

<<天书夜读>>

图书基本信息

书名：<<天书夜读>>

13位ISBN编号：9787121073397

10位ISBN编号：7121073390

出版时间：2008-10

出版时间：电子工业出版社

作者：谭文，邵坚磊 著

页数：270

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

前言

Windows是庞大复杂的系统。

由于Windows并不公开源代码，我们在调试程序的时候，往往就调到自己未知的领域去了。

没有C代码，只能看到令人眼花缭乱的汇编指令和机器码。

我曾对它们望而生畏，敬而远之。

尤其在这个黑客、破解、病毒、木马横行的时代，如果作为安全软件的开发者的作者，同样不能期盼病毒的作者提供可以阅读的高级语言代码。

如果那些东西，也和C语言一样亲切易懂，那多么好啊！

这样的话，即便是Windows这样庞大复杂而且封闭的系统，或者是再诡异和隐蔽的破坏技术，至少只要我愿意去探索，对我来说就不再秘密可言。

其实这个梦想并非不切实际。

既然我们能读懂C代码，何以就不能读懂汇编呢？

很多高手眼中，机器指令和C代码一样熟悉。

这本书并没有系统地介绍Windows系统底层。

但是我尝试寻找正确的方法和手段，为读者打开Windows底层知识宝库的大门，使读者可以在其中自由阅读，自己去获取所需知识。

内容概要

本书从基本的Windows程序与汇编指令出发，深入浅出地讲解了Windows内核的编程、调试、阅读，以及自行探索的方法。

读者在使用C/C++开发Windows程序的基础上，将熟练掌握汇编和C语言的应用，深入了解Windows底层，并掌握阅读Windows内核的基本方法，以及Windows内核的基本编程方法。

本书适合使用C/C++在Windows上编程的读者，尤其适合希望加深自己技术功底的Windows应用程序员、计算机专业的有志于软件开发的大中院校学生；专业的Windows内核程序员，亦可从本书得到超越一般内核程序开发的启发。

作者简介

谭文：从2002年到2008年，从事信息安全类软件的Windows内核驱动的开发工作。从2008年开始参与一个主要涉及不同架构之间二进制指令的实时翻译技术的项目的开发。业余时间驱动开发音（www.driverdevelop.com）以楚狂人为笔名发表了许多技术文章。

书籍目录

入手篇 熟悉汇编	第1章 汇编指令与C语言	1.1 上机建立第一个工程	1.1.1 用Visual Studio创建工程
	1.1.2 用Visual Studio查看汇编代码	1.2 简要复习常用的汇编指令	1.2.1 堆栈相关指令
	1.2.2 数据传送指令	1.2.3 跳转与比较指令	1.2.2 do循环
	1.2.3 跳转与比较指令	1.3 C函数的参数传递过程	1.2.3 while循环
	2.1 C语言的循环反汇编	2.1.1 for循环	2.1.2 do循环
	2.1.1 for循环	2.1.2 do循环	2.1.3 while循环
	2.2 C语言判断与分支的反汇编	2.2.1 if-else判断分支	2.2.2 switch-case判断分支
	2.2.1 if-else判断分支	2.2.2 switch-case判断分支	
	2.3 C语言的数组与结构	2.4 C语言的共用体和枚举类型	第3章 练习反汇编C语言程序
	2.4 C语言的共用体和枚举类型	第3章 练习反汇编C语言程序	3.1 算法的反汇编
	3.1.1 算法反汇编代码分析	3.1.2 算法反汇编阅读技巧	3.2 发行版的反汇编
	3.1.2 算法反汇编阅读技巧	3.2 发行版的反汇编	
	3.3 汇编反C语言练习基础篇	内核编程	第4章 内核字符串与内存
	4.1 字符串的处理	4.1.1 使用字符串结构	4.1.2 字符串的初始化
	4.1.1 使用字符串结构	4.1.2 字符串的初始化	4.1.3 字符串的拷贝
	4.1.2 字符串的初始化	4.1.3 字符串的拷贝	4.1.4 字符串的连接
	4.1.3 字符串的拷贝	4.1.4 字符串的连接	4.1.5 字符串的打印
	4.1.4 字符串的连接	4.1.5 字符串的打印	4.2 内存与链表
	4.1.5 字符串的打印	4.2 内存与链表	4.2.1 内存的分配与释放
	4.2 内存与链表	4.2.1 内存的分配与释放	4.2.2 使用LIST_ENTRY
	4.2.1 内存的分配与释放	4.2.2 使用LIST_ENTRY	4.2.3 使用长长整型数据
	4.2.2 使用LIST_ENTRY	4.2.3 使用长长整型数据	4.2.4 使用自选锁
	4.2.3 使用长长整型数据	4.2.4 使用自选锁	第5章 文件与注册表操作
	4.2.4 使用自选锁	第5章 文件与注册表操作	5.1 文件操作
	5.1 文件操作	5.1.1 使用OBJECT_ATTRIBUTES	5.1.2 打开和关闭文件
	5.1.1 使用OBJECT_ATTRIBUTES	5.1.2 打开和关闭文件	5.1.3 文件读/写操作
	5.1.2 打开和关闭文件	5.1.3 文件读/写操作	5.2 注册表操作
	5.1.3 文件读/写操作	5.2 注册表操作	5.2.1 注册表键的打开
	5.2 注册表操作	5.2.1 注册表键的打开	5.2.2 注册表值的读
	5.2.1 注册表键的打开	5.2.2 注册表值的读	5.2.3 注册表值的写
	5.2.2 注册表值的读	5.2.3 注册表值的写	第6章 时间与线程
	5.2.3 注册表值的写	第6章 时间与线程	6.1 时间与定时器
	6.1 时间与定时器	6.1.1 获得当前滴答数	6.1.2 获得当前系统时间
	6.1.1 获得当前滴答数	6.1.2 获得当前系统时间	6.1.3 使用定时器
	6.1.2 获得当前系统时间	6.1.3 使用定时器	6.2 线程与事件
	6.1.3 使用定时器	6.2 线程与事件	6.2.1 使用系统线程
	6.2 线程与事件	6.2.1 使用系统线程	6.2.2 在线程中睡眠
	6.2.1 使用系统线程	6.2.2 在线程中睡眠	6.2.3 使用同步事件
	6.2.2 在线程中睡眠	6.2.3 使用同步事件	第7章 驱动、设备与请求
	6.2.3 使用同步事件	第7章 驱动、设备与请求	7.1 驱动与设备
	7.1 驱动与设备	7.1.1 驱动入口与驱动对象	7.1.2 分发函数和卸载函数
	7.1.1 驱动入口与驱动对象	7.1.2 分发函数和卸载函数	7.1.3 设备与符号链接
	7.1.2 分发函数和卸载函数	7.1.3 设备与符号链接	7.1.4 设备的安全创建
	7.1.3 设备与符号链接	7.1.4 设备的安全创建	7.1.5 设备与符号链接的用户相关性
	7.1.4 设备的安全创建	7.1.5 设备与符号链接的用户相关性	7.2 请求处理
	7.1.5 设备与符号链接的用户相关性	7.2 请求处理	7.2.1 IRP与IO_STACK_LOCATION
	7.2 请求处理	7.2.1 IRP与IO_STACK_LOCATION	7.2.2 打开与关闭请求的处理
	7.2.1 IRP与IO_STACK_LOCATION	7.2.2 打开与关闭请求的处理	7.2.3 应用层信息传入
	7.2.2 打开与关闭请求的处理	7.2.3 应用层信息传入	7.2.4 驱动层信息传出探索篇
	7.2.3 应用层信息传入	7.2.4 驱动层信息传出探索篇	研究内核
	7.2.4 驱动层信息传出探索篇	研究内核	第8章 进入Windows内核
	8.1 开始Windows内核编程	8.1.1 内核编程的环境准备	8.1.2 用C语言写一个内核程序
	8.1.1 内核编程的环境准备	8.1.2 用C语言写一个内核程序	8.2 学习用WinDbg进行调试
	8.1.2 用C语言写一个内核程序	8.2 学习用WinDbg进行调试	8.2.1 软件的准备
	8.2 学习用WinDbg进行调试	8.2.1 软件的准备	8.2.2 设置Windows XP调试执行
	8.2.1 软件的准备	8.2.2 设置Windows XP调试执行	8.2.3 设置VMWare虚拟机调试
	8.2.2 设置Windows XP调试执行	8.2.3 设置VMWare虚拟机调试	8.2.4 设置被调试机为Vista的情况
	8.2.3 设置VMWare虚拟机调试	8.2.4 设置被调试机为Vista的情况	8.2.5 设置Windows内核符号表
	8.2.4 设置被调试机为Vista的情况	8.2.5 设置Windows内核符号表	8.2.6 调试例子diskperf
	8.2.5 设置Windows内核符号表	8.2.6 调试例子diskperf	8.3 认识内核代码函数调用方式
	8.2.6 调试例子diskperf	8.3 认识内核代码函数调用方式	8.4 尝试反写C内核代码
	8.3 认识内核代码函数调用方式	8.4 尝试反写C内核代码	8.5 如何在代码中寻找需要的信息
	8.4 尝试反写C内核代码	8.5 如何在代码中寻找需要的信息	第9章 用C++编写的内核程序
	8.5 如何在代码中寻找需要的信息	第9章 用C++编写的内核程序	9.1 用C++开发内核程序
	9.1 用C++开发内核程序	9.1.1 建立一个C++的内核工程	9.1.2 使用C接口标准声明
	9.1.1 建立一个C++的内核工程	9.1.2 使用C接口标准声明	9.1.3 使用类静态成员函数
	9.1.2 使用C接口标准声明	9.1.3 使用类静态成员函数	9.1.4 实现new操作符
	9.1.3 使用类静态成员函数	9.1.4 实现new操作符	9.2 开始阅读一个反汇编的类
	9.1.4 实现new操作符	9.2 开始阅读一个反汇编的类	9.2.1 new操作符的实现
	9.2 开始阅读一个反汇编的类	9.2.1 new操作符的实现	9.2.2 构造函数的实现
	9.2.1 new操作符的实现	9.2.2 构造函数的实现	9.3 了解更多的C++特性
	9.2.2 构造函数的实现	9.3 了解更多的C++特性	第10章 继续探索Windows内核
	9.3 了解更多的C++特性	第10章 继续探索Windows内核	10.1 探索Windows已有内核调用
	10.1 探索Windows已有内核调用	10.2 自己实现XP的新调用	10.2.1 对照调试结果和数据结构
	10.2 自己实现XP的新调用	10.2.1 对照调试结果和数据结构	10.2.2 写出C语言的对应代码
	10.2.1 对照调试结果和数据结构	10.2.2 写出C语言的对应代码	10.3 没有符号表的情况
	10.2.2 写出C语言的对应代码	10.3 没有符号表的情况	10.4 64位操作系统下的情况
	10.3 没有符号表的情况	10.4 64位操作系统下的情况	10.4.1 分析64位操作系统的调用
	10.4 64位操作系统下的情况	10.4.1 分析64位操作系统的调用	10.4.2 深入了解64位内核调用参数传递深入篇
	10.4.1 分析64位操作系统的调用	10.4.2 深入了解64位内核调用参数传递深入篇	修改内核
	10.4.2 深入了解64位内核调用参数传递深入篇	修改内核	第11章 机器码与反汇编引擎
	11.1 了解Intel的机器码	11.1.1 可执行指令与数据	11.1.2 单条指令的组成
	11.1.1 可执行指令与数据	11.1.2 单条指令的组成	11.1.3 MOD-REG-R/M的组成
	11.1.2 单条指令的组成	11.1.3 MOD-REG-R/M的组成	11.1.4 其他的组成部分
	11.1.3 MOD-REG-R/M的组成	11.1.4 其他的组成部分	11.2 反汇编引擎XDE32基本数据结构
	11.1.4 其他的组成部分	11.2 反汇编引擎XDE32基本数据结构	11.3 反汇编引擎XDE32具体实现
	11.2 反汇编引擎XDE32基本数据结构	11.3 反汇编引擎XDE32具体实现	第12章 CPU权限级与分页机制
	11.3 反汇编引擎XDE32具体实现	第12章 CPU权限级与分页机制	12.1 Ring0和Ring3权限级
	12.1 Ring0和Ring3权限级	12.2 保护模式下的分页内存保护	12.3 分页内存不可执行保护
	12.2 保护模式下的分页内存保护	12.3 分页内存不可执行保护	12.3.1 不可执行保护原理
	12.3 分页内存不可执行保护	12.3.1 不可执行保护原理	12.3.2 不可执行保护的漏洞
	12.3.1 不可执行保护原理	12.3.2 不可执行保护的漏洞	12.4 权限级别的切换
	12.3.2 不可执行保护的漏洞	12.4 权限级别的切换	12.4.1 调用门及其漏洞
	12.4 权限级别的切换	12.4.1 调用门及其漏洞	12.4.2 sysenter和sysexit指令
	12.4.1 调用门及其漏洞	12.4.2 sysenter和sysexit指令	第13章 开发Windows内核Hook
	12.4.2 sysenter和sysexit指令	第13章 开发Windows内核Hook	13.1 XP下Hook系统调用IoCallDriver
	13.1 XP下Hook系统调用IoCallDriver	13.2 Vista下IoCallDriver的跟踪	13.3 Vista下inline hook
	13.2 Vista下IoCallDriver的跟踪	13.3 Vista下inline hook	13.3.1 写入跳转指令并拷贝代码
	13.3 Vista下inline hook	13.3.1 写入跳转指令并拷贝代码	13.3.2 实现中继函数实战篇
	13.3.1 写入跳转指令并拷贝代码	13.3.2 实现中继函数实战篇	实际开发
	13.3.2 实现中继函数实战篇	实际开发	第14章 反病毒、木马实例开发
	14.1 反病毒、木马的设想	14.2 开发内核驱动	14.2.1 在内核中检查可执行文件
	14.2 开发内核驱动	14.2.1 在内核中检查可执行文件	14.2.2 在内核中生成设备接口
	14.2.1 在内核中检查可执行文件	14.2.2 在内核中生成设备接口	14.2.3 在内核中等待监控进程的响应
	14.2.2 在内核中生成设备接口	14.2.3 在内核中等待监控进程的响应	14.3 开发监控进程
	14.2.3 在内核中等待监控进程的响应	14.3 开发监控进程	14.4 本软件进一步展望
	14.3 开发监控进程	14.4 本软件进一步展望	第15章 Rootkit与HIPS
	14.4 本软件进一步展望	第15章 Rootkit与HIPS	15.1 Rootkit为何很重要
	15.1 Rootkit为何很重要	15.2 Rootkit如何逃过检测	15.3 HIPS如何检测Rootkit
	15.2 Rootkit如何逃过检测	15.3 HIPS如何检测Rootkit	第16章 手写指令保护代码
	15.3 HIPS如何检测Rootkit	第16章 手写指令保护代码	16.1 混淆字符串
	16.1 混淆字符串	16.2 隐藏内核函数	16.3 混淆流程与数据操作
	16.2 隐藏内核函数	16.3 混淆流程与数据操作	16.3.1 混淆函数出口
	16.3 混淆流程与数据操作	16.3.1 混淆函数出口	16.3.2 插入有意义的花指令
	16.3.1 混淆函数出口	16.3.2 插入有意义的花指令	第17章 用VMProtect保护代码
	16.3.2 插入有意义的花指令	第17章 用VMProtect保护代码	17.1 安装VMProtect
	17.1 安装VMProtect	17.2 使用VMProtect	17.3 查看VMProtect效果参考文献
	17.2 使用VMProtect	17.3 查看VMProtect效果参考文献	

章节摘录

第10章 继续探索Windows内核 10.1 探索Windows已有内核调用 基础知识 下面用本书前面介绍的知识来做一些有用的事情。

Windows从2000发展到XP后，XP DDK中出现了一些新的调用。

内核程序开发者有时会发现，这些调用非常有用（这也是这些新调用产生的原因），但是如果使用这些调用，会导致驱动在Windows 2000下无法使用。

目前Windows 2000的用户依然很多，存在这样的可移植性问题是非常遗憾的，退一步的方案是Windows 2000下限制某些功能。

在程序中动态加载系统调用，并小心地判断当前的版本。

在Windows 2000的情况下，一些功能被跳过。

这样比前者好。

但是依然不是最理想的解决方案。

有一些人开始使用未公开（Undocumented）的解决方案。

未公开的解决方案的危害就是，可能不兼容未来的操作系统。

但是由于现在这样做仅仅是针对一个过去版本的操作系统，问题就不复存在了。

在新版本的操作系统上，调用新的系统调用，而在某个已经存在而且不会再变的操作系统版本上，调用未公开的方法。

只要低版本操作系统测试无问题，以后的也不会有问题。

这里的所谓未公开的解决方案就是，在新版本的操作系统上，把新出现的功能调用进行调试，了解其实现的方法。

然后在我们自己开发的内核程序中，实现同样的或者等价的功能，同时检测当前操作系统为低版本时，调用这些特殊的代码。

下面举两个例子。

这个调用获得设备栈底的设备，这对过滤驱动非常有用。

编辑推荐

我作为驱动开发的老兵，深感资料缺乏的艰辛，很多信息无法在文档中找到，此时自力更生的能力更加重要。

当我们手中拿着神兵利器——WinDbg时，一切都在掌握之中。这本书将要告诉您的不是全面的汇编知识或未公开的Windows秘密，而是怎么从这些貌似天书的汇编代码中，一探Windows底层的核心实现。

在开发中出现的问题，能不能从Windows自身找到答案?如果您正在这样思考，无疑本书是为您度身订做的。

本书授人以渔，也授人以鱼；短小精悍，读之如一缕清风，读罢则有醍醐灌顶之感。

——驱动开发网站长 马勇 (znsoft)

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>