

随着网络空间安全的逐步发展以及对个人隐私的逐步重视，这几年略微降温的内核编程在个人隐私保护以及防泄密等领域渐渐升温，内核编程即将迎来第二春，而这本书不失为各位读者把握住这一技术潮流的最有利武器。

最后，在有幸作序之余，祝愿这本书能有更多的读者，能得到更多的认可，能发挥更多的作用，希望这本书能成为国内 Windows 内核编程领域的经典著作。

任晓琿

十五派信息安全教育创始人

《黑客免杀攻防》作者

黑客反病毒组织创始人

2020年2月



前 言

Windows 是当前主流的闭源操作系统，从第一个 NT 内核的 Windows 2000 至今，已经有 20 年左右的历史。在这漫长的 20 年内，为了满足日益变化的业务需求，以及应对不断升级的安全挑战，Windows 操作系统内核一直不断升级与完善，其主要表现是内核中增加了新的逻辑模块与安全机制，其中最为典型的是 64 位的 Windows 操作系统内核对比 32 位内核增加了“Patch Guard”模块，这个模块的主要作用是检查内核是否被第三方内核模块“污染”，目的是防止病毒木马使用内核挂钩或劫持的技术篡改内核。新的安全机制往往会对安全开发者带来一定的影响，其原因是一些软件过度依赖系统未公开的底层技术，而正确的做法是开发者需紧密依赖系统提供的公开机制，利用可利用的机制完成相同的功能，这要求开发者对整个 Windows 内核机制有深入的理解。作者编写本书的目的之一，正是希望读者能对 Windows 内核有更全面、更深入的认识。

本书的前身是《Windows 内核编程与驱动开发》，本书在前者的基础上，删除了部分过时的章节，重写了大部分基础章节并新增了部分当前较为热门的技术，同时为了使本书内容更为聚焦，删除了与 Windows 内核关联性不强的内容。

本书面向的人群主要有以下几类。

- 有一定 C 语言基础，有兴趣了解 Windows 内核的读者。
- 有一定 C 语言基础，并且希望从事 Windows 内核开发的读者。
- 有一定基础的 Windows 内核开发者，有意愿进一步提高的读者。

本书共分为三篇。第一篇为基础篇（第 1 章～第 6 章），从初学者的角度出发，介绍 Windows 内核的基本概念、开发环境的搭建、系统机制以及内核编程的技巧。第 1 章与第 2 章是本书最为基础的部分，介绍了内核编程的基本概念与开发环境搭建，初学者应该首先学习这部分内容。第 3～第 5 章重点介绍了系统的常用机制，这些机制的使用会贯穿本书所有章节，掌握这些常用机制是内核开发者最基本的要求。第 6 章介绍了内核编程的注意事项与技巧，这些注意事项与技巧可以帮助初学者少走弯路，快速入门。

第二篇为过滤篇（第 7 章～第 18 章），是本书的核心内容，由易到难详细介绍了 Windows 系统的过滤机制。首先以最简单的串口过滤驱动开始，剖析了一个过滤驱动的最基本要素，然后分别介绍了键盘过滤、磁盘过滤、文件过滤以及网络过滤。对于网络过滤，本篇从不同的网络层次与角度介绍了 TDI、WFP 以及 NDIS 等机制。本篇内容涵盖了目前 Windows 系统绝大部分主流的过滤技术。



第三篇为应用篇（第 19 章～第 23 章），结合前两篇的知识点，本篇综合介绍了 Windows 安全领域所需的其他技术，通过对本篇的学习，读者将会发现安全技术并不局限于系统提供的现成机制。本篇选取了目前主流安全软件所使用到的典型技术，深入浅出，首先介绍了 CPU 的基本知识点，然后基于上述知识点，第 20 章重点介绍了 Windows 下的挂钩技术，挂钩技术常被用于安全软件的检测、审计、拦截等技术；第 22 章和第 23 章从守护的角度，为读者介绍了自我保护技术。

笔者拥有十余年的 Windows 开发经验，主导过数据安全、主机安全、服务器安全等项目，涉及 To C（面向消费者群体）和 To B（面向企业用户群体）行业，深知 Windows 内核的复杂性，由于行业的特殊环境，To C 和 To B 的内核技术方案选型不尽相同，因为不同的用户群体，其主机上软件存在参差不齐的同类安全软件，安全软件之间也存在大量的兼容性问题，这些问题的引入使得原本并不简单的内核编程更为复杂化。记得有很多读者问过我一个相同的问题：如何编写一个稳定的内核模块。这个问题其实没有标准答案，以笔者的经验来看，读者首先应该养成良好的编程习惯，然后深入理解系统的各种安全机制以及拦截方法，在编写代码时，请思考如下几个问题：①这句代码是否会被其他软件拦截导致失败；②这句代码是否会触发一些第三方的回调函数；③这句代码失败后应该怎么处理。本书在为读者介绍技术的同时，也为读者介绍了笔者的内核开发心得体会与技巧，希望这些体会与技巧可以为读者带来更多的思考。内核编程类似于武林秘籍的内功修炼，需要时间沉淀，并非一蹴而就，请读者赋予更多的耐心，“成功之道，贵在坚持”。

写作是一项工作量巨大而繁杂的工作，而对于技术书籍来说更是如此。由于我个人工作的缘故，写作只能在晚上或周末进行，有时为了整章内容的思路连贯而需要挑灯夜战，通宵达旦。一路走来，感谢我的父母、妻子和女儿，在每天有限的的时间里，我需要投入更多的时间精力专注在写作上而缺少对他们的陪伴，尤其是我的女儿淇淇，在深感愧疚的同时，也感谢他们的理解与支持。

感谢我的挚友黄瀚，在本书写作过程中一直支持我，并为我提供了大量的技术性资源。感谢电子工业出版社的李冰老师和冯琦老师，她们的编辑工作确保了书本所有文字内容严谨与通畅。

感谢“安全圈”内朋友们的支持，尤其感谢数篷科技的科学家吴焯、CTO 杨一飞以及架构师王柏达，他们在工作中为我提供大量的帮助。感谢上一本书籍热心的读者，他们反馈的问题更好地完善了本书的内容。

最后，希望本书能为“安全圈”内的读者或者即将进入“安全圈”的读者带来更大的收获。

陈铭霖

2020 年 1 月于深圳



本书的作者和贡献者

本书的前身是《Windows 内核安全与驱动开发》，除了直接编写本书的作者，还有业内技术人士协同编写了部分章节，所有作者一并介绍如下。

谭文，网名楚狂人，已有十七年客户端安全软件开发经验。先后在 NEC、英特尔亚太研发有限公司、腾讯科技任职。曾从事过企业安全软件、x86 版 Android 的 houdini 项目、腾讯电脑管家、腾讯游戏安全等开发工作。对 Windows 内核有深入研究，现任腾讯科技游戏安全团队驱动程序开发负责人，专家工程师。指导了本书的主题思想，编写了核心过滤章节，并审核了所有章节。

陈铭霖，现任职数篷科技终端安全负责人，负责终端安全开发。曾任腾讯科技高级工程师，主导腾讯电脑管家客户端安全项目；深信服科技 Windows 架构师，以及虚拟化产品架构师。有十余年终端安全开发经验，覆盖 To C、To B 及 To G 行业，具有千万级 DAU 安全产品的研发经验。主导了全书内容，重写了大部分章节，以及新增了部分章节。

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部门工作。曾著有《竹林蹊径——深入浅出 Windows 驱动开发》一书。为本书贡献了若干个网络驱动相关的章节。

杨潇，曾任 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。编写了本书“磁盘驱动”相关章节。

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。编写了本书部分章节并提供了部分代码实例。

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。编写了本书“文件系统微端口过滤”一章。

本书读者反馈的 QQ 群是 4088102，想了解更多信息也可以关注微信公众号：终端安全编程。



特别致谢

在本书的著作过程中，获得了安全界内热心朋友们的鼎力支持，他们的专业知识能力分布在国内不同的安全领域，为本书的基础定位、技术重点以及应用方向给出了科学的优化性建议。下面是为本书提供过帮助的专家们。

任晓琿 (A1Pass)，十五派信息安全教育 CEO，《黑客免杀攻防》作者，一位具有远大理想抱负的技术专家，笔者的挚友，他的价值观深深影响着笔者，受邀为本书作序，并且审阅了本书的主体脉络，结合当前行业形势为本书定位给出了宝贵的建议。

李常坤，奇安信技术总监，审阅了全书的主体脉络，为本书的技术构成给出了宝贵的建议。

黄瀚 (EvilKnight)，安全技术专家，笔者的挚友，在本书最开始进行技术选型时提供了大量的安全人脉资源，使本书的技术选型可以很好地满足行业内的不同需要。在本书编写的后期，审阅了本书的主要章节，提出了很多宝贵的建议。

杨一飞，数篷科技 CTO，百度前安全专家，笔者工作中的伙伴，生活上的挚友，行事低调但对技术有着狂热追求，在工作中为笔者提供了大量的帮助以及先进的技术理念，部分先进理念提炼后也被编写到本书中。

丰生强 (@非虫)，《Android 软件安全与逆向分析》、《macOS 软件安全与逆向分析》书籍作者，审阅了本书的主要章节，站在跨纬度的系统角度为本书给出了宝贵的意见。

胡训国，准动网络科技首席技术专家，前腾讯高级安全工程师，审阅了本书第三篇，帮忙勘正内容中的一些错误。

祁伟，华为终端安全专家，腾讯前高级安全工程师，审阅了本书的第一篇，帮忙勘正了内容中的一些错误。

王淙，Windows 内核驱动专家，审阅了本书第二篇的部分章节，帮忙勘正了内容中的一些错误。

邹冠群，资深软件安全专家，为本书的技术细节提供了部分参考资料。

张天郁，数篷科技 Windows 驱动专家，上海 2345 前内核高级工程师，受邀帮忙整理了本书的部分代码，以及更正了代码中的部分错误。

周子淇，软件安全专家，为本书的技术细节提供了部分参考资料。

丁健海，高级安全研究员，软件安全专家，为本书的技术细节提供了部分参考资料。

目 录

第 1 篇 基础篇

第 1 章 内核编程环境	002	3.3 驱动异常	033
1.1 下载开发编译环境	002	3.4 字符串操作	034
1.1.1 编译环境介绍	002	3.5 链表	036
1.1.2 下载 Visual Studio 与 WDK	004	3.5.1 头节点初始化	038
1.2 编写第一个 C 文件	006	3.5.2 节点插入	038
1.2.1 通过 Visual Studio 新建工程	006	3.5.3 链表遍历	039
1.2.2 内核入口函数	007	3.5.4 节点移除	040
1.2.3 编写入口函数体	008	3.6 自旋锁	040
1.3 编译第一个驱动	010	3.6.1 使用自旋锁	040
1.3.1 通过 Visual Studio 编译	010	3.6.2 在双向链表中使⽤自旋锁	041
1.3.2 通过 WDK 直接编译	011	3.6.3 使用队列自旋锁提高性能	042
第 2 章 内核驱动运行与调试	013	3.7 内存分配	043
2.1 驱动的运行	013	3.7.1 常规内存分配	043
2.2 服务的基本操作	015	3.7.2 旁视列表	045
2.2.1 打开服务管理器	015	3.8 对象与句柄	049
2.2.2 服务的注册	016	3.9 注册表	054
2.2.3 服务的启动与停止	018	3.9.1 注册表的打开与关闭	054
2.2.4 服务的删除	019	3.9.2 注册表的修改	056
2.2.5 服务的例子	020	3.9.3 注册表的读取	057
2.2.6 服务小结	022	3.10 文件操作	060
2.3 驱动的调试	022	3.10.1 文件的打开与关闭	060
2.3.1 基于 VS+WDK 环境调试	022	3.10.2 文件的读写	063
2.3.2 基于 Windbg 调试	026	3.11 线程与事件	066
		3.11.1 使用系统线程	066
第 3 章 内核编程基础	029	3.11.2 使用同步事件	067
3.1 上下文环境	029	第 4 章 应用与内核通信	070
3.2 中断请求级别	031	4.1 内核方面的编程	071
		4.1.1 生成控制设备	071



4.1.2 控制设备的名字和符号链接	073	7.1.3 生成过滤设备并绑定	108
4.1.3 控制设备的删除	074	7.1.4 从名字获得设备对象	110
4.1.4 分发函数	074	7.1.5 绑定所有串口	111
4.1.5 请求的处理	076	7.2 获得实际数据	112
4.2 应用方面的编程	077	7.2.1 请求的区分	112
4.2.1 基本的功能需求	077	7.2.2 请求的结局	113
4.2.2 在应用程序中打开与关闭设备	077	7.2.3 写请求的数据	114
4.2.3 设备控制请求	078	7.3 完整的代码	114
4.2.4 内核中的对应处理	080	7.3.1 完整的分发函数	114
4.2.5 结合测试的效果	082	7.3.2 如何动态卸载	116
第 5 章 64 位和 32 位内核开发差异	083	7.3.3 代码的编译与运行	117
5.1 64 位系统新增机制	083	第 8 章 键盘的过滤	119
5.1.1 WOW64 子系统	083	8.1 技术原理	120
5.1.2 PatchGuard 技术	086	8.1.1 预备知识	120
5.1.3 64 位驱动的编译、安装与运行	086	8.1.2 Windows 中从击键到内核	120
5.2 编程差异	087	8.1.3 键盘硬件原理	122
5.2.1 汇编嵌入变化	087	8.2 键盘过滤的框架	122
5.2.2 预处理与条件编译	088	8.2.1 找到所有的键盘设备	122
5.2.3 数据结构调整	088	8.2.2 应用设备扩展	125
第 6 章 内核编程技巧	090	8.2.3 键盘过滤模块的 DriverEntry	127
6.1 初始化赋值问题	090	8.2.4 键盘过滤模块的动态卸载	127
6.2 有效性判断	091	8.3 键盘过滤的请求处理	129
6.3 一次性申请	092	8.3.1 通常的处理	129
6.4 独立性与最小化原则	095	8.3.2 PNP 的处理	130
6.5 嵌套陷阱	097	8.3.3 读的处理	131
6.6 稳定性处理	098	8.3.4 读完成的处理	132
6.6.1 事前处理	098	8.4 从请求中打印出按键信息	133
6.6.2 事中处理	100	8.4.1 从缓冲区中获得 KEYBOARD_	
6.6.3 事后处理	104	INPUT_DATA	133
第 2 篇 过滤篇		8.4.2 从 KEYBOARD_INPUT_DATA	
第 7 章 串口的过滤	106	中得到键	134
7.1 过滤的概念	106	8.4.3 从 MakeCode 到实际字符	134
7.1.1 设备绑定的内核 API 之一	106	8.5 Hook 分发函数	136
7.1.2 设备绑定的内核 API 之二	107	8.5.1 获得类驱动对象	136
7.1.3 生成过滤设备并绑定	108	8.5.2 修改类驱动的分发函数指针	137
7.1.4 从名字获得设备对象	110	8.5.3 类驱动之下的端口驱动	138
7.1.5 绑定所有串口	111		
7.2 获得实际数据	112		
7.2.1 请求的区分	112		
7.2.2 请求的结局	113		
7.2.3 写请求的数据	114		
7.3 完整的代码	114		
7.3.1 完整的分发函数	114		
7.3.2 如何动态卸载	116		
7.3.3 代码的编译与运行	117		

8.5.4	端口驱动和类驱动之间的协作机制	139	9.7	Ramdisk 的编译和安装	175
8.5.5	找到关键的回调函数的条件	140	9.7.1	编译	175
8.5.6	定义常数和数据结构	140	9.7.2	安装	175
8.5.7	打开两种键盘端口驱动寻找设备	141	9.7.3	对安装的深入探究	175
8.5.8	搜索在 KbdClass 类驱动中的地址	143	第 10 章	磁盘的过滤	177
8.6	Hook 键盘中断反过滤	145	10.1	磁盘过滤驱动的概念	177
8.6.1	中断: IRQ 和 INT	146	10.1.1	设备过滤和类过滤	177
8.6.2	如何修改 IDT	147	10.1.2	磁盘设备和磁盘卷设备过滤驱动	177
8.6.3	替换 IDT 中的跳转地址	148	10.1.3	注册表和磁盘卷设备过滤驱动	178
8.6.4	QQ 的 PS/2 反过滤措施	149	10.2	具有还原功能的磁盘卷过滤驱动	178
8.7	直接用端口操作键盘	150	10.2.1	简介	178
8.7.1	读取键盘数据和命令端口	150	10.2.2	基本思想	179
8.7.2	p2cUserFilter 的最终实现	151	10.3	驱动分析	179
第 9 章	磁盘的虚拟	153	10.3.1	DriverEntry 函数	179
9.1	虚拟的磁盘	153	10.3.2	AddDevice 函数	180
9.2	一个具体的例子	153	10.3.3	PnP 请求的处理	184
9.3	入口函数	154	10.3.4	Power 请求的处理	188
9.3.1	入口函数的定义	154	10.3.5	DeviceIoControl 请求的处理	189
9.3.2	Ramdisk 驱动入口函数	155	10.3.6	bitmap 的作用和分析	192
9.4	EvtDriverDeviceAdd 函数	156	10.3.7	boot 驱动完成回调函数和稀疏文件	198
9.4.1	EvtDriverDeviceAdd 的定义	156	10.3.8	读/写请求的处理	200
9.4.2	局部变量的声明	157	第 11 章	文件系统的过滤与监控	209
9.4.3	磁盘设备的创建	157	11.1	文件系统的设备对象	210
9.4.4	如何处理发往设备的请求	158	11.1.1	控制设备与卷设备	210
9.4.5	用户配置的初始化	160	11.1.2	生成自己的一个控制设备	211
9.4.6	链接给应用程序	161	11.2	文件系统的分发函数	212
9.5	FAT12/16 磁盘卷初始化	163	11.2.1	普通的分发函数	212
9.5.1	磁盘卷结构简介	163	11.2.2	文件过滤的快速 IO 分发函数	213
9.5.2	Ramdisk 对磁盘的初始化	164	11.2.3	快速 IO 分发函数的一个实现	215
9.6	驱动中的请求处理	170	11.2.4	快速 IO 分发函数逐个简介	216
9.6.1	请求的处理	170	11.3	设备的绑定前期工作	217
9.6.2	读/写请求	171	11.3.1	动态地选择绑定函数	217
9.6.3	DeviceIoControl 请求	172			



11.3.2	注册文件系统变动回调	219	11.9.4	插入请求回调	257
11.3.3	文件系统变动回调的一个实现	220	11.9.5	如何利用 sfilter.lib	259
11.3.4	文件系统识别器	221	第 12 章 文件系统透明加密 263		
11.4	文件系统控制设备的绑定	222	12.1	文件透明加密的应用	263
11.4.1	生成文件系统控制设备的过滤设备	222	12.1.1	防止企业信息泄密	263
11.4.2	绑定文件系统控制设备	223	12.1.2	文件透明加密防止企业信息泄密	263
11.4.3	利用文件系统控制请求	225	12.1.3	文件透明加密软件的例子	264
11.5	文件系统卷设备的绑定	227	12.2	区分进程	265
11.5.1	从 IRP 中获得 VPB 指针	227	12.2.1	机密进程与普通进程	265
11.5.2	设置完成函数并等待 IRP 完成	228	12.2.2	找到进程名字的位置	266
11.5.3	卷挂载 IRP 完成后的工作	231	12.2.3	得到当前进程的名字	267
11.5.4	完成函数的相应实现	233	12.3	内存映射与文件缓冲	268
11.5.5	绑定卷的实现	234	12.3.1	记事本的内存映射文件	268
11.6	读/写操作的过滤	236	12.3.2	Windows 的文件缓冲	269
11.6.1	设置一个读处理函数	236	12.3.3	文件缓冲：明文还是密文的选择	270
11.6.2	设备对象的区分处理	237	12.3.4	清除文件缓冲	271
11.6.3	解析读请求中的文件信息	238	12.4	加密标识	274
11.6.4	读请求的完成	241	12.4.1	保存在文件外、文件头还是文件尾	274
11.7	其他操作的过滤	244	12.4.2	隐藏文件头的大小	275
11.7.1	文件对象的生存周期	244	12.4.3	隐藏文件头的设置偏移	277
11.7.2	文件的打开与关闭	245	12.4.4	隐藏文件头的读/写偏移	277
11.7.3	文件的删除	247	12.5	文件加密表	278
11.8	路径过滤的实现	248	12.5.1	何时进行加密操作	278
11.8.1	取得文件路径的三种情况	248	12.5.2	文件控制块与文件对象	279
11.8.2	打开成功后获取路径	249	12.5.3	文件加密表的数据结构与初始化	280
11.8.3	在其他时刻获得文件路径	250	12.5.4	文件加密表的操作：查询	281
11.8.4	在打开请求完成之前获得路径名	251	12.5.5	文件加密表的操作：添加	282
11.8.5	把短名转换为长名	253	12.5.6	文件加密表的操作：删除	283
11.9	把 sfilter 编译成静态库	254	12.6	文件打开处理	284
11.9.1	如何方便地使用 sfilter	254	12.6.1	直接发送 IRP 进行查询与设置操作	285
11.9.2	初始化回调、卸载回调和绑定回调	254	12.6.2	直接发送 IRP 进行读/写操作	287
11.9.3	绑定与回调	256			

12.6.3	文件的非重入打开	288	14.2.2	唯一的分发函数	331
12.6.4	文件的打开预处理	291	14.2.3	过滤框架的实现	333
12.7	读/写加密和解密	296	14.2.4	主要过滤的请求类型	335
12.7.1	在读取时进行解密	296	14.3	生成请求: 获取地址	336
12.7.2	分配与释放 MDL	297	14.3.1	过滤生成请求	336
12.7.3	写请求加密	298	14.3.2	准备解析 IP 地址与端口	337
12.8	crypt_file 的组装	300	14.3.3	获取生成的 IP 地址和端口	338
12.8.1	crypt_file 的初始化	300	14.3.4	连接终端的生成与相关信息 的保存	340
12.8.2	crypt_file 的 IRP 预处理	301	14.4	控制请求	341
12.8.3	crypt_file 的 IRP 后处理	304	14.4.1	TDI_ASSOCIATE_ADDRESS 的过滤	341
第 13 章	文件系统微过滤驱动	308	14.4.2	TDI_CONNECT 的过滤	343
13.1	文件系统微过滤驱动简介	308	14.4.3	其他的次功能号	344
13.1.1	文件系统微过滤驱动的由来	308	14.4.4	设置事件的过滤	345
13.1.2	Minifilter 的优点与不足	309	14.4.5	TDI_EVENT_CONNECT 类型 的设置事件的过滤	346
13.2	Minifilter 的编程框架	309	14.4.6	直接获取发送函数的过滤	348
13.2.1	微文件系统过滤的注册	310	14.4.7	清理请求的过滤	350
13.2.2	微过滤器的数据结构	311	14.5	本书例子 tdifw.lib 的应用	351
13.2.3	卸载回调函数	314	14.5.1	tdifw 库的回调接口	351
13.2.4	预操作回调函数	314	14.5.2	tdifw 库的使用例子	353
13.2.5	后操作回调函数	317	第 15 章	Windows 过滤平台	355
13.2.6	其他回调函数	318	15.1	WFP 简介	355
13.3	Minifilter 如何与应用程序通信	320	15.2	WFP 框架	355
13.3.1	建立通信端口的的方法	320	15.3	基本对象模型	357
13.3.2	在用户态通过 DLL 使用通信 端口的范例	322	15.3.1	过滤引擎	357
13.4	Minifilter 的安装与加载	325	15.3.2	垫片	357
13.4.1	安装 Minifilter 的 INF 文件	325	15.3.3	呼出接口	357
13.4.2	启动安装完成的 Minifilter	326	15.3.4	分层	358
第 14 章	网络传输层过滤	328	15.3.5	子层	359
14.1	TDI 概要	328	15.3.6	过滤器	360
14.1.1	为何选择 TDI	328	15.3.7	呼出接口回调函数	364
14.1.2	从 socket 到 Windows 内核	329	15.4	WFP 操作	369
14.1.3	TDI 过滤的代码例子	330	15.4.1	呼出接口的注册与卸载	369
14.2	TDI 的过滤框架	330	15.4.2	呼出接口的添加与移除	370
14.2.1	绑定 TDI 的设备	330			



15.4.3	子层的添加与移除	371	16.6.3	指派设备的完成	413
15.4.4	过滤器的添加	372	16.6.4	处理读请求	416
15.5	WFP 过滤例子	372	16.6.5	处理写请求	418
第 16 章 NDIS 协议驱动			16.7 协议驱动接收回调		
16.1	以太网包和网络驱动架构	380	16.7.1	和接收包有关的回调函数	422
16.1.1	以太网包和协议驱动	380	16.7.2	ReceiveHandler 的实现	423
16.1.2	NDIS 网络驱动	381	16.7.3	TransferDataCompleteHandler 的实现	427
16.2	协议驱动的 DriverEntry	382	16.7.4	ReceivePacketHandler 的实现	428
16.2.1	生成控制设备	382	16.7.5	接收数据包的入队	430
16.2.2	注册协议	383	16.7.6	接收数据包的出队和读请求的完成	432
16.3	协议与网卡的绑定	385	第 17 章 NDIS 小端口驱动		
16.3.1	协议与网卡的绑定概念	385	17.1	小端口驱动的应用与概述	437
16.3.2	绑定回调处理的实现	386	17.1.1	小端口驱动的应用	437
16.3.3	协议绑定网卡的 API	388	17.1.2	小端口驱动示例	438
16.3.4	解决绑定竞争问题	389	17.1.3	小端口驱动的运作与编程概述	438
16.3.5	分配接收和发送的包池与缓冲池	390	17.2	小端口驱动的初始化	439
16.3.6	OID 请求的发送和请求完成回调	391	17.2.1	小端口驱动的 DriverEntry	439
16.3.7	ndisprotCreateBinding 的最终实现	395	17.2.2	小端口驱动的适配器结构	441
16.4	绑定的解除	400	17.2.3	配置信息的读取	442
16.4.1	解除绑定使用的 API	400	17.2.4	设置小端口适配器上下文	443
16.4.2	ndisprotShutdownBinding 的实现	402	17.2.5	MPInitialize 的实现	444
16.5	在用户态操作协议驱动	405	17.2.6	MPHalt 的实现	447
16.5.1	协议的收包与发包	405	17.3	打开 ndisprot 设备	447
16.5.2	在用户态编程打开设备	405	17.3.1	IO 目标	447
16.5.3	用 DeviceIoControl 发送控制请求	407	17.3.2	给 IO 目标发送 DeviceIoControl 请求	449
16.5.4	用 WriteFile 发送数据包	409	17.3.3	打开 ndisprot 接口并完成配置设备	451
16.5.5	用 ReadFile 发送数据包	410	17.4	使用 ndisprot 发送包	453
16.6	在内核态完成功能的实现	412	17.4.1	小端口驱动的发包接口	453
16.6.1	请求的分发与实现	412	17.4.2	发送控制块 (TCB)	454
16.6.2	等待设备绑定完成与指定设备名	412	17.4.3	遍历包组并填写 TCB	456
			17.4.4	写请求的构建与发送	458

