基于 Windows 文件系统过滤驱动文件透明加密的研究*

易 丹

（武汉交通职业学院，湖北 武汉 430065）

摘 要：文件作为计算机系统最重要的资源，对其加密保护是实现信息安全的主要途径。文章综合比较分析几种常见的文件加密方式后，认为文件系统过滤驱动加密在性能、成本和安全性上达到了平衡。本文以此为研究对象，通过对 Windows 内核分析，详细阐述过滤驱动加密的原理和透明加密的过程。

关键词：透明加密；文件系统；过滤驱动；信息安全

中图分类号：TP309 **文献标识码**：A **文章编号**：1672－9846(2013)01－0080－05

1 文件加密发展现状

随着计算机与网络应用的普及，人们之间的信息交互越来越方便，但由此产生的信息安全问题也日益突出。为了保证信息的安全，我们需要对存储在计算机系统上的文件信息进行加密。目前常见的文件加密方式有四种：用户层文件加密、系统调用层加密（APIHOOK）、文件系统过滤驱动加密和硬件加密。本文综合比较四种加密方式的原理及效率（见表 1），发现文件系统过滤驱动加密达到了在性能、成本和安全性上的平衡。接下来，本文重点阐述基于 Windows 文件系统过滤驱动的文件透明加密技术。

表 1 四种文件加密方式比较

	加密原理	加密效率
用户层加密	用户使用特定的加密工具，自己选择密钥实现加解密	实现起来简单，但加密过程繁琐，用户参与步骤过多，容易被破解
系统调用层加密	利用 Windows 的钩子技术，监控应用程序对文件的操作过程，读写文件前分别进行解密和加密	工作在 Windows 的应用层容易实现，但可靠性差，速度慢，当软件升级时可能需要根据版本重新开发。
文件系统过滤驱动加密	拦截发往文件系统驱动的文件读写请求，在内核级进行加解密	工作在 Windows 的内核层，安全性高，不容易被攻击，但开发难度大
硬件加密	在存储设备和主机之间加一个硬件控制模块，它可以拦截发给磁盘的读写请求，由硬件实现加解密	此方法成本高、灵活性差，加密算法一旦确定就不能更改

2 文件系统过滤驱动加密的工作系统——Windows 操作系统

文件系统过滤驱动工作在 Windows 内核模式下。因此，我们需要对 Windows 操作系统进行比较深入的学习和了解。

2.1 Windows 体系结构组成

如图 1 所示，在 Windows 系统中，应用软件是“看不见”操作系统内核的。应用软件需要通过

*收稿日期：2012－12－17

作者简介：易 丹（1975－），女，湖北武汉人，武汉交通职业学院教师，主要从事计算机网络技术研究。

API 函数调用获取操作系统的支持,才能访问内核级的代码。

Windows 内核模块由以下几部分组成[1]:(1)硬件抽象层(HAL)。HAL 为 Windows 其它组件提供硬件平台低层接口。它隐藏了与硬件相关的细节,比如,中断控制器,I/O 接口等与机器相关的功能。Windows 内部组件可以调用 HAL 的例程来实现可移植性。(2)设备驱动层。设备驱动程序是可以动态加载卸载的内核模块,它和 I/O 管理协同工作,在用户和真实的物理设备之间建立连接,保证用户请求能够由相应的硬件设备顺利完成。(3)Windows kernel。Windows 内核为执行体提供了操作系统的工作机制和底层原语。提供了执行体组件需要使用的(Ntoskrnl.exe 中的)一组函数,例如,中断和异常分发,线程调度和同步等。使得执行体可以实现更高层次的功能。(4)Windows 执行体。Windows 执行体包含了操作系统可以提供的服务操作,包括系统服务函数(system service),调用设备驱动器的函数,以及内存管理、对象管理等系统组件的管理。

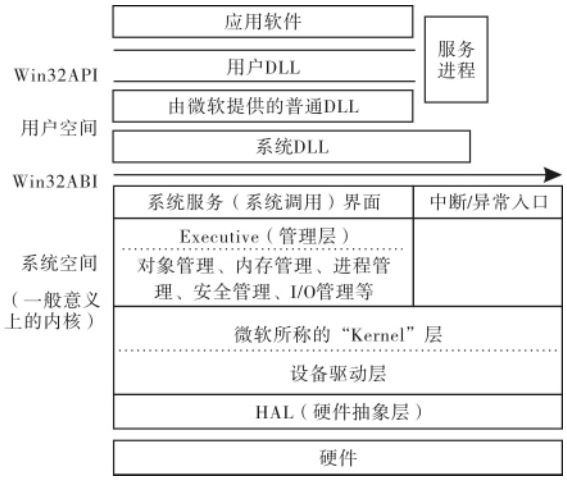

图 1　windows 系统结构

2.2　I/O 管理和内存管理

Windows 的文件操作主要涉及到 I/O 管理和内存管理,这两个内核组件提供了由过滤驱动实现文件加解密的理论基础。[2]

2.2.1　I/O 管理

WindowsI/O 系统的作用是为上层用户程序提供一个有关设备的抽象,用特定的数据结构实现与底层物理或者虚拟设备的数据传递,将用户的请求分发给相应的设备驱动程序处理。此设备抽象具有以下特性:①统一的,跨设备的安全性和命名机制;②高性能的,异步的,基于数据包的 I/O;③提供支持用高级语言编写驱动的服务;④动态的加载和卸载驱动程序,允许加入新的驱动程序;⑤支持即插即用和电源管理;⑥支持多个安装的文件系统,包括 FAT、CD－ROM、UDF、NTFS 等。

I/O 管理器作为 I/O 系统的核心,监视着 windowsI/O 子系统,它能够把来至用户的请求构建成一个 IRP(即 I/O 请求包),把该 IRP 传递给目标驱动程序;驱动程序根据 IRP 中的相应参数将这些请求转变成与硬件相关的操作。文件系统过滤驱动就位于 I/O 管理器和文件系统驱动之间。

2.2.2　内存管理

支持多进程的现代操作系统一般都会采用基于页面映射的"虚拟内存机制",内存管理器提供了一组系统服务来完成并发进程下的各种任务,如下所示:分配和释放虚拟内存;在进程之间共享内存;将文件映射到内存;将虚拟页面刷新到磁盘上;获得一定范围内虚拟页面的信息;改变虚拟页面的保护属性;将虚拟页面锁在内存中。

内存管理服务允许其调用者提供一个进程句柄,来指明哪个特定进程的虚拟内存需要进行操作。这些服务中的大多数是通过 Windows API 暴露给用户的,Windows API 有三组函数可用来管理应用程序中的内存:页面粒度的虚拟内存函数、内存映射文件函数、堆函数。

另外,为了缓解文件读写过程中反复读写磁盘造成的高负荷,引入了缓存管理,缓存管理能够实现"预读"和"延迟写"的功能。文件系统在接到读写文件的请求时,会通过缓存管理器将文件读写到缓存缓冲区中,当一段时间内出现多次对该文件的读写操作时,就避免了多次读写磁盘的麻烦。

由以上可以看出,来自于用户的文件请求,经过 I/O 系统处理后,最终由文件系统在内存管理器的配合下直接完成或者下发给底层驱动完成。

3　文件系统过滤驱动透明加密实现

3.1　Windows 文件系统过滤驱动透明加解密原理

文件系统驱动是一个虚拟设备驱动,它与内存管理协同工作实现用户在计算机系统中的信息管理与操作。

利用文件系统过滤驱动实现文件透明加解密

就是在 I/O 管理器和文件系统驱动之间加一层过滤驱动。本来发给文件系统的来至于用户的所有请求，都会先被过滤驱动截取，只要在过滤驱动中替换或者扩展原始请求的功能，就能够实现文件的加解密，并且这一过程是在内核中自动完成的，不会影响用户的任何体验，所以称作透明加密。图 2 展示了过滤驱动在设备栈中的位置。

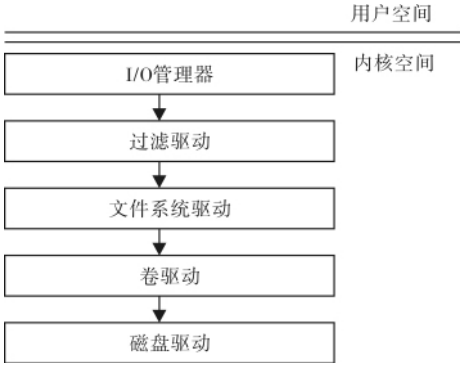

图 2　过滤驱动在设备堆栈中的位置

3.2　文件透明加解密实现

Windows 驱动程序工作在内核模式下，属于直接和硬件打交道的模块，主要由驱动程序入口函数、派遣例程、驱动程序卸载函数三部分组成。[3]

驱动程序入口函数 DriverEntry(IN DriverObject, IN RegistryPath)，驱动程序派遣函数 DispatchRutine(IN DeviceObject, IN irp)，驱动卸载函数 XXXUnload(IN DriverObject)。

其中，参数 DriverObject 是每一个驱动程序的唯一驱动对象，它是在驱动加载的时候，被内核中的对象管理程序所创建的；参数 RegistryPath 是一个 Unicode(宽字节)字符串，指向此驱动负责的注册表；参数 DeviceObject 是设备对象，每个驱动程序会有一个或多个设备对象，设备对象是由程序员自己创建的，其中保存真实物理设备的特征和设备的状态信息；参数 irp 就是 I/O 管理器送过来的请求包，驱动程序会根据包的内容确定调用哪一个派遣例程，常见的有读(read)，写(write)，打开(open)，关闭(close)等。

当驱动程序被卸载的时候，由 I/O 管理器负责调用驱动卸载函数，遍历系统中所有的此类设备对象，然后删除这些设备对象。

(1)配置过滤驱动

将创建的过滤驱动的设备对象挂接到文件系统驱动所在的驱动设备栈。通过函数 IoAttachDeviceToDeviceStack(IN SourceDevice, IN TargetDevice)来实现。接下来对过滤驱动进行配置，拦截相应的文件操作请求，代码如下：

```
if(irpsp->MajorFunction != IRP_MJ_CREATE &&
    irpsp->MajorFunction != IRP_MJ_CLOSE &&
    irpsp->MajorFunction != IRP_MJ_READ &&
    irpsp->MajorFunction != IRP_MJ_WRITE &&
    ……
  ……
  )
        return SF_IRP_PASS;
```

以上代码设置了需要过滤的 IRP，如果是与加解密无关的操作，则直接跳过过滤驱动，正常下发到底层驱动。

预处理：

```
if(Irpsp->MajorFunction==IRP_MJ_READ&&
(irp->Flags&(IRP_NOCACHE|IRP_PAGING_IO|IRP_SYNCHRONOUS_PAGING_IO)))
{
EncryptReadPre(irp, Irpsp);
}
if(Irpsp->MajorFunction==IRP_MJ_WRITE&&
(irp->Flags&(IRP_NOCACHE|IRP_PAGING_IO|IRP_SYNCHRONOUS_PAGING_IO)))
{
EncryptWritePre(irp, Irpsp, context)
}
```

后处理：

```
if(Irpsp->MajorFunction==IRP_MJ_READ)
{
EncryptReadPost(irp, irpsp);
}
else if(Irpsp->MajorFunction==IRP_MJ_WRITE)
{
EncryptWritePost(irp, Irpsp, context);
}
```

其中，写时加密是在写到磁盘前进行的，即在预处理中实现；读时解密是在从磁盘读到内存后进行的，即在后处理中实现。

(2)加解密实现过程

加解密时需要知道将要被处理的数据在内存中的具体位置，我们可以通过了解 IO 的读写方式来获取这

一信息。I/O 操作一般有三种方式：缓冲方式、直接方式及其他方式。文件读写一般采用后两种方式[4]。

直接方式使用 IRP－＞MdlAddress 来传递缓冲区。因为来自于用户的请求，其指针指向的缓冲区都在用户空间中，工作于内核模式下的驱动程序虽然可以正常使用该指针，但是，如果在完成的时侯，需要其他进程的线程协同处理，那么就形成了进程切换，这些缓冲区的指针就失效了。引入 MDL 的作用是把由用户空间映射的一段物理地址再用内核（系统）空间映射一次，即同一段物理地址既属于用户空间又属于内核空间，因为所有的进程共享内核空间，所以进程切换时，指向内核的同一指针的内容是相同的。

其他方式使用 IRP－＞UserBuffer 来传递缓冲区。直接把用户空间的指针传递到内核，不做任何其他处理。前提是整个请求只在当前线程的上下文中处理，没有进行切换。

在 WindowsNT 及 XP 系统中，应用程序的读写文件请求一般遵循如图 3 的流程，应该根据此流程编写加解密代码。

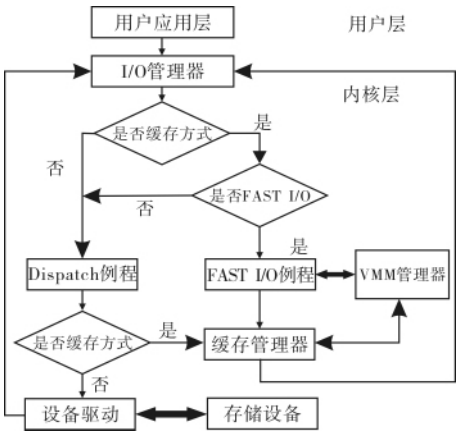

图 3　应用程序的文件读写请求过程

加密过程：即函数 EncryptWritePre（irp，Irpsp，context）的处理：

```
BOOLEAN EncryptWritePre (PIRP irp, PIO__STACK_LOCATION irpsp, PVOID* context)
{……
……
//在这里得到缓冲进行加密。要注意的是写请求的缓冲区是不可以直接改写的。必须重新分配。
ASSERT(irp－＞UserBuffer！＝NULL||irp－＞MdlAddress！＝NULL);
if(irp－＞MdlAddress！＝NULL)
{//产生一个新的 MDL 结构，用于加密处理的缓冲区
buffer＝MmGetSystemAddressForMdlSafe(irp－＞MdlAddress,
NormalPagePriority);
Encrypt_mdl＝EncryptMdlMemoryAlloc(length);
Encrypt __buffer ＝ MmGetSystemAddressForMdlSafe(Encrypt_mdl,
NormalPagePriority);
}
else
{//如果是采用其他方式读写，则同上进行相似处理，分配新的 UserBuffer
buffer＝irp－＞UserBuffer;
Encrypt__buffer ＝ ExAllocatePoolWithTag (NonPagedPool,length,CF_MEM_TAG);
}
RtlCopyMemory(Encrypt_buffer,buffer,length);
//加密处理
……
……
}
```

解密过程：即函数 EncryptReadPost(irp，Irpsp)的处理

```
//读请求时，对从磁盘读进来的数据传到用户空间之前进行解密。
BOOLEAN EncryptReadPost(PIRPirp, PIO_STACK__LOCATION Irpsp)
{
//先获取缓冲区再解密。
……
…….
ASSERT(irp－＞UserBuffer！＝NULL ||irp－＞MdlAddress！＝NULL);
if(irp－＞MdlAddress！＝NULL)
Encrypt_buffer＝MmGetSystemAddressForMdlSafe(irp－＞MdlAddress,
NormalPagePriority);
else
Encrypt_buffer＝irp－＞UserBuffer;
//相应的解密处理
……
……
//提示：假如加密只是对原文件的异或处理，那么解密也只进行异或处理如下
```

```
//for(i=0;i<length;++i)
//Encrypt_buffer[i]^=0X77;
}
```

4　总结

基于文件系统过滤驱动开发的文件透明加密系统与 windows 内核紧密关联，开发难度大，容易出错。但是正因为它是在 Windows 内核实现的，所以，安全机制完善，不容易被攻击和破解，并且得益于驱动程序的动态加载卸载，开发灵活性高，利用过滤驱动进行加解密在安全领域具有很高的实用价值。

参考文献：

[1]毛德操. Windows 内核情景分析[M]. 北京：电子工业出版社，2009：1099－1175.

[2]Mark E. Russinovich. Microsoft Windows Internals[M]. 潘爱明，译. 北京：电子工业出版社，2010：537－555.

[3]张帆，史彩成. Windows 驱动开发技术详解[M]. 北京：电子工业出版社，2008：87－117.

[4]谭文，杨潇. Windows 内核安全编程[M]. 北京：电子工业出版社，2009：236－262.

构建综合交通运输服务体系

据《中国交通报》2012 年 12 月 17 日报道，日前，国务院印发《服务业发展“十二五”规划》（简称《规划》），提出加快发展交通运输业、现代物流业等生产性服务业，拓展海洋服务业领域。

1. 交通运输网更趋完善。《规划》提出：“十二五”时期，交通运输基础设施网络更趋完善，创新能力不断增强，管理能力不断提高，服务质量和效率不断提升，构建网络设施配套衔接、技术装备先进适用、运输服务安全高效的综合交通运输服务体系。

为实现上述目标，要加快国家高速公路网剩余路段、“瓶颈”路段建设，加强路网运行监测和交通出行信息服务；继续推进农村公路建设，推进城乡客运一体化。加快发展内河水运，推进重庆长江上游和武汉长江中游航运中心建设；推进沿海港口协调有序发展，加快推进上海国际航运中心、天津北方国际航运中心和大连东北亚国际航运中心建设；形成具有国际竞争力的海运服务体系。实施公共交通优先发展战略，大力发展农村客运和农村物流，推进综合运输大通道和综合交通枢纽建设。

同时，建设国家快速铁路网，强化重载货运网。建立通达通畅的国内国际航线网络，加快发展通用航空。加快邮政服务业发展，提高服务能力和水平。

2. 初步建立现代物流体系。“十二五”时期，物流业信息化、智能化和标准化水平将明显提高，重点行业物流服务能力显著增强，初步建立社会化、专业化、信息化的现代物流体系。

《规划》提出，大力发展第三方物流，建设覆盖全国的物流通道网络，加快推进城市配送体系建设。鼓励物流业与制造业联动发展，拓展邮政物流，推动快递与电子商务、制造业协同发展，培育一批具有国际竞争力的现代物流企业。

3. 增强国际海运竞争力。《规划》提出，大力发展海洋运输业，壮大海运船队，增强国际海运竞争力，提升能源、原材料等战略物资运输保障能力。不断提高航海保障、海上救生和救助服务水平。发展海峡、岛屿间客滚运输和海上旅游、游艇经济，在有条件的港口发展集娱乐、休闲、餐饮、购物于一体的邮轮经济。

4. 完善珠三角与港澳跨界交通体系。在扩大开放方面，《规划》提出，引导外商投资发展交通运输、现代物流等生产性服务业；分类指导，积极引导运输等有比较优势的企业对外投资。

完善珠三角地区与港澳跨界交通运输体系，建立跨界交通监管合作机制，加强口岸综合配套服务功能和区域物流信息平台建设。其中，粤港澳服务业合作重大项目主要包括：港珠澳大桥、广深港客运专线、港深西部快速轨道线、莲塘/香园围口岸、深圳前海开发、广州南沙新区开发和珠海横琴新区开发。

海峡两岸服务业合作将进一步推进。合理调控运力，适时增加直航航点及定期航班班次。拓展两岸邮政合作领域。推进两岸冷链物流产业合作试点项目建设，共同提升两岸物流业的国际竞争力。