文章编号:1001 - 9081(2009)08 - 2146 - 03

一种介质设备控制方法

马金鑫[1,2],袁 丁[1]
(1. 四川师范大学 计算机科学学院,成都 610101; 2. 四川省计算机研究院 电子自动研究所,成都 610041)
(majinxin2003@126.com)

摘 要:研究了传统的以 WDM 过滤驱动的方法来实现的设备控制原理。通过对这种传统方法的阐述说明了这种方法存在不够安全、不够灵活等缺点,对微软 Windows 的高中断级别的自旋锁源码进行反汇编分析,然后以 NT 过滤驱动的方式尝试实现 WDM 过滤驱动的机制,最后利用内核例程 hook 的方法,使用对驱动对象的派遣例程函数入口地址替换提出一种新的实现思路。经过仿真实验表明,这种方法能取得较好的效果。

关键词:过滤驱动;小型计算机系统接口;驱动对象;完成例程
中图分类号: TP316 **文献标志码**:A

One method of media device control

MA Jin-xin[1,2], YUAN Ding[1]
(1. *School of Computer Science, Sichuan Normal University, Chengdu Sichuan* 610101, *China*;
2. *Electronic Technology Institute, Sichuan Institute of Computer Science, Chengdu Sichuan* 610041, *China*)

Abstract: The traditional principle of media device control in Windows Device Model (WDM) filter driver was researched and its unsafe and inflexible shortcomings were illustrated. The source code of high-level spin lock in Microsoft Windows was disassembled and analyzed. The mechanism to realize the WDM architecture with NT filter driver was proposed. A new exchange technology of dispatching routine's entry address of the driver object was advanced and verified.

Key words: filter driver; Small Computer System Interface (SCSI); driver object; completion routine

0 引言

计算机和计算机网络已经成为企业、政府和其他组织的重要信息载体和传输渠道。但是在享受计算机以及计算机网络所带来的方便性的同时,也出现了内部网络安全和外部网络安全问题。防火墙、入侵检测、内外网隔离以及其他对外部网络的访问控制系统,难以解决内部的网络安全问题。内部人员可以轻松地将计算机中的机密信息通过移动存储设备或者网络泄露出去,而且不会留下任何痕迹。

内部泄密事件的主要途径就是通过存储媒体、介质外流,因此对存储媒体、介质的防护有其重要的价值。存储媒体、介质防护包括软盘驱动器防护、可移动存储防护和 CD-ROM 防护。其中可移动存储器包括 USB 接口的所有可移动硬盘,包括 U 盘、可移动硬盘。外设接口包括 SCSI 接口、红外接口、PCMCIA 接口等。存储媒体、外设接口是内部泄密的主要途径,在不影响用户的正常行为的前提下,对这些设备进行合理的管理与控制,可以有效地防止内部人员的泄密行为[1]。

1 传统的介质设备控制方法

1.1 WDM 过滤驱动原理

WDM 模型使用了如图 1 和 2 的层次结构[2-3]。处于功能驱动程序之上的过滤器驱动程序称为上层过滤器;处于功能驱动程序之下的过滤器驱动程序(仍处于总线驱动程序之上)称为下层过滤器。虽然这两种驱动程序本身用于不同的目的,但创建这两种驱动程序的机制完全相同。实际上,创建过滤器驱动程序和创建任何其他 WDM 驱动程序一样,都有 DriverUnload 例程、AddDevice 例程、一组派遣函数,等等。上层过滤器驱动程序的用途是帮助支持这样的设备,这种设备的大多数方面都像其所属类的普通设备,但有一些附加功能。

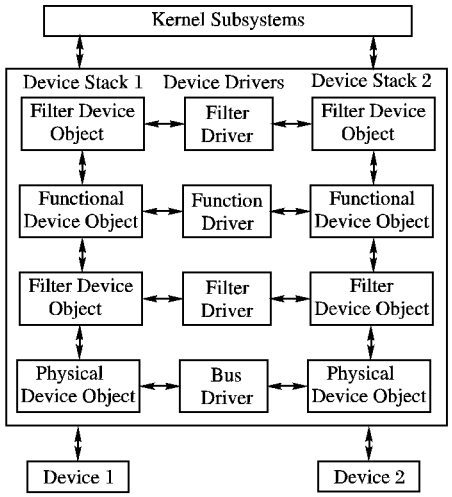

图 1 WDM 模型的层次结构

上层过滤器驱动程序的另一个用途是修正硬件或功能驱动程序中出现的 bug。下层过滤器驱动程序不能干涉功能驱动程序直接执行的正常操作,因为功能驱动程序可能通过 HAL 调用直接访问硬件来实现大部分实质的请求。而下层过滤器仅能看到经过它传递的 IRP,而看不到 HAL 调用。下层过滤器驱动程序可以用于 USB 设备的驱动程序堆栈中。

收稿日期:2009 - 02 - 18;**修回日期**:2009 - 04 - 03。
基金项目:国家自然科学基金资助项目(60473030);四川省科技厅科技攻关项目(05GG007-008)。
作者简介:马金鑫(1986 -),男,山西吕梁人,硕士研究生,主要研究方向:信息安全与网络安全; 袁丁(1967 -),男,四川宜宾人,教授,博士,主要研究方向:信息安全与网络安全、电子商务。

对于 USB 设备,其功能驱动程序把内部控制 IRP 作为 URB(USB 请求块)的容器,下层过滤器驱动程序可能会监视并修改这些 IRP。

当有新设备接入时,发向 PDO 的 AddDevice 例程就会首先经过上层过滤器驱动程序,最后也必然会经过下层过滤驱动程序,在 AddDevice 例程里过滤驱动的工作通常是创建一个控制设备,然后将其附着到 PDO 上,这样就可以实现 IRP 的过滤功能。

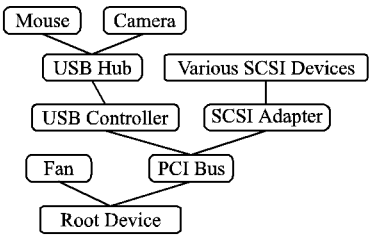

图 2　设备树结构

1.2　WDM 过滤驱动的加载机制

WDM 过滤驱动在 DriverEntery 中进行初始化(包括分配派遣例程,分配 AddDevice、DriverUnload)之后,接下来就是要在 AddDevice 例程中创建过滤驱动的控制设备,并将其附着在物理设备上。

注册表在 WDM 过滤驱动中发挥着重要的作用,其中包括三种键:硬件键、类键和服务键。概括地讲,硬件键包含单个设备的信息,类键涉及所有相同类型设备的共同信息,服务键包含驱动程序信息。设备的硬件键出现在注册表 local machine 分支的 \System\CurrentControlSet\Enum 子键上。通常你不能查看到该键的内部信息,系统只允许拥有系统管理员权限的用户访问该键。即只有内核模式程序和运行在系统管理员下的用户模式服务可以读写 Enum 键和其子键,但是即使是管理员也不应该直接修改这些键的内容。所有设备类的类键都出现在 HKLM\System\CurrentControlSet\Control\Class 键中。它们的键名是由 Microsoft 赋予的 GUID 值。对设备驱动程序重要的最后一个键是服务键,它指出驱动程序执行文件的位置,以及控制驱动程序装入的一些参数。服务键位于 HKLM\System\CurrentControlSet\Services 键中。

当 PnP 管理器遇到一个新设备时,它打开设备的硬件键和类键,然后按如下顺序装入驱动程序。

1)硬件键中指定的任何低层过滤器驱动程序。由于 LowerFilter 值的类型是 REG_MULTI_SZ,所以它可以指定多个驱动程序,其装入顺序由它们在串中的位置决定。

2)类键中指定的任何低层过滤器驱动程序。同样,它也可以指定多个驱动程序。

3)硬件键中 Service 值指定的驱动程序。

4)硬件键中指定的任何高层过滤器驱动程序,同样,它也可以指定多个驱动程序。

5)类键中指定的任何高层过滤器驱动程序,同样,它也可以指定多个驱动程序。

由于硬件键中的值不能随意地修改,所以只能修改类键中的值。类键下有以 GUID 为标志的所有计算机 PnP 设备,包括 Disk、USB、Keyboard 等外设和存储设备。只要在这些类键中添加 UpperFilters 或 LowerFilters 就可以为某一类设备添加过滤驱动。

在为某一类设备添加过滤驱动后,当有新介质设备插入到系统时,AddDevice 例程或先或后必然会经过过滤驱动程序,而在过滤驱动的 AddDevice 例程中,传进来的第二个参数就是所在设备栈的物理设备,所要做的工作就是创建控制设备并将其附着在物理设备上。

1.3　WDM 驱动机制的不足

WDM 过滤驱动技术对移动存储和外设都能实现很稳定可靠的监控,但是,用 WDM 做过滤驱动也有一些局限。1)过滤驱动必须对注册表进行操作,指定某一类设备的过滤驱动。这样,了解驱动知识的用户就可以通过人为地删除注册表来使过滤驱动无效。2)WDM 即插即用驱动不支持动态启动关闭的功能,使用起来不够灵活。这就使我们迫切需要寻求一种更完善,不易暴露目标的方法来对移动存储和外设进行监控。

2　设备控制实现

对于一般的外设来讲,当外设接入时只要将其 PnP 例程中的以 IRP_MN_START_DEVICE 副功能码为标志的例程返回无效,或者直接屏蔽其 SCSI 例程也可以达到禁用的目的。

对于移动存储设备,问题就变得复杂很多,如果只是用 UpperFilters 来对其 IRP_MJ_READ 或者 IRP_MJ_WRITE 进行控制,就会出现如图 3 的系统错误。

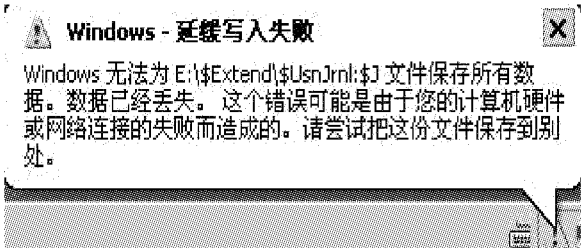

图 3　仿真实验结果(缓存写入失败)

SCSI 命令中为 1AH 的 OpCode 码是读设备参数的命令 MODESENCE,MODESENCE 有 6 字节和 10 字节版本,用来从一个设备读取模式参数列表。当读取的参数列表超过 255 字节时,可以使用命令的 10 字节版本。设备用 MODESENCE 从设备中读取数据,在内存中编辑,并用 MODESENCE 写回去[4-5]。因此,在过滤驱动中截获 SCSI 命令,为其设置完成例程,当命令请求完成返回时,修改其中的读写权限参数就可以对移动存储设备进行监控。其关键代码如下:

```
IoAcquireRemoveLock(&StorExtension->RemoveLock, Irp);
CurSrb = ExAllocatePoolWithTag(NonPagedPool,
  sizeof(SCSI_REQUEST_BLOCK), DISK_TAG_SRB);
RtlZeroMemory(CurSrb, SCSI_REQUEST_BLOCK_SIZE);
if(irpStack->MajorFunction == IRP_MJ_INTERNAL_DEVICE_
  CONTROL)
{
  //Get Current Scsi SRB, Analysis SCSI Command here!
  CurSrb = irpStack->Parameters.Scsi.Srb;
  cdb = (PCDB)CurSrb->Cdb;
  opCode = cdb->CDB6GENERIC.OperationCode;
  if(opCode == SCSIOP_MODE_SENSE && CurSrb->
    DataBuffer&& CurSrb->DataTransferLength >=
    sizeof(MODE_PARAMETER_HEADER))
  { modeData = (PMODE_PARAMETER_HEADER)CurSrb->
      DataBuffer;
    modeData->DeviceSpecificParameter| =
      MODE_DSP_WRITE_PROTECT;
  }
}
if( Irp->PendingReturned)
```

```
{
  IoMarkIrpPending( Irp) ;
}
IoReleaseRemoveLock( &StorExtension - > RemoveLock, Irp) ;
```

3 开拓新的思路

3.1 尝试 NT 过滤驱动的方法

如果用 NT 驱动来模拟 WDM 驱动，首先要模仿其 AddDevice 例程(因为 NT 驱动里没有 AddDevice 例程)，可以用 IoRegisterPlugPlayNotification 注册热插拔响应回调函数，当有新设备接入时，就会响应注册的回调函数，并传递设备的符号链接名，通过符号的链接名就可以获得相对应的设备对象。

但是在绑定设备时，又有新的问题出现，就是 IoAttachDeviceToDeviceStack 这个函数，通过反汇编可以得出这个函数的微软源码如下：

```
PDEVICE_OBJECT deviceObject;
PDEVOBJ_EXTENSION sourceExtension;
KIRQL irql;
sourceExtension = SourceDevice - > DeviceObjectExtension;
ExAcquireFastLock( &IopDatabaseLock, &irql ) ;
deviceObject = IoGetAttachedDevice( TargetDevice ) ;
ASSERT( sourceExtension - > AttachedTo == NULL ) ;
if ( deviceObject - > Flags & DO_DEVICE_INITIALIZING ||
  deviceObject - > DeviceObjectExtension - > ExtensionFlags&(
  DOE_UNLOAD_PENDING| DOE_DELETE_PENDING|
  DOE_REMOVE_PENDING|DOE_REMOVE_PROCESSED) )
deviceObject = (PDEVICE_OBJECT) NULL;
else
{
  //设置 deviceObject 的成员
  ...
}
ExReleaseFastLock( &IopDatabaseLock, irql ) ;
return deviceObject;
```

IoAttachDeviceToDeviceStack 的实际操作是把设备附在当前设备栈最顶端的那个设备上，即无法模拟像 WDM 过滤驱动那样可以保持在任意一设备的上、下端[6]。

3.2 内核例程 Hook 实现

当对内核源码有了一定的认识后，就算是内核源码一样可以对其进行很小心的改动，使其实现某些特定的功能。驱动程序被加载后，可以获得其在内存中的地址，也可以获得其派遣例程的入口地址，可以将入口地址替换成伪函数的地址，这样驱动接收到 IRP 后就会将 IRP 直接交给伪函数处理，当伪函数处理完之后，如果需要可以将 IRP 发到真正的例程处理函数的入口地址。具体实现如下：用 ObReferenceObjectByName 得到驱动对象的地址 drvierObejct(必须是已在运行的驱动)，driverObject - > [IRP_MJ_PNP] 就是此驱动处理 IRP_MJ_PNP 这个例程的入口函数的地址，再用 InterlockedExchange 将其替换成伪函数的入口地址，并保存原函数的地址，以备需要时恢复用[7]。

对于一般的 USB 设备来说，所有的 USB 都会与 USB 总线交互信息，只要过滤 USB 总线上的 IRP 就可以达到对设备启用和禁用的效果。USB 总线的驱动程序是 usbhub. sys 对应的名称是\driver\usbhub，而且是随系统启动而运行的，通过驱动名称得到驱动对象的地址，然后修改 IRP_MJ_PNP 的入口地址为伪函数的入口地址，如果是以副功能码 IRP_MN_START_DEVICE 为标志的 IRP，令其返回无失败就可以对设备禁用：return STATUS_INSUFFICIENT_RESOURCES。

而移动存储设备的读写控制不能只用控制 PnP 的方法来实现了，还是参照第 2 章里的方法，用修改权限参数的方法来对移动存储设备的读写进行控制。这个修改过程要在 SCSI 例程的完成例程里来实现，所以就要对完成例程进行过滤。

完成例程的入口函数替换方法与普通 IRP 入口函数替换方法有所不同，如要对 SCSI 例程的完成例程入口函数地址进行替换，要在 SCSI 入口函数中进行上下文保存工作，并且给 SCSI 指定完成例程的函数地址，关键代码如下：

```
Ctx = (HP_CONTEXT * ) ExAllocatePoolWithTag(
  NonPagedPool, sizeof( HP_CONTEXT) , DISK_TAG_SRB) ;
ctx - > oldIocomplete = irpStack - > CompletionRoutine;
ctx - > oldCtx = irpStack - > Context;
irpStack - > CompletionRoutine = HookedIrpRoutine;
irpStack - > Context = ctx;
if ( ( irpStack - > Control & SL_INVOKE_ON_SUCCESS) ==
  SL_INVOKE_ON_SUCCESS)
  ctx - > bShouldInvolve = TRUE;
else
  ctx - > bShouldInvolve = FALSE;
irpStack - > Control | = SL_INVOKE_ON_SUCCESS;
```

目前主流的移动储设备(如 U 盘，移动硬盘等)都是以 USBSTOR. sys 为功能驱动，经实验表明：只有对 USBSTOR. sys 的 SCSI 命令进行监控才能对截获读设备参数的请求。在移动设备接入计算机后，USBSTOR. sys 才会开始运行。在 USB 总线的 IRP_MJ_PNP 中进行判断，如果有设备接入，并且判断是否为移动存储设备，如果是则动态地对其进行的入口函数地址替换。

在仿真实验中发现：如果像 WDM 的 SCSI 完成例程中一样去修改读写权限参数 ModeData 时并不能成功，经过大量的调试后，发现在完成例程中的 SCSI 请求块 Srb 出现了地址偏移，不能正确得到原来 Srb 的地址，因此需要在 SCSI 例程中保存 Srb 的地址，到完成例程中就可以直接对其进行修改操作。

3.3 实验结果与结论

经测试表明，这种内核驱动技术稳定，兼容性强。并且这种方法未对用户的其他正常行为造成任何影响，其中对 U 盘写控制的测试结果如图 4 所示，可以达到预期的功能目标；而且，这种方式的驱动不需要对注册表进行任何操作，只需要动态地创建服务和删除服务就可以加载和卸载驱动，不会给非法用户任何漏洞破坏防护机制。显然，这种方法要比 WDM 过滤驱动的方法更加完善和安全。

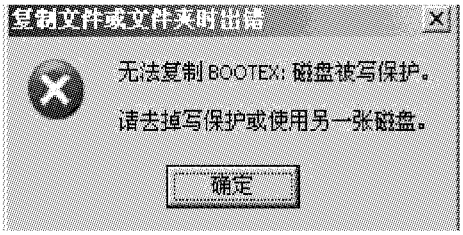

图4 实验结果(U 盘写控制)

基于 WDM 过滤驱动模式的方法不支持动态的开启和关闭。而这种方法以系统服务的形式动态地加载、卸载，不需要随系统启动而启动，并且可以在适当的时候停止它，大大地增加了上层软件的灵活性。基于WDM过滤驱动模式的方法需

(下转第 2187 页)

ence on Sensor, Mesh and Ad Hoc Communications and Networks. New York: ACM Press, 2006: 109 - 118.

[8] NGAI E C H, LYU M R, LIU J C. A real-time communication framework for wireless sensor-actuator networks [EB/OL]. [2009 - 01 - 10]. http://user.it.uu.se/~eding810/paper/aero06-Oct-25.pdf.

[9] 吴震东. 无线传感器网络多节点协同相关问题研究[D]. 杭州: 浙江大学, 2007.

[10] YUAN H D, MA H D, LIAO H Y. Coordination mechanism in wireless sensor and actor networks [C]// Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences. Washington, DC: IEEE Computer Society, 2006: 627 - 634.

[11] GUNGOR V C, AKAN ÖZGUR B, AKYILDIZ I F. A real-time and reliable transport $(RT)^2$ protocol for wireless sensor and actor networks [J]. IEEE Transactions on Networking, 2008, 16(2): 359 - 370.

[12] CHATZIGIANNAKIS I, KINALIS A, NIKOLETSEAS S. Priority based adaptive coordination of wireless sensors and actors[C]// Proceedings of the 2nd ACM International Workshop on Quality of Service & Security for Wireless and Mobile Networks. New York: ACM Press, 2006: 37 - 44.

[13] AKAN B O, AKYILDIZ I F. Event-to-sink reliable transport for wireless sensor networks [J]. IEEE Transactions on Networking, 2005, 13(5): 1003 - 1016.

[14] BOUKERCHE A, ARAUJO R B, VILLAS L. A novel QoS based routing protocol for wireless actor and sensor networks [C]// Proceedings of the 2007 IEEE Global Telecommunications Conference. Washington, DC: IEEE Computer Society, 2007: 4931 - 4935.

[15] CARDEI M, YANG S H, WU J. Algorithms for fault-tolerant topology in heterogeneous wireless sensor networks [J]. IEEE Transactions on Parallel and Distributed Systems, 2008, 19(3): 545 - 558.

[16] OZAKI K, WATANABE K, ITAYA S, *et al.* A fault-tolerant model of wireless sensor-actor network [C]// Proceedings of the 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing. Los Alamitos: IEEE Computer Society, 2006: 185 - 193.

[17] VEDANTHAM R, ZHUANG Z Y, SIVAKUMAR R. Mutual exclusion in wireless sensor and actor networks [C]// Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks. Reston: IEEE Computer Society, 2006: 346 - 355.

[18] HOLLAND G, VAIDYA N, Analysis of TCP performance over mobile Ad Hoc networks [J]. Wireless Networks, 2002, 8(2): 275 - 288.

[19] SUNDARESAN K, ANANTHARAMAN V, HSIEH R V, *et al.* ATP: A reliable transport protocol for Ad Hoc networks [J]. IEEE Transactions on Mobile Computing, 2005, 4(6): 588 - 603.

[20] ABBASI A A, AKKAYA K, YOUNIS M. A distributed connectivity restoration algorithm in wireless sensor and actor networks [C]// Proceedings of the 32nd IEEE Conference on Local Computer Networks. Washington DC: IEEE Computer Society, 2007: 496 - 503.

[21] LIU X, XIAO L, KRELING A, *et al.* Optimizing overlay topology by reducing cut vertices [C]// Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video. New York: ACM Press, 2006: 17 - 17.

[22] OZAKI K, HAYASHIBARA N, TAKIZAWA M. Coordination protocols of multiple actuator nodes in a multi-actuator/multi-sensor model [C]// Proceedings of the 21th International Conference on Advanced Information Networking and Applications Workshops. Washington, DC: IEEE Computer Society, 2007: 62 - 67.

[23] OZAKI K, HAYASHIBARA N, ENOKIDO T, *et al.* Fault-tolerant semi-passive coordination protocol for a multi-actuator/multi-sensor (MAMS) model [C]// Proceedings of the 2nd International Conference on Availability, Reliability and Security. Washington, DC: IEEE Computer Society, 2007: 506 - 516.

[24] BOUKERCHE A, ARAUJO R B, SILVA F H S, *et al.* Wireless sensor and actor networks context interpretation for the emergency preparedness class of applications [J]. Computer Communications, 2007, 30(13): 2593 - 2602.

[25] MUHL G, ULBRICH A, RITTER H. Content evolution driven data propagation in wireless sensor networks [EB/OL]. [2008 - 11 - 20]. http://doc.tm.uka.de/tr/TM-2004-5.pdf.

[26] BOUKERCHE A, MARTIROSYAN A. An efficient algorithm for preserving events temporal relationships in wireless sensor actor networks [C]// Proceedings of the 32nd IEEE Conference on Local Computer Networks. Washington, DC: IEEE Computer Society, 2007: 771 - 780.

[27] BOUKERCH A, ARAUJO R B, SILVA F H S. An efficient event ordering algorithm that extends the lifetime of wireless actor and sensor networks [J]. Performance Evaluation, 2007, 64(5): 480 - 494.

[28] VEDANTHAM R, ZHUANG Z Y, SIVAKUMAR R. Hazard avoidance in wireless sensor and actor networks [J]. Computer Communications, 2006, 29(13/14): 2578 - 2598.

(上接第 2148 页)

要操作注册表来指定其控制设备在设备栈中的具体位置,如果恶意操作把注册表中的这些信息删除,那么监控就如同虚设。而这种方法是基于“HOOK”的原理对驱动对象的例程入口地址进行锁定修改,完全不需要对注册表有任何操作,具有非常好的隐藏性,大大地增加了监控不被恶意关闭的安全性。

4　结语

本文针对 WDM 过滤驱动实现移动介质和存储设备监控方法所表现出的不足进行了改进,以另一种思路来实现它,不仅可以完成移动介质和存储设备监控的功能,而且弥补了 WDM 过滤驱动的不足。通过对仿真实验的测试表明,并没有对系统内核造成任何负面的影响。本文进一步的工作即是研究适合 Vista 操作系统及以后的“长角系统”的 KMDF 框架[9]开发模式来对设备进行控制,实现向上兼容。

参考文献:

[1] 中国软件与技术服务股份有限公司. 中软防水墙系统 WaterBox™ 7.2R5 [Z]. 北京: 中国软件与技术服务股份有限公司, 2008.

[2] ONEY W. Programming the Microsoft Windows Driver Model [M]. [S.l.]: Microsoft Press, 2003.

[3] ORWICK P, SMITH G. Developing Drivers with the Microsoft Windows Driver Foundation [M]. [S.l.]: Microsoft Press, 2007.

[4] SCHMIDT F. The SCSI Bus & IDE Interface Protocols, Applications & Programming [M]. Upper Saddle River: Addison Wesley, 2001.

[5] HYDE J. USB Design by Example[M]. 2nd ed. [S.l.]: Intel University Press, 2002.

[6] HOGLUND. IRP hooking and Device Chains[EB/OL]. [2009 - 02 - 01]. http://www.rootkit.com/newsread.php?newsid=846.

[7] 楚狂人. 天书夜读[M]. 北京: 电子工业出版社, 2007.

[8] CARDMAGIC. NSI ModuleHook: Hiding Port Under Windows Vista [EB/OL]. [2009 - 02 - 01]. http://www.rootkit.com/newsread.php?newsid=735.

[9] MICROSOFT. Architecture of the Kernel - Mode Driver Framework [EB/OL]. [2009 - 02 - 11]. http://www.microsoft.com/whdc/driver/driver/wdf/KMDF.