



Windows内核安全 驱动开发

谭文 陈铭霖 等著



Windows内核安全 与驱动开发



谭文 陈铭霖 等著

电子工业出版社

Publishing House of Electronics Industry

内 容 简 介

本书的前身是《天书夜读——从汇编语言到 Windows 内核编程》和《寒江独钓——Windows 内核安全编程》。与 Windows 客户端安全软件开发相关的驱动程序开发是本书的主题。书中的程序使用环境从 32 位到 64 位，从 Windows XP 到 Windows 8 都有涉及，大部分程序不经过修改即可在 Windows 10 上运行。同时本书也深入浅出地介绍了进行内核安全编程所需要的操作系统、汇编等基础知识。

本书共分三篇，基础篇囊括了驱动开发的基础知识，降低了入门的难度；开发篇介绍了在实际工作中可能遇到的各种开发需求的技术实现，包括：串口的过滤、键盘的过滤、磁盘的虚拟、磁盘的过滤、文件系统的过滤与监控、文件系统透明加密、文件系统微过滤驱动、网络传输层过滤、Windows 过滤平台、NDIS 协议驱动、NDIS 小端口驱动、NDIS 中间层驱动、IA-32 汇编基础、IA-32 体系中的内存地址、处理器权限级别切换、IA-32 体系结构中的中断和 Windows 内核挂钩；高级篇包含了汇编语言、操作系统原理、处理器体系架构相关的内容。本书是由长期从事这个行业的工程师自己写的，所以处处以实用为准。对细节的考究主要体现在对实际问题的解决，而不是知识的详尽程度上。

本书适合计算机安全软件从业人员、计算机相关专业院校学生以及有一定 C 语言和操作系统基础知识的编程爱好者阅读。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

Windows 内核安全与驱动开发 / 谭文等著. —北京：电子工业出版社，2015.6
ISBN 978-7-121-26215-9

I. ①W… II. ①谭… III. ①Windows 操作系统—程序设计 IV. ①TP316.7

中国版本图书馆 CIP 数据核字 (2015) 第 118552 号

策划编辑：李 冰

责任编辑：葛 娜

印 刷：北京中新伟业印刷有限公司

装 订：三河市皇庄路通装订厂

出版发行：电子工业出版社

北京市海淀区万寿路173信箱

邮编：100036

开 本：787×1092 1/16

印张：42.5 字数：1121千字

版 次：2015年6月第1版

印 次：2015年6月第1次印刷

定 价：139.00元（含光盘1张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010) 88258888。

本书的作者和贡献者

本书的内容是《天书夜读——从汇编语言到 Windows 内核编程》（以下简称《天》）和《寒江独钓——Windows 内核安全编程》（以下简称《寒》）这两本书的原版内容重新编辑整理，并新添了一些章节而成的。除了直接编写本书的作者外，《天》与《寒》两本书也有多位业内技术人士协同编写。所有的作者一并介绍如下：

谭文，网名楚狂人，长期从事客户端安全的开发工作。先后在 NEC、英特尔亚太研发有限公司、腾讯科技任职。曾经从事过企业安全软件、x86 版 Android、腾讯电脑管家等开发工作。编写了《天》与《寒》的大部分章节，并为本书重写了部分章节，添加了一些新的内容。

陈铭霖，腾讯电脑管家团队高级工程师，长期从事 Windows 系统研究，目前从事客户端安全的开发工作。编写了本书中部分新的章节，并统稿全书，重新整理和编辑了《天》和《寒》的全部内容。对本书的面世做出了巨大的贡献。

张佩，Windows 驱动开发技术专家，长期从事声卡、显卡等硬件驱动程序的开发、调试工作。目前在英特尔亚太研发有限公司平板电脑相关的部门工作。曾著有《竹林蹊径——深入浅出 Windows 驱动开发》一书。为《寒》贡献了若干个网络驱动相关的章节，这些内容大多原版整合到了本书中。

杨潇，曾为 Windows 客户端安全工程师，先后在上海贝尔和北京 Comodo 工作。后来离职创业，目前为西安一家医疗科技公司的 CEO。为《寒》一书编写了磁盘驱动相关的章节，这些章节也整合到了本书中。

邵坚磊，网名 wowocock，业内著名的 Windows 安全技术专家。长期从事 Windows 安全相关的内核开发工作。目前在奇虎 360 任职。为《天》和《寒》都贡献了部分章节和代码实例，这些内容有一部分整合到了本书中。

卢冠豪，中国台湾人。毕业于辅仁大学资讯工程学系。长期从事 C、C++、网络与通信程序设计的工作；参与过“端点安全”、“资产管理”、“网络流量分析”等项目的开发与维护；擅长 Windows 项目开发。编写了《寒》一书中的文件系统微端口过滤一章，此章也整合到了本书中。

前 言

吉 炳 贡 研 究 所 编 译 本

本书的前身是《天书夜读——从汇编语言到 Windows 内核编程》和《寒江独钓——Windows 内核安全编程》。实际上，本书相当于这两本书（重写和充实了不少章节）的合体。因为这两本书讲述的内容本来就是同一类技术，只不过各有侧重，合并成一本书之后体系更加完整了。

与 Windows 客户端安全软件开发相关的驱动程序开发是本书的主题。因为这些驱动程序都运行在 Windows 内核中，所以本书称为《Windows 内核安全与驱动开发》。这些驱动程序主要分成三大类：第一类是串口、键盘等输入/输出设备驱动程序；第二类是硬盘、文件系统驱动程序；第三类是网络驱动程序。覆盖了客户端安全的 Windows 编程中所涉及的各种驱动程序。书中的程序使用环境从 32 位到 64 位，从 Windows XP 到 Windows 8 都有涉及，大部分程序不经过修改即可在 Windows 10 上运行。同时本书也深入浅出地介绍了进行内核安全编程所需要的操作系统、汇编等基础知识。

与前身相比，本书重新排布了章节，主体上分为三个部分。第一个部分为“基础篇”，囊括了驱动开发的基础知识，降低了入门的难度；第二个部分为“开发篇”，包含了在实际工作中可能遇到的各种开发需求的技术实现；第三个部分是“高级篇”，汇编语言、操作系统原理、处理器体系架构相关的内容都放在了本篇中介绍。本篇内容对读者来说可能会感觉比较困难，但由于有前两篇的基础，相信读者不会太难以跨越。

本书是由长期从事这个行业的工程师自己写的，所以处处以实用为准。对细节的考究主要体现在对实际问题的解决，而不是知识的详尽程度上。有些读者可能会觉得这不太像一本“教科书”，在某些细节点上很有“囫囵吞枣”的嫌疑。当然，本书本来就不是教科书，但它是可以用来学习的。只是学习的方法，以实际应用为唯一的动力。

以前常有读者和我交流。我发现最常问的问题是：“我对内核编程有兴趣，请问应该如何学习呢？”

学习的方法应该根据每个人不同的需要有所不同。每个人性格不同，学习的路线也会不一样。比如有的人就擅长计划，他们会先制定周全的步骤，然后按部就班地执行。先从有经验的人那里讨教“如何学习”也是步骤之一。我曾很努力地想这样去做，但总是没有足够的毅力而失败。所以我成了一个随波逐流、随遇而安的家伙。我的学习都是迫不得已，没有计划和方法可言。所以被问到这个问题时，我只能说，我都是在解决实际问题的过程中学习的。这个风格也被带到了本书中。

在读大学的时候，我对驱动开发并无任何概念，我也不为此担心。什么时候我会需要为一个硬件去编写驱动程序呢？这不是硬件厂商的事吗？我为一家硬件厂商工作的机会本来就低，更何况几千个员工里轮到我去写驱动的概率更是低到不可思议的程度。但世事就这么难料。我刚进第一家公司，自以为 VC 6.0 用得多么熟练，准备大干一场的时候，老板说：“我这里有个无盘驱动，你去做试试看……”

当时，他们需要的是一种类似“网络邻居”的共享盘。硬盘的主体在服务器上，上面有很多文件共享给所有的客户机。Windows 本身的“网络邻居”不足以满足全部需求。他们要求每个客户机对这个磁盘都有读写的权力，可以新建、修改或者删除磁盘上的文件。如果直接用 Windows 的目录共享，很快这个磁盘上的文件就会被修改得乱七八糟。所以另一个要求是，任何人都可以修改，但任何人的修改都不得影响他人。

这听起来有点怪，其实很简单。大家共享一台服务器的磁盘上的所有文件，但各自对这些文件的修改都保存到本地。虽然听起来是很合理、简洁的一个需求，但是对我来说却完全摸不着头脑。我写过很多程序，但还从未写过一个东西能让计算机的硬盘的行为发生改变。这是因为我一直只会写用户软件，从没动过操作系统。

操作系统非常努力地提供了完善的功能，来应付各种用户的需求。但用户的需求真是取之不尽、用之不竭的，仅仅一家开发操作系统的公司，永远也无法满足用户的需求。是否允许第三方拥有操作系统的权限来满足这些需求呢？这就形成了两种截然不同的应对方案：Windows 是允许的，任何第三方都可以通过开发驱动程序的形式来改变系统内核的行为；而像苹果的 iOS 这种操作系统是完全禁止的。

为了证明我的价值足够保底薪水，我不得不开始研究这个东西怎么实现。它肯定不是我以前实现过的任何一类工程。当时还是 Windows 9x 的时代，资料奇缺，甚至连开发用的工具包都不知道去哪里找。具体的过程我已经记不太清了，侯捷翻译后又免费放出电子版的一本专门讲 VXD 的书起了很大的作用。

最后我把这个东西做了出来。那时，我每天加班到晚上，早晨一睁眼就想往公司里赶。我敢保证任何人换了是我也会一样：第一次参加工作，到手的第一个任务，谁都想证实自己的价值。我不记得有多少知识被我“囫囵吞枣”了，但我确实满足了用户的需求。离奇的是，那时我还根本不会用 WinDbg，整个过程只靠“蓝屏试验”，压根就没像样地调试过。这根本无须任何学习方法。在有如此强烈需求的境况下，一切的不合理都会顺理成章。

开发驱动程序类似于为普通的软件开发一个“组件”。与别人约定好调用的接口，你就可以和别人协同合作，开发一个“dll”之类的组件去给别人使用。对操作系统内核也是如此，只不过这个接口由操作系统硬性规定，容不得协商。

在那个项目中我们写了种种功能来“愚弄”操作系统。这听起来像是一个磁盘相关的驱动，实际上却是一个文件系统过滤驱动。它过滤文件系统的行为，比如：读取文件时，我们让操作系统以为在操作普通的本地文件系统，其实却是去网络上取文件内容；而修改文件时，我们也让操作系统以为修改成功了，实际上数据并没有写入到网络上的服务器中，而是写到本地了。这是我第一次接触这类驱动程序。

有了这些项目的经验，后来就有了《天书夜读——从汇编语言到 Windows 内核编程》《寒江独钓——Windows 内核安全编程》之类的书，也有了本书。虽然这个跨度有点长，一晃七八年过去了，但这和我的拖拖拉拉有很大的关系。《寒江独钓——Windows 内核安全编程》绝版之

后，编辑一直催促我修改出新的版本。但我继续磨蹭，直到我到了腾讯电脑管家的团队，这里遇到的优秀同事愿意帮我整理已经乱七八糟的稿子，并填充最新的章节，最后耐心地审校和完善代码，本书才能最终得以面世。至于如何学习，我已经说了自己的糟糕的例子。我可以大概总结如下：

有计划和毅力是最好的。耐心地读完本书，调通上边所有的例子。虽然工作中不一定真会用到，但里面的东西都是别人实际用到过的。

没有必要害怕接受自己不熟悉的工作任务。泰山压顶的工作压力也是学习最好的动力——好吧，因为我没有前面一条，所以这一条就成了我唯一的动力了。碰到急需完成的任务，本书上有时也有可以参考的现成代码，不妨翻一翻看看。懂不懂没关系，先用了再说。

理想是美好的，可以创造一些令人意外的东西。但最后还是要着眼于实际。

写代码是一件很爽快的事，但调代码会很烦。读者如果买了本书并且打算从事这方面的工作，希望不要为调试的各种困难而烦躁，因为以后这就是工作的主要部分了。我因为第一个任务的缘故，一不小心就陷入十多年。开头几年里大多数时间我都很烦躁，常常被 BUG 逼得感觉走投无路。现在好多了，差不多有七分自信能将事做成。当然，剩下三分还是要靠天意的。

谭文

2015 年 5 月

目 录

基 础 篇

第 1 章 内核上机指导	2	2.3 重要的数据结构	21
1.1 下载和使用 WDK	2	2.3.1 驱动对象	21
1.1.1 下载并安装 WDK	2	2.3.2 设备对象	22
1.1.2 编写第一个 C 文件	4	2.3.3 请求	24
1.1.3 编译一个工程	5	2.4 函数调用	25
1.2 安装与运行	6	2.4.1 查阅帮助	25
1.2.1 下载一个安装工具	6	2.4.2 帮助中有的几类函数	26
1.2.2 运行与查看输出信息	7	2.4.3 帮助中没有的函数	28
1.2.3 在虚拟机中运行	8	2.5 Windows 的驱动开发模型	29
1.3 调试内核模块	9	2.6 WDK 编程中的特殊点	30
1.3.1 下载和安装 WinDbg	9	2.6.1 内核编程的主要调用源	30
1.3.2 设置 Windows XP 调试执行	9	2.6.2 函数的多线程安全性	30
1.3.3 设置 Vista 调试执行	10	2.6.3 代码的中断级	32
1.3.4 设置 VMware 的管道虚拟串口	11	2.6.4 WDK 中出现的特殊代码	32
1.3.5 设置 Windows 内核符号表	12	第 3 章 字符串与链表	35
1.3.6 实战调试 first	13	3.1 字符串操作	35
第 2 章 内核编程环境及其特殊性	16	3.1.1 使用字符串结构	35
2.1 内核编程的环境	16	3.1.2 字符串的初始化	36
2.1.1 隔离的应用程序	16	3.1.3 字符串的拷贝	37
2.1.2 共享的内核空间	17	3.1.4 字符串的连接	38
2.1.3 无处不在的内核模块	18	3.1.5 字符串的打印	38
2.2 数据类型	19	3.2 内存与链表	40
2.2.1 基本数据类型	19	3.2.1 内存的分配与释放	40
2.2.2 返回状态	19	3.2.2 使用 LIST_ENTRY	41
2.2.3 字符串	20	3.2.3 使用长长整型数据	43
		3.3 自旋锁	44

3.3.1 使用自旋锁	44	5.1.3 控制设备的删除	71
3.3.2 在双向链表中使用时旋锁.....	45	5.1.4 分发函数	72
3.3.3 使用队列自旋锁提高性能.....	46	5.1.5 请求的处理	73
第 4 章 文件、注册表、线程.....	47	5.2 应用方面的编程	74
4.1 文件操作	47	5.2.1 基本的功能需求	74
4.1.1 使用 OBJECT_ATTRIBUTES ...	47	5.2.2 在应用程序中打开与关闭设备....	75
4.1.2 打开和关闭文件	48	5.2.3 设备控制请求	75
4.1.3 文件读/写操作	51	5.2.4 内核中的对应处理	77
4.2 注册表操作	53	5.2.5 结合测试的效果	79
4.2.1 注册表键的打开	53	5.3 阻塞、等待与安全设计	80
4.2.2 注册表键值的读	55	5.3.1 驱动主动通知应用	80
4.2.3 注册表键值的写	57	5.3.2 通信接口的测试	81
4.3 时间与定时器	58	5.3.3 内核中的缓冲区链表结构.....	83
4.3.1 获得当前“滴答”数.....	58	5.3.4 输入：内核中的请求处理中 的安全检查.....	84
4.3.2 获得当前系统时间	58	5.3.5 输出处理与卸载清理	85
4.3.3 使用定时器	59	第 6 章 64 位和 32 位内核开发差异.....	88
4.4 线程与事件	62	6.1 64 位系统新增机制	88
4.4.1 使用系统线程	62	6.1.1 WOW64 子系统.....	88
4.4.2 在线程中睡眠	63	6.1.2 PatchGuard 技术.....	91
4.4.3 使用同步事件	64	6.1.3 64 位驱动的编译、安装与运行... 91	
第 5 章 应用与内核通信	67	6.2 编程差异	92
5.1 内核方面的编程	68	6.2.1 汇编嵌入变化	92
5.1.1 生成控制设备	68	6.2.2 预处理与条件编译	93
5.1.2 控制设备的名字和符号链接....	70	6.2.3 数据结构调整	93

开 发 篇

第 7 章 串口的过滤	96	7.3 完整的代码	105
7.1 过滤的概念	96	7.3.1 完整的分发函数	105
7.1.1 设备绑定的内核 API 之一	97	7.3.2 如何动态卸载	106
7.1.2 设备绑定的内核 API 之二	98	7.3.3 代码的编译与运行	107
7.1.3 生成过滤设备并绑定.....	98	第 8 章 键盘的过滤.....	109
7.1.4 从名字获得设备对象.....	100	8.1 技术原理	110
7.1.5 绑定所有串口	101	8.1.1 预备知识	110
7.2 获得实际数据	102	8.1.2 Windows 中从击键到内核	110
7.2.1 请求的区分	102	8.1.3 键盘硬件原理	112
7.2.2 请求的结局	103	8.2 键盘过滤的框架	112
7.2.3 写请求的数据	104	8.2.1 找到所有的键盘设备	112

8.2.2	应用设备扩展	115	9.4	EvtDriverDeviceAdd 函数	146
8.2.3	键盘过滤模块的 DriverEntry ...	117	9.4.1	EvtDriverDeviceAdd 的定义 ...	146
8.2.4	键盘过滤模块的动态卸载.....	117	9.4.2	局部变量的声明	146
8.3	键盘过滤的请求处理	119	9.4.3	磁盘设备的创建	147
8.3.1	通常的处理	119	9.4.4	如何处理发往设备的请求.....	148
8.3.2	PNP 的处理	120	9.4.5	用户配置的初始化	149
8.3.3	读的处理	121	9.4.6	链接给应用程序	151
8.3.4	读完成的处理	122	9.4.7	小结	152
8.4	从请求中打印出按键信息	123	9.5	FAT12/16 磁盘卷初始化	152
8.4.1	从缓冲区中获得 KEYBOARD_INPUT_DATA... 123		9.5.1	磁盘卷结构简介	152
8.4.2	从 KEYBOARD_INPUT_DATA 中得到键	124	9.5.2	Ramdisk 对磁盘的初始化	154
8.4.3	从 MakeCode 到实际字符.....	124	9.6	驱动中的请求处理	160
8.5	Hook 分发函数	126	9.6.1	请求的处理	160
8.5.1	获得类驱动对象	126	9.6.2	读/写请求	160
8.5.2	修改类驱动的分发函数指针... 127		9.6.3	DeviceIoControl 请求.....	162
8.5.3	类驱动之下的端口驱动.....	128	9.7	Ramdisk 的编译和安装	164
8.5.4	端口驱动和类驱动之间的协 作机制	129	9.7.1	编译	164
8.5.5	找到关键的回调函数的条件... 129		9.7.2	安装	164
8.5.6	定义常数和数据结构.....	130	9.7.3	对安装的深入探究	165
8.5.7	打开两种键盘端口驱动寻找 设备	131	第 10 章 磁盘的过滤	167	
8.5.8	搜索在 KbdClass 类驱动中的 地址	133	10.1	磁盘过滤驱动的概念	167
8.6	Hook 键盘中断反过滤.....	135	10.1.1	设备过滤和类过滤	167
8.6.1	中断: IRQ 和 INT	136	10.1.2	磁盘设备和磁盘卷设备过滤 驱动	167
8.6.2	如何修改 IDT	136	10.1.3	注册表和磁盘卷设备过滤 驱动	168
8.6.3	替换 IDT 中的跳转地址	137	10.2	具有还原功能的磁盘卷过滤驱动... 168	
8.6.4	QQ 的 PS/2 反过滤措施	139	10.2.1	简介	168
8.7	直接用端口操作键盘	139	10.2.2	基本思想	169
8.7.1	读取键盘数据和命令端口.....	139	10.3	驱动分析	169
8.7.2	p2cUserFilter 的最终实现.....	140	10.3.1	DriverEntry 函数	169
第 9 章 磁盘的虚拟	143		10.3.2	AddDevice 函数	170
9.1	虚拟的磁盘	143	10.3.3	PnP 请求的处理	174
9.2	一个具体的例子	143	10.3.4	Power 请求的处理	178
9.3	入口函数	144	10.3.5	DeviceIoControl 请求的处理... 178	
9.3.1	入口函数的定义	144	10.3.6	bitmap 的作用和分析	182
9.3.2	Ramdisk 驱动入口函数	145	10.3.7	boot 驱动完成回调函数和 稀疏文件	187
			10.3.8	读/写请求的处理	190

第 11 章 文件系统的过滤与监控	199	11.8.4 在打开请求完成之前获得 路径名.....	240
11.1 文件系统的设备对象.....	200	11.8.5 把短名转换为长名.....	242
11.1.1 控制设备与卷设备.....	200	11.9 把 sfilter 编译成静态库.....	243
11.1.2 生成自己的一个控制设备.....	201	11.9.1 如何方便地使用 sfilter.....	243
11.2 文件系统的分发函数.....	202	11.9.2 初始化回调、卸载回调和 绑定回调.....	244
11.2.1 普通的分发函数.....	202	11.9.3 绑定与回调.....	245
11.2.2 文件过滤的快速 IO 分发函数.....	203	11.9.4 插入请求回调.....	246
11.2.3 快速 IO 分发函数的一个实现.....	205	11.9.5 如何利用 sfilter.lib.....	249
11.2.4 快速 IO 分发函数逐个简介.....	206	第 12 章 文件系统透明加密	252
11.3 设备的绑定前期工作.....	207	12.1 文件透明加密的应用.....	252
11.3.1 动态地选择绑定函数.....	207	12.1.1 防止企业信息泄密.....	252
11.3.2 注册文件系统变动回调.....	208	12.1.2 文件透明加密防止企业信 息泄密.....	253
11.3.3 文件系统变动回调的一个 实现.....	209	12.1.3 文件透明加密软件的例子.....	253
11.3.4 文件系统识别器.....	211	12.2 区分进程.....	254
11.4 文件系统控制设备的绑定.....	212	12.2.1 机密进程与普通进程.....	254
11.4.1 生成文件系统控制设备的 过滤设备.....	212	12.2.2 找到进程名字的位置.....	255
11.4.2 绑定文件系统控制设备.....	213	12.2.3 得到当前进程的名字.....	256
11.4.3 利用文件系统控制请求.....	215	12.3 内存映射与文件缓冲.....	257
11.5 文件系统卷设备的绑定.....	217	12.3.1 记事本的内存映射文件.....	257
11.5.1 从 IRP 中获得 VPB 指针.....	217	12.3.2 Windows 的文件缓冲.....	258
11.5.2 设置完成函数并等待 IRP 完成.....	218	12.3.3 文件缓冲：明文还是密文 的选择.....	259
11.5.3 卷挂载 IRP 完成后的工作.....	221	12.3.4 清除文件缓冲.....	260
11.5.4 完成函数的相应实现.....	223	12.4 加密标识.....	263
11.5.5 绑定卷的实现.....	224	12.4.1 保存在文件外、文件头还是 文件尾.....	263
11.6 读/写操作的过滤.....	226	12.4.2 隐藏文件头的大小.....	264
11.6.1 设置一个读处理函数.....	226	12.4.3 隐藏文件头的设置偏移.....	266
11.6.2 设备对象的区分处理.....	227	12.4.4 隐藏文件头的读/写偏移.....	267
11.6.3 解析读请求中的文件信息.....	228	12.5 文件加密表.....	267
11.6.4 读请求的完成.....	230	12.5.1 何时进行加密操作.....	267
11.7 其他操作的过滤.....	234	12.5.2 文件控制块与文件对象.....	268
11.7.1 文件对象的生存周期.....	234	12.5.3 文件加密表的数据结构与 初始化.....	269
11.7.2 文件的打开与关闭.....	235	12.5.4 文件加密表的操作：查询.....	270
11.7.3 文件的删除.....	237	12.5.5 文件加密表的操作：添加.....	271
11.8 路径过滤的实现.....	238	12.5.6 文件加密表的操作：删除.....	272
11.8.1 取得文件路径的三种情况.....	238		
11.8.2 打开成功后获取路径.....	238		
11.8.3 在其他时刻获得文件路径.....	240		

12.6 文件打开处理	273	14.2.2 唯一的分发函数	320
12.6.1 直接发送 IRP 进行查询与 设置操作	274	14.2.3 过滤框架的实现	322
12.6.2 直接发送 IRP 进行读/写操作 ..	276	14.2.4 主要过滤的请求类型	323
12.6.3 文件的非重入打开	277	14.3 生成请求: 获取地址	324
12.6.4 文件的打开预处理	280	14.3.1 过滤生成请求	324
12.7 读/写加密和解密	285	14.3.2 准备解析 IP 地址与端口	326
12.7.1 在读取时进行解密	285	14.3.3 获取生成的 IP 地址和端口 ...	327
12.7.2 分配与释放 MDL	286	14.3.4 连接终端的生成与相关信 息的保存	329
12.7.3 写请求加密	287	14.4 控制请求	330
12.8 crypt_file 的组装	289	14.4.1 TDI_ASSOCIATE_ADDRESS 的过滤	330
12.8.1 crypt_file 的初始化	289	14.4.2 TDI_CONNECT 的过滤	332
12.8.2 crypt_file 的 IRP 预处理	290	14.4.3 其他的次功能号	333
12.8.3 crypt_file 的 IRP 后处理	293	14.4.4 设置事件的过滤	334
第 13 章 文件系统微过滤驱动	297	14.4.5 TDI_EVENT_CONNECT 类 型的设置事件的过滤	336
13.1 文件系统微过滤驱动简介	297	14.4.6 直接获取发送函数的过滤	337
13.1.1 文件系统微过滤驱动的由来 ..	297	14.4.7 清理请求的过滤	339
13.1.2 Minifilter 的优点与不足	298	14.5 本书例子 tdifw.lib 的应用	341
13.2 Minifilter 的编程框架	298	14.5.1 tdifw 库的回调接口	341
13.2.1 微文件系统过滤的注册	299	14.5.2 tdifw 库的使用例子	342
13.2.2 微过滤器的数据结构	300	第 15 章 Windows 过滤平台	345
13.2.3 卸载回调函数	303	15.1 WFP 简介	345
13.2.4 预操作回调函数	303	15.2 WFP 框架	345
13.2.5 后操作回调函数	306	15.3 基本对象模型	347
13.2.6 其他回调函数	307	15.3.1 过滤引擎	347
13.3 Minifilter 如何与应用程序通信	309	15.3.2 垫片	347
13.3.1 建立通信端口的办法	310	15.3.3 呼出接口	347
13.3.2 在用户态通过 DLL 使用通 信端口的范例	311	15.3.4 分层	348
13.4 Minifilter 的安装与加载	314	15.3.5 子层	349
13.4.1 安装 Minifilter 的 INF 文件 ..	314	15.3.6 过滤器	350
13.4.2 启动安装完成的 Minifilter ..	316	15.3.7 呼出接口回调函数	354
第 14 章 网络传输层过滤	317	15.4 WFP 操作	359
14.1 TDI 概要	317	15.4.1 呼出接口的注册与卸载	360
14.1.1 为何选择 TDI	317	15.4.2 呼出接口的添加与移除	360
14.1.2 从 socket 到 Windows 内核 ...	318	15.4.3 子层的添加与移除	361
14.1.3 TDI 过滤的代码例子	319	15.4.4 过滤器的添加	362
14.2 TDI 的过滤框架	319	15.5 WFP 过滤例子	362
14.2.1 绑定 TDI 的设备	319		

第 16 章 NDIS 协议驱动	370	16.7.4 ReceivePacketHandler 的实现...	418
16.1 以太网包和网络驱动架构.....	370	16.7.5 接收数据包的入队	420
16.1.1 以太网包和协议驱动.....	370	16.7.6 接收数据包的出队和读请求	
16.1.2 NDIS 网络驱动	371	的完成	422
16.2 协议驱动的 DriverEntry	372	第 17 章 NDIS 小端口驱动	427
16.2.1 生成控制设备	372	17.1 小端口驱动的应用与概述	427
16.2.2 注册协议	374	17.1.1 小端口驱动的应用	427
16.3 协议与网卡的绑定	375	17.1.2 小端口驱动示例	428
16.3.1 协议与网卡的绑定概念.....	375	17.1.3 小端口驱动的运作与编程	
16.3.2 绑定回调处理的实现.....	376	概述	429
16.3.3 协议绑定网卡的 API.....	378	17.2 小端口驱动的初始化	429
16.3.4 解决绑定竞争问题	379	17.2.1 小端口驱动的 DriverEntry	429
16.3.5 分配接收和发送的包池与		17.2.2 小端口驱动的适配器结构....	431
缓冲池	380	17.2.3 配置信息的读取	433
16.3.6 OID 请求的发送和请求完		17.2.4 设置小端口适配器上下文....	433
成回调	381	17.2.5 MPInitialize 的实现	434
16.3.7 ndisprotCreateBinding 的最		17.2.6 MPHalt 的实现.....	437
终实现	385	17.3 打开 ndisprot 设备	438
16.4 绑定的解除	390	17.3.1 IO 目标	438
16.4.1 解除绑定使用的 API.....	390	17.3.2 给 IO 目标发送	
16.4.2 ndisprotShutdownBinding 的		DeviceIoControl 请求	439
实现	392	17.3.3 打开 ndisprot 接口并完成	
16.5 在用户态操作协议驱动	395	配置设备	441
16.5.1 协议的收包与发包	395	17.4 使用 ndisprot 发送包	443
16.5.2 在用户态编程打开设备.....	396	17.4.1 小端口驱动的发包接口.....	443
16.5.3 用 DeviceIoControl 发送控制		17.4.2 发送控制块 (TCB)	444
请求	397	17.4.3 遍历包组并填写 TCB.....	446
16.5.4 用 WriteFile 发送数据包	399	17.4.4 写请求的构建与发送	449
16.5.5 用 ReadFile 发送数据包	400	17.5 使用 ndisprot 接收包	451
16.6 在内核态完成功能的实现.....	402	17.5.1 提交数据包的内核 API.....	451
16.6.1 请求的分发与实现	402	17.5.2 从接收控制块 (RCB)	
16.6.2 等待设备绑定完成与指定设		提交包	452
备名	402	17.5.3 对 ndisprot 读请求的完成	
16.6.3 指派设备的完成	403	函数	454
16.6.4 处理读请求	406	17.5.4 读请求的发送	456
16.6.5 处理写请求	408	17.5.5 用于读包的 WDF 工作任务... 457	
16.7 协议驱动的回回调	412	17.5.6 ndisedge 读工作任务的生成	
16.7.1 和接收包有关的回调函数....	412	与入列	459
16.7.2 ReceiveHandler 的实现.....	413	17.6 其他的特征回调函数的实现.....	461
16.7.3 TransferDataCompleteHandler		17.6.1 包的归还	461
的实现	417		

17.6.2	OID 查询处理的直接完成.....	462	19.1.2	x86 中的 mov 指令	512
17.6.3	OID 设置处理	465	19.1.3	x86 中的寄存器与内存.....	512
第 18 章 NDIS 中间层驱动.....		467	19.1.4	赋值语句的实现	513
18.1	NDIS 中间层驱动概述	467	19.2	x86 中函数的实现	514
18.1.1	Windows 网络架构总结	467	19.2.1	一个函数的例子	514
18.1.2	NDIS 中间层驱动简介	468	19.2.2	堆栈的介绍	515
18.1.3	NDIS 中间层驱动的应用	469	19.2.3	寄存器的备份和恢复	516
18.1.4	NDIS 包描述符结构深究	470	19.2.4	内部变量与返回值	518
18.2	中间层驱动的入口与绑定.....	473	19.3	x86 中函数的调用与返回	521
18.2.1	中间层驱动的入口函数.....	473	19.3.1	函数的调用指令 call.....	521
18.2.2	动态绑定 NIC 设备.....	474	19.3.2	通过堆栈传递参数	521
18.2.3	小端口初始化 (MpInitialize)	475	19.3.3	从函数返回	523
18.3	中间层驱动发送数据包	477	19.3.4	三种常见的调用协议	524
18.3.1	发送数据包原理	477	19.4	从 32 位汇编到 64 位汇编	526
18.3.2	包描述符“重利用”	478	19.4.1	Intel 64 与 IA-32 体系架构 简介	526
18.3.3	包描述符“重申请”	481	19.4.2	64 位指令与 32 位指令.....	526
18.3.4	发送数据包的异步完成.....	482	19.4.3	通用寄存器	527
18.4	中间层驱动接收数据包	484	19.5	64 位下的函数实现	528
18.4.1	接收数据包概述	484	19.5.1	函数概览	528
18.4.2	用 PtReceive 接收数据包	485	19.5.2	32 位参数的传递	529
18.4.3	用 PtReceivePacket 接收.....	490	19.5.3	64 位参数与返回值	530
18.4.4	对包进行过滤	491	19.5.4	栈空间的开辟与恢复	531
18.5	中间层驱动程序查询和设置.....	494	第 20 章 IA-32 体系中的内存地址.....		534
18.5.1	查询请求的处理	494	20.1	内存的虚拟地址	534
18.5.2	设置请求的处理	496	20.1.1	C 语言中的内存地址.....	534
18.6	NDIS 句柄.....	498	20.1.2	虚拟地址的构成	535
18.6.1	不可见的结构指针	498	20.1.3	段的选择	536
18.6.2	常见的 NDIS 句柄	499	20.2	全局描述符表和段描述符	538
18.6.3	NDIS 句柄误用问题	500	20.2.1	全局描述符表	538
18.6.4	一种解决方案	502	20.2.2	段类型	539
18.7	生成普通控制设备	503	20.2.3	段寄存器与段选择子	540
18.7.1	在中间层驱动中添加普通 设备	503	20.2.4	64 位模式下的段	541
18.7.2	使用传统方法来生成控制 设备	505	20.3	分段编程实践	542
第 19 章 IA-32 汇编基础		511	20.3.1	系统表寄存器的结构	542
19.1	x86 内存、寄存器与堆栈	511	20.3.2	在汇编语言中获取全局描 述表的位置	543
19.1.1	_asm 关键字	511	20.3.3	调试范例: sgdt 指令的错 误使用	545

20.3.4	在 64 位下获得全局描述符表	547
20.4	线性地址基础	549
20.4.1	分页控制机制	550
20.4.2	线性地址的转换	551
20.4.3	混合页面大小	552
20.4.4	32 位物理地址的页目录和页表项	552
20.5	各种特殊分页方式	555
20.5.1	PAE 分页方式	555
20.5.2	PSE-36 分页机制	558
20.5.3	IA-32e 模式下的线性地址	559
20.6	分页编程实践	562
20.6.1	页目录和页目录指针表的获取	562
20.6.2	页表的获取	564
20.6.3	线性地址的结构	567

第 21 章 处理器权限级别切换.....571

21.1	Ring0 和 Ring3 权限级别	571
21.2	保护模式下的分页内存保护	572
21.3	分页内存不可执行保护	574
21.3.1	不可执行保护原理	574
21.3.2	不可执行保护的漏洞	575
21.3.3	上机实践	577
21.4	权限级别的切换	579
21.4.1	调用门及其漏洞	579
21.4.2	sysenter 和 sysexit 指令	581
21.4.3	上机实践	583

第 22 章 IA-32 体系结构中的中断.....585

22.1	中断基础知识	585
22.1.1	中断描述符表	585
22.1.2	中断处理过程	587
22.1.3	64 位模式下的中断处理机制	589
22.1.4	多核下的中断	589
22.2	Windows 中断机制	593
22.3	中断编程实践	596
22.3.1	IDT Hook	596
22.3.2	巧用 IDT Hook 实现安全防护	598

第 23 章 Windows 内核挂钩.....601

23.1	系统服务描述符表挂钩	602
23.1.1	系统服务描述符表 (SSDT)	602
23.1.2	系统服务描述符表挂钩的意图	603
23.1.3	寻找要挂钩的函数的地址	604
23.1.4	函数被挂钩的过程	605
23.1.5	具体实现的代码	606
23.2	函数导出表挂钩	608
23.2.1	内核函数的种类	608
23.2.2	挂钩 IoCallDriver	610
23.2.3	对跳转地址进行修改	611
23.3	Windows 7 系统下 IoCallDriver 的跟踪	612
23.4	Windows 7 系统下内联挂钩	615
23.4.1	写入跳转指令并拷贝代码	615
23.4.2	实现中继函数	617

高级篇

第 24 章 Windows 通知与回调.....620

24.1	Windows 的事件通知与回调	620
24.2	常用的事件通知	620
24.2.1	创建进程通知	621
24.2.2	创建线程通知	625
24.2.3	加载模块通知	626
24.2.4	注册表操作通知	629
24.3	Windows 回调机制	636
24.3.1	回调对象	636

24.3.2	回调对象的创建	637
24.3.3	回调对象的注册	637
24.3.4	回调的通告	638
24.4	安全的死角, 回调的应用	639

第 25 章 保护进程.....640

25.1	内核对象简介	640
25.2	内核对象的结构	641
25.3	保护内核对象	642

25.3.1 处理对象的打开	643	25.4.1 保护原理	652
25.3.2 处理句柄的复制	644	25.4.2 Vista 以后的进程对象保护 ...	654
25.3.3 处理句柄的继承	646	25.4.3 进程的其他保护	655
25.4 进程的保护	652		
附录 A 如何使用本书的源码光盘	656		
附录 B 练习题	659		

基础篇

- 第 1 章 内核上机指导
- 第 2 章 内核编程环境及其特殊性
- 第 3 章 字符串与链表
- 第 4 章 文件、注册表、线程
- 第 5 章 应用与内核通信
- 第 6 章 64 位和 32 位内核开发差异