文件透明加密技术研究

王全民[1,2],周 清[2],刘宇明[2],朱二夫[2]
(1.天津大学 计算机学院,天津 300072;
2.北京工业大学 计算机学院,北京 100124)

摘 要:文件作为计算机系统最重要的资源,加密是保护它的最重要的方法之一。目前的加密方法很多,但是大部分都是用户态的密码加密,操作繁琐且加密效果不理想。文中介绍了两种区别于前者的透明加密方法:钩子透明加密和过滤驱动透明加密,并对这两种方法进行了比较。对目前的热点技术——过滤驱动加密技术进行了比较深入详细的研究,该方法由于是在内核模式下运行,较之于用户态下的加密具有高效、安全、灵活等特点,本质上就是文件系统功能的扩展。

关键词:透明加密;APIHOOK;过滤驱动
中图分类号:TP309.7 **文献标识码**:A **文章编号**:1673-629X(2010)03-0147-04

Research on File System Transparent Encryption Techniques

WANG Quan-min[1,2],ZHOU Qing[2],LIU Yu-ming[2],ZHU Er-fu[2]
(1.College of Computer Science,Tianjin University,Tianjin 300072,China;
2.College of Computer Science,Beijing University of Technology,Beijing 100124,China)

Abstract:File system is the most important resource in computer system,and encryption is a good method to protect them. At present,the majority encrypt methods are implemented on user mode using keys,but unfortunately the effect is not very well. This paper will introduce two kinds of transparent encryption methods:API hook and file system filter driver,and meanwhile comparing this two methods, especially focus on the latter one- file system filter driver,for it is run on kernel mode, contrast with user mode encryption,it is more efficient, secure,and flexible, in essence it is an extension of the file system.

Key words:transparent encryption;API HOOK;filter driver

1 文件透明加密技术简介

文件透明加密技术是近些年针对企业文件保密需求应运而生的一种文件加密技术,区别于常见的文件密码加密。所谓透明,是指对用户而言加解密过程不会被觉察,当用户打开或编辑受保护文件时,系统将自动对未加密的文件进行加密,对已加密的文件解密。文件在硬盘上以密文形式存储,在内存中则为明文,一旦改变使用环境,由于无法获得自动解密服务而无法打开,从而达到保护文件内容的目的。

透明加解密技术是与操作系统紧密结合的一种技术,Windows允许程序设计人员在内核和用户两个级别操作文件,因此,加密进程就可以在这两个层次截获文件读写操作,嵌入自己的加密算法进行加解密,通常应用级别的截获采用API HOOK(俗称钩子)技术,称为钩子透明加密,内核级采用文件过滤驱动,称驱动加密。两者的加密层次如图1所示。

下面分别详细讨论这两种加密方法。

2 钩子透明加密技术

早期的透明加密的技术手段大多是通过API钩子技术进行,其工作方式为静态加密和重定向,基本思想为钩子程序拦截到受保护文件的打开操作,先将已加密的文件拷贝到一个临时目录中,然后告诉驱动程序把这个文件隐藏起来,然后解密这个文件,并将这个临时文件返回给打开文件的进程,这样打开的就是磁盘上的一个明文文件,用户程序可以进行正常的处理,这里文件是进行了整体的拷贝并解密;在文件关闭时,钩子拦截到以后,将那个明文的临时文件加密,然后再拷贝回来覆盖掉原文件。

简单的讲,就是打开文件时,将加密文件的副本拷贝到隐蔽位置,然后把这个文件解密、打开,保存时再把临时文件加密后覆盖原文件。这种加密实质上是通过临时文件来实现的,为了确保明文和密文之间的一

收稿日期:2009-06-03;**修回日期**:2009-10-20
作者简介:王全民(1963-),男,山西人,副教授,博士,研究方向为信息安全。

致性,必须在每次用户存盘或者系统自动存盘的时候,对整个文件进行一次整体加密和并复制到用户文件原来存储的位置,因此效率较低,且由于在计算机上保存了完整的明文文件,只要跟踪到临时文件所在位置即可能泄密,此外,文件在打开和存储过程中需要多次复制,效率损失很大,Hook API 的方式对于一些程序不能够完美的支持,兼容性有待考证。

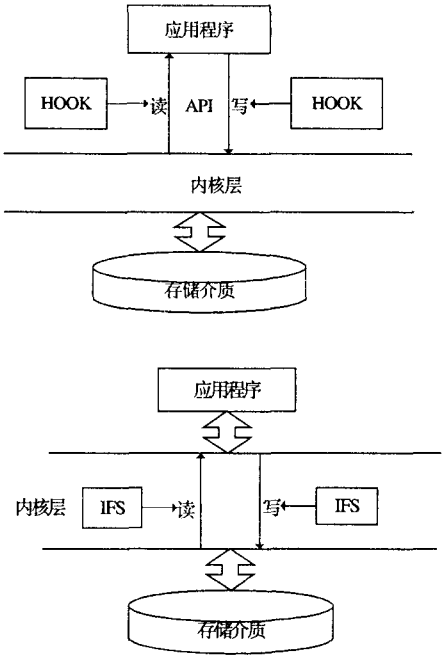

图 1 钩子透明加密和驱动透明加密的截获文件操作的层次

由于文件映射的存在,在应用层进行动态加密比较困难,虽然原理上有一定的可行性,但是其加密效率、兼容性、可移植性、稳定性较之于内核层的过滤驱动加密要差很多,好处在于开发难度较小,网络操作能力强。

3 过滤驱动透明加密技术

3.1 驱动技术简介

过滤驱动加密技术是基于 Windows 的文件系统过滤驱动(IFS)技术,工作在 Windows 的内核层。它是目前炙手可热的一门技术,其特点是技术门槛较高,需要深入理解 Windows 系统内核,开发过程中稍有不慎就会破坏系统内核,因此核心技术仅被少数几家实力雄厚的公司所掌握。

文件过滤驱动是把文件作为一种设备来处理的一种虚拟驱动,Windows NT 系统内核采用堆栈式可扩展驱动模型[1],原则上可以在任何一个层次上加载自己的过滤驱动,但是加载的层次不同,开发的难度和应用价值也不同,现在的技术主要还是加载在文件系统驱动的上层,这样就可以充分利用并扩充文件系统的现有功能[2],该技术在病毒实时监控与防护、数据备份与还原以及文件访问控制等领域都有着非常广泛的应用。

过滤驱动中几种非常重要的概念包括驱动对象、设备对象、设备堆栈、I/O 堆栈,驱动对象标示驱动程序,设备对象记录加载的设备,一个驱动对象可以有多个设备对象构成设备对象链表,设备堆栈用于记录所有驱动程序的设备,I/O 栈用来记录穿越设备堆栈时的操作属性,操作的对象为 IRP(I/O Request Package),IRP 包由系统的某个组件创建,类似于 Windows 应用程序的"消息"概念,要处理某种数据只需要把 IRP 包传送到相应驱动的相应派遣函数中[3]。

3.2 加密的基本思想

在 Windows NT 内核操作系统中,应用程序的一次数据请求的过程如图 2 所示[4]:只有在无缓存或者 FAST I/O 不可用的情况下才在硬盘上读取数据,因此只用拦截 IRP—>Flags 为 IRP_NOCACHE(表示 I/O 请求从存储的媒介而不是高速缓存中读取数据)、IRP_PAGING_IO(表示此时执行内存页的 I/O 操作)和 IRP_SYNCHRONOUS_PAGING_IO(表示内存页

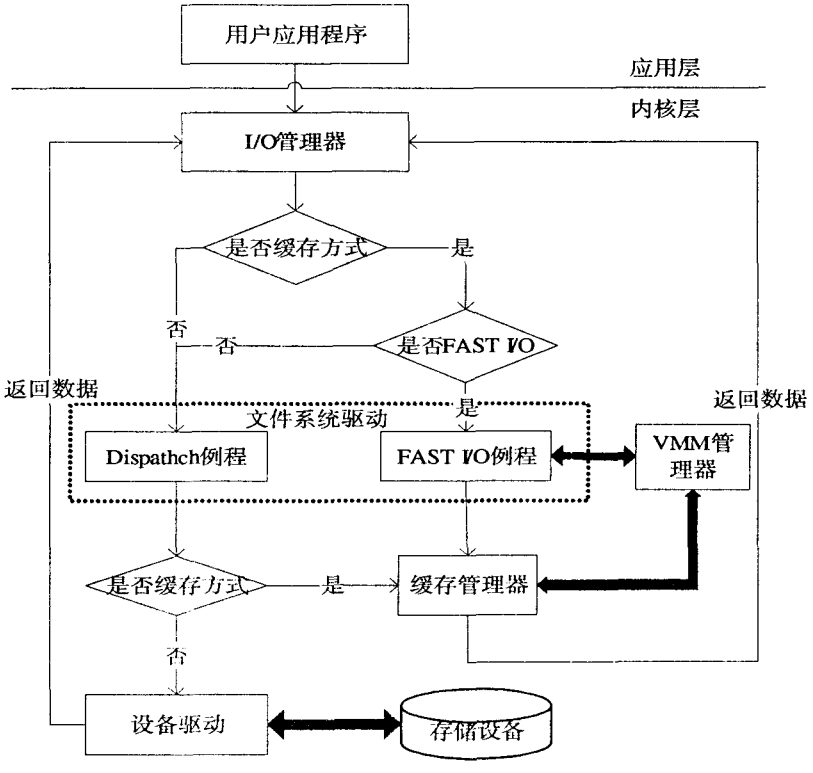

图 2 应用程序的数据请求过程

需要同步更新,此标志也是由内存管理器使用)的数据包,对于写数据的 IRP 包,在其分发例程中嵌入加密算法完成加密动作,而对于读数据的 IRP 包,在 IRP 包的完成例程中嵌入解密算法,对得到的数据进行解密处理[5]。

3.3 写数据加密的实现

在用户写受保护的文件时,文件加解密客户端能透明地对文件写入的内容进行加密。加密文件判别模块判别出此文件为受保护文件后,将密钥传递给加解密过滤驱动,并由过滤驱动完成写操作。为了使过滤驱动能够加密对文件的写操作,需要拦截 IRP_MJ_WRITE 和 IRP_MJ_DEVICE_CONTROL 两种 IRP。当应用层进程调用 API:DeviceIOControl 时,向系统发出 IRP_MJ_DEVICE_CONTROL,过滤驱动可以通过处理此 IRP 获得传递给驱动的加密密钥,进而在拦截 IRP_MJ_WRITE 的例程中进行加密[6,7]。

下面将分别介绍这两个例程的实现。

(1)处理 IRP_MJ_DEVICE_CONTROL 例程。

首先获得应用层程序在 IRP 中携带的输入数据其操作代码,如果携带了传递密钥的操作代码,例程将存储密钥,并直接完成此 IRP,否则将 IRP 交给底层驱动处理[8]。

伪代码如下:

```
NTSTATUS EncryptDeviceControl(IN PDEVICE_OBJECT DeviceObject,IN PIRP Irp)
{
PVOID inputBuffer;
ULONG inputBufferLength;
Irp->IoStatus.Status=STATUS_SUCCESS;
Irp->IoStatus.Information=0;
ioControlCode=irpStack->Parameters.DeviceIoControl.IoControlCode;
inputBuffer=Irp->AssociatedIrp.SystemBuffer;
inputBufferLength=irpStack->Parameters.DeviceIoControl.InputBufferLength;
if(ioControlCode==INPUT_KEY)
{
SetKey(inputBuffer);//将输入的加密密钥保存起来
IoCompleteRequest(Irp,IO_NO_INCREMENT);//完成此 IRP
Return STATUS_SUCCESS;
}
//接下来将 IRP 继续下发传递给下层的驱动处理
.....
.....
}
```

(2)处理 IRP_MJ_WRITE 例程。

需要区分 IRP 携带数据的两种方式,由 MDL 结构指定或直接包含数据的指针。

其实现的伪代码如下:

```
NTSTATUS EncryptWrite(IN PDEVICE_OBJECT DeviceObject,
IN PIRP Irp)
{
.....//完成初始化的工作
PIO_STACK_LOCATION irpsp=IoGetCurrentIrpStackLocation
(Irp);
switch(irpsp->MinorFunction)
{
case IRP_MN_NORMAL:
{
if(Irp->MdlAddress!=NULL)//判断写的数据是否是在
MDL 结构中
{
产生一个新的 MDL 结构,将其指向一个加密后的数据库,并将
Irp->MdlAddress 指针指向新的 MDL 结构
}
else
{
buffer=Irp->UserBuffer;
Encrypt(buffer,GetKey());//加密所写的数据
}
.....//其它的必要操作
}
```

3.4 读数据解密的实现

在用户读受保护的文件时,文件加解密客户端能透明地对读出的原始内容进行解密。加密文件判别器判别出此文件为受保护文件后,将密钥传递给加解密过滤驱动,并由过滤驱动完成读操作。和写操作类似,需要拦截 IRP_MJ_WRITE 和 IRP_MJ_DEVICE_CONTROL 两种 IRP。IRP_MJ_DEVICE_CONTROL 例程的用途和实现方式和加密时相同。过滤驱动拦截 IRP_MJ_READ 的例程中进行加密。IRP_MJ_READ 例程的实现需要利用完成例程,在完成例程中完成加解密过程。

下面将介绍这个例程的实现。

```
NTSTATUS EncryptWrite(IN PDEVICE_OBJECT DeviceObject,
IN PIRP Irp)
{
.....初始化操作
KeInitializeEvent(&waitEvent,NotificationEvent,FALSE);
IoCopyCurrentIrpStackLocationToNext(Irp);
IoSetCompletionRoutine(Irp,EncryptReadCompletion,
&waitEvent,TRUE,TRUE,TRUE);//设置完成例程
status=IoCallDriver(devExt->AttachedToDeviceObject,Irp);
if(STATUS_PENDING==status)
{
```

```
status = KeWaitForSingleObject ( &waitEvent, Executive, KernelMode,FALSE,NULL);//等待完成例程发出的解密完成消息
}
IoCompleteRequest(Irp,IO_NO_INCREMENT);
Return STATUS_SUCCESS;
}
```

代码中首先初始化一个消息事件,将消息事件作为参数传递给完成例程,接着调用底层驱动获得读出的数据,IRP 返回后系统将启动一个新的线程来执行完成例程。

主线程等待消息事件,同时完成例程执行数据的解密操作,解密完成后完成例程发送消息事件,这样主线程被唤醒并正常返回。

完成例程的伪代码如下:

```
NTSTATUS SfReadCompletion ( IN PDEVICE_ OBJECT DeviceObject,IN PIRP Irp,IN PVOID Context)
{
PKEVENT event=Context;
ASSERT(IS_MY_DEVICE_OBJECT(DeviceObject));
……//完成初始化的工作
PIO_ STACK_ LOCATION irpsp = IoGetCurrentIrpStackLocation (Irp);
switch(irpsp->MinorFunction)
{
case IRP_MN_NORMAL:
{
if(Irp->MdlAddress! =NULL)//判断读的数据是否是在MDL结构中
{
产生一个新的MDL结构,将其指向一个解密后的数据块,并将Irp->MdlAddress指针指向新的MDL结构,从而返回被解密的数据
}
else
{
buffer=Irp->UserBuffer;
Decrypt(buffer,GetKey());//解密文件
}
……//其它的必要操作
KeSetEvent(event,IO_NO_INCREMENT,FALSE);//向主例程发送完成解密的消息
return STATUS_MORE_PROCESSING_REQUIRED;
}
```

4 两种加密技术比较。

以上两种加密技术由于工作在不同的层面,从应用效果、开发难度上各有特点。表 1 从各方面进行了简单比较。

表 1 钩子透明加密和过滤驱动加密的比较

项目	钩子透明加密	驱动透明加密
工作原理	拦截打开和关闭的动作,在这 2 个动作里做文件的静态加解密处理	拦截的读和写的动作,在这 2 个 IRP 请求中进行动态的加解密处理
工作方式	HOOK 应用程序和文件类型	接受系统 IRP 进行处理
工作层面	用户层	内核层
与应用程序的关联度	直接与程序工作方式相关,对新应用程序加密或应用程序升级可能需要再开发	与程序工作方式无关,但要监控应用程序名单
加密效率	应用层加解密,速度慢,大文件容易死机	内核层加解密,受 Windows 保护,稳定性好,速度快
网络操作能力	不受限制	需要专门处理
开发难度	HOOK 技术相对容易	驱动技术开发难度较大

5 结束语

钩子透明加密和过滤驱动加密是目前两种主流的透明加密方法,二者的区别在于截获文件操作的位置不同,比较而言过滤驱动加密效果更好,但是实现的难度也相当的大,在安全领域具有很高的研究价值和实用价值。

参考文献:

[1] Awan M A, Khiyal S H. Stackably extensible template layer for file system development under windows NT family[M]. [s.l.]:IEEE,2004.

[2] Nagar R. Windows NT File System Internals: A Developer's Guide[M]. [s.l.]:O'Reilly,1997.

[3] John H, Popek Gerald J. File - system development with stackable layers[J]. ACM Transactions on Computer Systems, 1994,12(1):58-89.

[4] 武安河. Windows 2000/xp WDM 设备驱动程序开发[M]. 第 2 版. 北京:电子工业出版社,2005.

[5] 邢书宝,李 刚,薛惠锋. 一次一密加密系统设计与实现[J]. 计算机技术与发展,2007,17(3):150-155.

[6] 郑 磊,马兆丰,顾 明. 基于文件系统过滤驱动的安全增强型加密系统技术研究[J]. 小型微型计算机系统,2007(7):1181-1184.

[7] 邵 昱,萧蕴诗. 基于文件系统过滤驱动器的加密软件设计[J]. 计算机应用,2005(5):1151-1152.

[8] Richter J. Windows 核心编程[M]. 北京:机械工业出版社,2000.