`。这两种句柄都可以用 `CloseServiceHandle()` 函数关

闭。

装入一个驱动程序需要如下几步：

- 1、调用 `OpenSCManager()`，获得一个管理器句柄。
- 2、调用 `CreateService()`，将新的驱动加入到系统中，并得到服务句柄。
- 3、调用 `StartService()`，利用服务句柄，启动服务，使其开始运行。
- 4、调用 `CloseServiceHandle()`，释放管理器句柄和服务句柄。

针对这种方法，我们也可以通过挂钩注册表系统服务 `NtSetValueKey`、`NtCreateKey` 来阻止修改 SCM 的数据库，从而控制驱动程序的安装。

三、使用直接写物理内存的方法

通过直接向 `\device\physicalmemory` 中写入数据的方法来载入驱动程序。直接写物理内存的方法可以参见 5.4，这里不再阐述。

针对这种方法，我们可以通过挂钩系统服务 `NtOpenSection` 来阻止直接写物理内存，从而控制驱动程序的安装。

四、使用 `SystemLoadAndCallImage` 系统调用

当使用 `SystemLoadAndCallImage` 来载入驱动程序时，既不需要修改注册表键值，也不需使用 SCM，取而代之的是使用一个 NT 系统调用 `ZwSetSystemInformation`^[30]。

示例代码如下：

```
#define SystemLoadAndCallImage 38
typedef struct _SYSTEM_LOAD_AND_CALL_IMAGE {
    UNICODE_STRING ModuleName;
} SYSTEM_LOAD_AND_CALL_IMAGE;
SYSTEM_LOAD_AND_CALL_IMAGE LoadImage;

//要加载模块的 native NT 格式的完整路径
WCHAR daPath[] = "\\??\\C:\\load.sys";
RtlInitUnicodeString( &(LoadImage.ModuleName), daPath );
//将 LoadImage 指定的模块加载进内核空间
ZwSetSystemInformation(           //设置影响操作系统的信息
    SystemLoadAndCallImage,       //将被设置的系统信息的类型: 加载模块
    &LoadImage,                   //将要被加载的模块
    sizeof(SYSTEM_LOAD_AND_CALL_IMAGE)
);
```

针对这种方法，我们可以通过挂钩系统服务 `NtSetSystemInformation` 来控制

驱动程序的安装。

5.5 入侵防护体系的自身安全

IGBOB 所要实现的目标是：

Windows 系统进程和用户可信进程免受其它进程，服务，驱动程序，以及系统上的其它形式的可执行代码的攻击。

只有被用户认可的程序才被允许执行。

从 IGBOB 的实现目标可以看出，它担当着安全防护的重任，是系统进程和用户可信进程的安全警卫。但 IGBOB 本身是一个应用程序，它也存在自身的安全问题；同时，由于它的特殊性，必然受到可疑程序的特别的关照，成为可疑程序必须除掉的目标之一。所以，IGBOB 自身的安全问题非常重要。

一、可疑程序对 IGBOB 发动的攻击

1、修改控制规则库

可疑程序试图通过修改 IGBOB 的控制规则库，非法获取终止进程和写进程的权限，从而对系统进程和用户可信进程实施攻击。

修改决策中心进程空间中的代码。

可疑程序试图对策略中心 (Decision.exe) 所对应的进程进行写操作，对决策部分的代码进行非法修改，改变执行路径，从而使 IGBOB 的安全防护形同虚设。

终止决策中心所对应进程。

可疑程序试图终止策略中心 (Decision.exe) 所对应的进程，从而使 IGBOB 停止工作。

删除 IGBOB 的相关文件。

可疑程序试图删除 IGBOB 的相关文件，例如 IGBOB 的参数文件、控制规则库、进程检测器对应的驱动程序、控制中心对应的映像文件等，从而使得系统下次启动后 IGBOB 无法正常工作。

删除 IGBOB 在注册表中的相关键值。

IGBOB 的进程检测器是一个内核驱动程序，决策中心是一个服务，它们之所以在系统启动后就处于运行状态，是因为它们在注册表中分别被注册成驱动和服务，并且运行模式被设成自动。如果可疑程序删除 IGBOB 在注册表中的相关键值，系统下次启动后 IGBOB 的进程检测器、决策中心均无法启动。

二、IGBOB 对自身安全的防护

进程检测器在拦截进程操作的同时，肩负着防护自身安全的责任。根据可疑程序对 IGBOB 可能发动的攻击，IGBOB 将采取下列措施来保护自身的安全。

映像加载规则库中，决策中心（假设为 Decision.exe）的保护权限被设置为防止其它进程对它进行修改和终止等操作，也不允许 IGBOB 的控制中心修改它的保护权限。这样可疑程序就无法对 Decision.exe 发起有效攻击。

IGBOB 建立一个保护名单，在该名单中列出了需要保护的文件和注册表键值。进程检测器通过挂钩系统服务：NtDeleteKey、NtDeleteValueKey、NtSetValueKey、NtWriteFile、NtDeleteFile 等来防止可疑程序删除 IGBOB 的相关文件和键值。

通过采取上面两项措施，可以保证 IGBOB 自身的安全，从而使可疑程序无机可乘。

5.6 决策中心的技术实现

决策中心在 IGBOB 中扮演着重要的角色，起着承上起下的作用。决策中心是一个服务，它根据决策的前提条件来查询映像加载规则库，通过规则的匹配情况或用户的抉择来确定进程能否创建，并将决策结果下传给进程监测器。同时，它还负责把允许或拒绝进程操作的情况记录到日志库，以供用户日后查询和分析。

5.6.1 映像加载规则库的设计

映像加载规则库(Image Load Rule Database, 以下简称 ILRD)工作在用户层，它是控制程序能否加载运行的重要依据。其控制规则项的格式用五元组表示为 (O, P1, P2, A, H)。其中 O、P1、P2、A、H 的含义分别如下：

O —— 可执行程序的路径及名字。

P1—— IGBOB 对由 O 加载的进程实施的保护措施(例如，不能对该进程进行终止、修改等操作)。

P2——IGBOB 给由 O 加载的进程赋予的权限 (例如，该进程可以对其它进程进行终止、修改等操作)。

A ——IGBOB 将允许还是拒绝 O 的运行，它可分为下列二种运行类别

ALLOW ——如果 O 没有修改的话，永远允许。

BLOCK ——如果 O 没有修改的话，永远阻止 O 的运行。

H ——对 O 采用 MD5 算法计算出的文件校验和。

ILRD 中的规则项除了决策中心可以进行自动添加外，也允许用户通过 IGBOB 的控制中心进行添加、删除、修改。

5.6.2 控制进程创建的决策过程

决策中心在初始化时创建了一个决策线程,该线程对是否允许进程的创建作出决策,进程检测器在拦截到进程创建操作后,取得决策因子(即映像文件名和路径),然后唤醒决策中心的决策线程,决策线程根据决策的前提条件来查询ILRD,通过规则的匹配情况或用户的选择来确定进程能否创建,并将决策结果下传给进程监测器。下面简要介绍决策过程。

决策过程如图 6-1 所示:

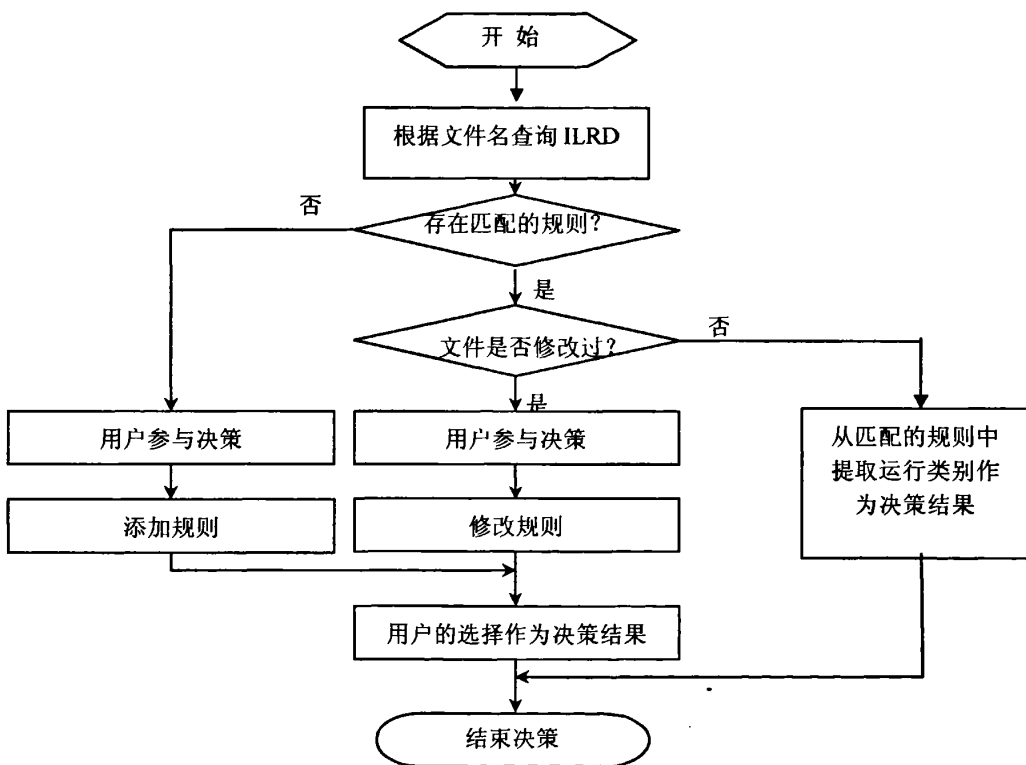


图 5-4 控制进程创建的决策过程

例如,决策线程需要对映像文件 `c:\explorer.exe` 是否允许运行作出决策。其决策过程如下:决策中心根据映像文件名 `c:\explorer.exe` 来查询映像加载规则库,查询结果为:

ILRD 中找不到与 `c:\explorer.exe` 相匹配的规则。

说明

进程检测器第一次检测到 `c:\explorer.exe` 想要运行
控制中心将它的规则从 IRDB 中删除了。

上次运行过，但没有添加到 ILRD 中。

这种情况需要用户参与决策。

ILRD 中存在与之匹配的规则 (c:\explorer.exe, P1, P2, A, H)。

重新计算 c:\explorer.exe 的校验码 CheckSum，计算结果为：

(1) CheckSum 不等于 H，那么说明 c:\explorer.exe 被添加到 ILRD 中后，c:\explorer.exe 被修改过，这种情况也需要用户参与决策。

CheckSum 等于 H

决策结果就是 A，决策过程到此结束。

如果需要用户参与决策，决策线程询问用户是允许还是拒绝 c:\explorer.exe 的运行，用户有四种选择：ALLOW（永远允许）、ALLOW ONE（允许运行一次）、BLOCK（永远阻止）、BLOCK ONE（阻止运行一次）。如果用户选择的是 ALLOW 或 BLOCK，那么决策线程还要在 ILRD 中添加规则或者修改规则。

添加规则

如果 ILRD 中找不到与 c:\explorer.exe 相匹配的规则，那么决策线程计算 c:\explorer.exe 的文件校验码 CheckSum，然后在 IRDB 中添加这样一条规则：

(c:\explorer.exe, 缺省保护权限, 缺省访问权限, 询问的结果, CheckSum)

修改规则

如果 IRDB 中找到与 c:\explorer.exe 相匹配的规则，那么决策线程将规则 (c:\explorer.exe, P1, P2, A, H) 修改为 (c:\explorer.exe, P1, P2, 询问的结果, CheckSum)。

5.7 本章小结

本章首先介绍了进程检测器的总体结构和进程操作规则库的设计，其次研究了控制进程创建、控制进程终止、控制访问物理内存、控制安装驱动等的方法和实现技术，对如何防护 IGBOB 自身安全提出了自己的想法；设计了一个映像加载规则库的结构，然后详细描述了决策中心对进程创建进行决策的过程。

第六章 结束语

1. 依据 IGBOB 模型所建立的入侵防护系统所起的作用:

- 从系统的最底层开始控制, 保护重要的安全程序不被修改;
- 防止新的可疑程序执行, 能够有效阻止病毒发作和木马运行。

2. 入侵防护系统的缺陷:

- 以 Window NT 内核为基础进行的设计, 因此针对性的防护也只能是 Windows NT, Windows 2000, Windows XP 和 Windows 2003, 对操作系统 Windows 98/ME 无法实施保护;
- 必须安装在干净的机器上, 也就是说, 在安装入侵防护系统之前, 必须确保机器上没有病毒、木马和其它可疑程序。
- 虽能防住未知病毒和木马, 但对用户的要求比较高, 当一个新的普通程序、病毒、木马需要加载运行时, 入侵防护系统将询问是否允许该程序运行, 这时候, 将全靠用户自己判断, 因此, 此系统应用于主机的防护更为理想。

总之, 单一的安全保护往往效果不会理想, 基于行为的入侵防护体系也是如此。只有对系统采用安全漏洞评估、防病毒软件、入侵防护系统、个人防火墙等多层安全防护措施才能对系统进行全方位的保护。

参考文献

- [1] 吴海旺 基于网络的入侵检测系统的研究与设计: [硕士学位论文], 北方交通大学 2004-2.
- [2] shotgun. 揭开木马的神秘面纱(二),
<http://hi.baidu.com/calmly520/blog/item/d1d56ddac05463dab6fd48d5.html>,
2007-4.
- [3] 谭曙光, 中科网威让入侵检测与防火墙实现互动, 中国计算机报, 2002-4
- [4] THE MOST POWERFUL SECURITY PROGRAM FOR WINDOWS,
<http://www.diamondcs.com.au/processguard/>, 2004-12.
- [5] 中华人民共和国计算机信息系统安全保护条例,
<http://www.china.org.cn/chinese/zhuanti/198455.htm>, 1994-2-18.
- [6] 张友生, 米安然, 计算机病毒与木马程序剖析, 北京: 科海电子出版社,
2003-3. 25~262
- [7] 天缘, 安全宝典——病毒及攻击防御手册,
<http://www.yesky.com/SoftChannel/72356682675519488/20040729/1836659.shtml>,2004-7.
- [8] Microsoft Security Bulletin (MS00-052).
<http://www.microsoft.com/technet/security/bulletin/MS00-052.asp>,
2000-7-28.
- [9] jasonye163, Win32.Troj.ADNavihelper,
<http://baike.baidu.com/view/446400.html> , 2006-3.
- [10] TOo2y, 一个修改 NT 内核的真实 RootKit,
<http://www.xfocus.net/articles/200306/558.html>, 2003-6.
- [11] 必需掌握的病毒知识,
<http://www.sz.ah163.net/Article/jsjs/200501/1073.html>, 2005-1.
- [12] 特洛伊木马藏身处大揭密,
<http://ntsafes.heut.edu.cn/list.asp?classid=47&articleid=675>, 2004-4.
- [13] MSDN Knowledge base Q197551,
<http://support.microsoft.com/default.aspx?scid=kb;en-us;197571>, 2005-1.
- [14] 刘宝旭, 入侵检测与防火墙互动,
<http://www.ciw.com.cn/media/ciw/1058/b4301.html>, 2001-9.
- [15] 吴海燕, 威丽, 校园数据中心网络安全防范体系研究,

- 实验技术与管理 (双月刊), 第 21 卷, 第 3 期, 总第 96 期), 2004-6.
- [16] 张红旗, 信息安全, 北京: 清华大学出版社, 2002. 11. 210~212
- [17] NJUE, 反病毒引擎设计之实时监控篇,
http://network.ccidnet.com/pub/article/c1101_a81772_p1.html, 2004-1.
- [18] 段刚, 加密与解密, 北京: 电子工业出版社, 2003-6. 15~25
- [19] 尤晋元, 史美林等, Windows 操作系统原理, 北京: 机械工业出版社, 2001-08. 71~81
- [20] 詹剑锋, 张文耀, 黄艳. Windows 2000 内部揭密, 北京: 机械工业出版社, 2001-10. 30~336
- [21] David A Solomon, Mark Russinovich. Inside Microsoft Windows 2000. Microsoft Press, 2001-4. 40~69
- [22] 王建华, 张焕生, 侯丽坤等, Windows 核心编程, 北京: 机械工业出版社, 2000-5. 45~55
- [23] qazmju, 钩子函数(HOOK)完整的教程,
http://qazmju.blog.hexun.com/372830_d.html, 2005-6.
- [24] Peter Szor, 计算机病毒防范艺术 (段海新, 杨波等), 北京: 机械工业出版社, 2007-1. 137~140
- [25] Garry Nebbet, Windows NT/2000 native API reference, MacMillan Technical Publishing, 2000-6.
- [26] Matt Pietrek, Peering Inside the PE: A Tour of the Win32 Portable Executable File Format, From MSDN Magazine March 1994 on Internet,
<http://msdn2.microsoft.com/en-US/library/ms809762.aspx>
- [27] Walter Oney, Programming the Windows Driver Model, Microsoft Press International, 2002-12.
- [28] 长河落日, Windows 核心编程,
http://www.vchelp.net/itbookreview/view_paper.asp?paper_id=1194, 2004-3.
- [29] 火翼., 如何在 windows 程序中读取 bios 内容,
<http://dev.rdx.com/VC/VCSYS/2005-7/20/012810279.shtml>, 2005-7
- [30] Sephiroth.V, 用 SystemLoadAndCallImage 加载 Rootkit,
<http://www.xfocus.net/articles/200309/619.html>, 2003-9.

致 谢

本论文的工作是在我的导师张钢教授的悉心指导下完成的,张钢教授严谨的治学态度和科学的工作方法以及豁达与宽容的为人风度是我终生学习的榜样。在此衷心感谢张钢老师对我的关心和指导。

在撰写论文期间,得到了我的同学李国宏的大力帮助,在此也向他表达我的感激之情。此外,还要感谢所有那些在天津大学曾经给予我知识的老师们。

同时,对百忙之中审阅我论文的老师,以及参加答辩会的各位专家表示深深的谢意。

最后,还要感谢我的单位领导和我的家人,他们的关心和支持使我能够在学校专心完成我的学业。

衷心感谢所有关心、支持、帮助过我的人。