题目：

防终止的基于Windows透明加密过滤系统研究与实现

目录

答辩内容：

目录

选题背景及意义

安全事件屡见不鲜

- 2008年初“艳照门事件”
- 2011年，国际上RSA、Sony大批大型公司接连受到黑客攻击，大量信息泄露。
- 国内CSDN社区数白万用户数据被窃取。
- 关于信息泄露的安全事件远不仅于此，信息安全问题已不容忽视，现今热门话题。

透明加密技术应运而生

- 钩子技术和过滤驱动技术。
- 过滤驱动工作于内核层，安全性高，主流手段。
- sfilter 和 minifilter。Minifilter微软首推，可移植性、稳定性好，复杂度低等优点。

内网文档安全亟待解决

- 最常用的信息安全保护技术，防火墙技术和入侵检测技术，在一定程度上阻断网络威胁，对企业内部人员主动泄密的行为却无能为力。
- 然而，内部泄漏主要原因。据FBI/CSI的调查，文件信息安全的威胁80%以上来自企业员工的不法操作。
- 采取的安全策略是：禁止连接网络、禁止USB接口、禁止使用软盘等。一定程度上保证文档安全，却极其不便。
- 保证企业内部敏感信息的安全不仅要阻断来自网络的威胁，还要防范非法人员进入公司内部的终端设备，泄漏信息。

国内外研究

- 透明加密的概念最早由微软发布Windows 2000 操作系统时提出，其加密文件系统（EFS）提供加密功能。
- 赛门铁克推出的企业版端点安全软件SEP、趋势科技推出的云安全软件Pc-cillin、诺顿安全软件Norton中的文件加密工具DISKREET等。
- 国内起步较晚，2000年开始，早期较多采用基于应用层的文件加密技术，之后开始较多尝试调用层和内核层的透明加密。但仍处于起步阶段，研究阶段。并且大多基于传统型过滤驱动开发，稳定性和兼容性较差，开发过程繁琐，开发周期长。

选题背景及意义

基于以上考虑，本文提出了一个完备的电子文档保护解决方案，并采用Minifilter框架实现了防终止的基于Windows透明加密过滤系统。

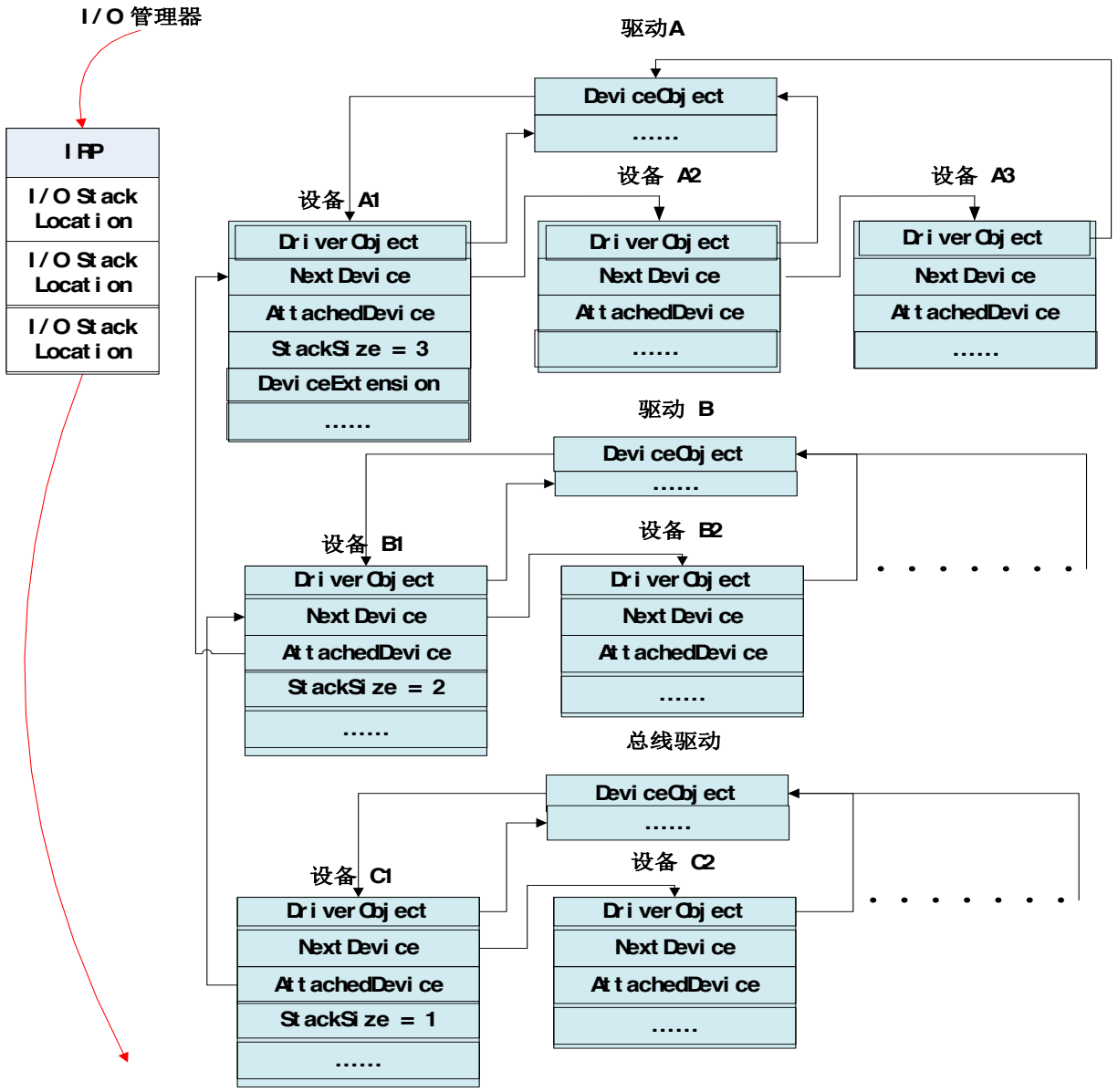
目录

本文研究内容

- Windows系统、驱动开发、内核编程等原理的学习与研究。
- 死机或蓝屏是常态。搭建主机-虚拟机的双机调试环境。
- 深入学习并研究Minifilter，在此基础上实现内核透明加密过滤驱动模块，支持自动加解密功能。
- 充分研究应用层和内核层的通信机制，在此基础上实现内核层和用户层的信息交互。
- 构建C/S架构。服务器模块，主要负责策略的制定与下发，客户端的认证、监控和审计，密钥管理与分发等功能。客户端应用层控制模块，主要任务是与服务器建立连接，作为服务器与内核加密模块通信的桥梁。在此基础上深入研究socket通信机制与心跳机制，实现服务器和客户端的通信功能。此外还制定较为完善的客户端-服务器认证机制，内网IP认证和口令认证。
- 深入研究进程防终止技术，防止蓄意关闭操作。
- 深入研究Windows剪切板机制，在此基础上实现对剪切板的监控，防止机密数据被复制/剪切带走。

目录

驱动设备框架图



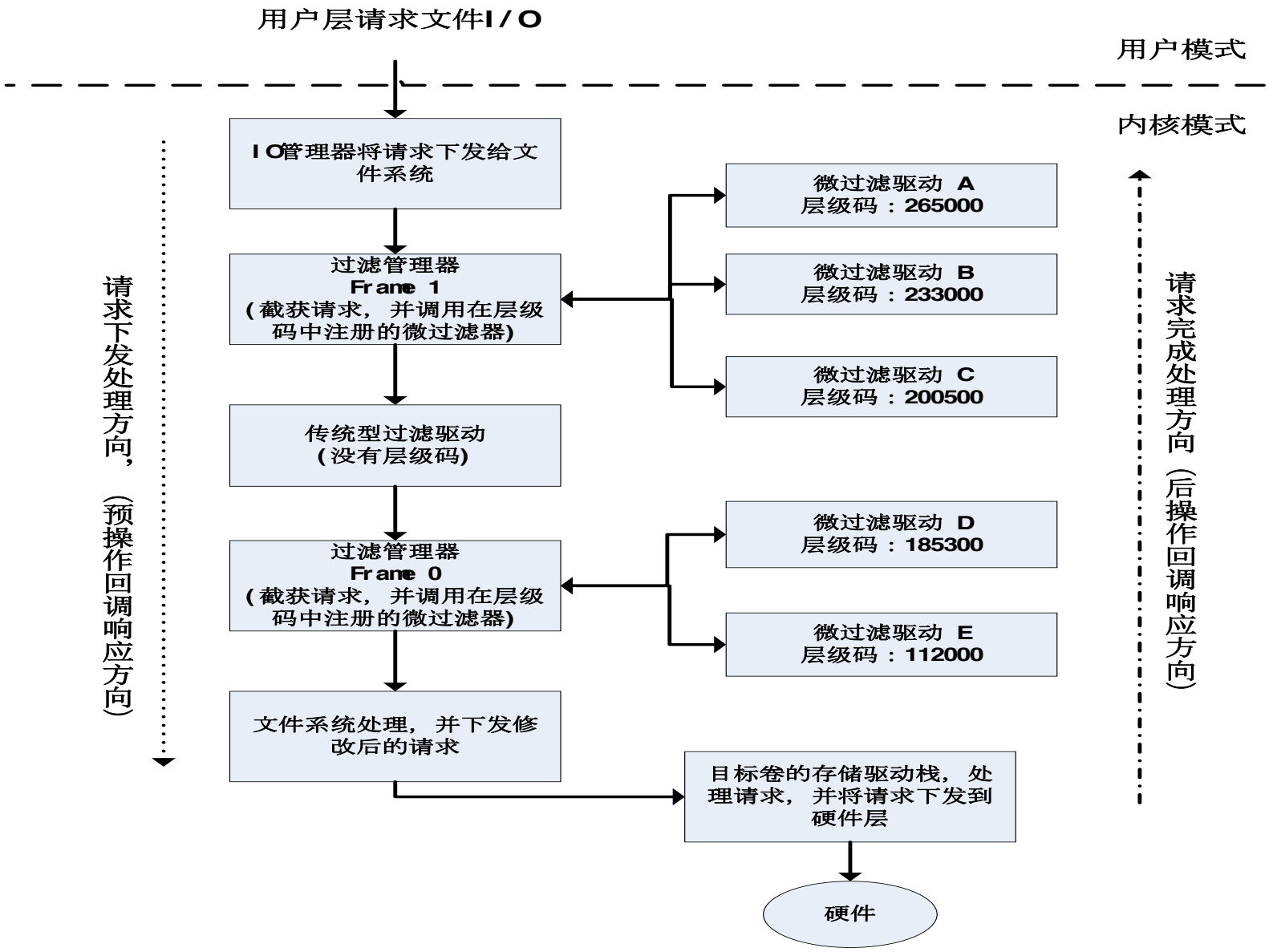
驱动对象：每个成功加载的内核驱动镜像都有且只有一个驱动对象表示，作为I/O管理器加载的一个实例。

设备对象：由驱动创建，设备对象间通过NextDevice域连接。设备链表由I/O管理器维护。此外，设备对象是唯一能接收IRP的实体。

IRP：该数据结构被用来描述I/O请求。使用这样一个结构对通信过程中涉及的大量参数进行封装，使得通信过程变得很简单。

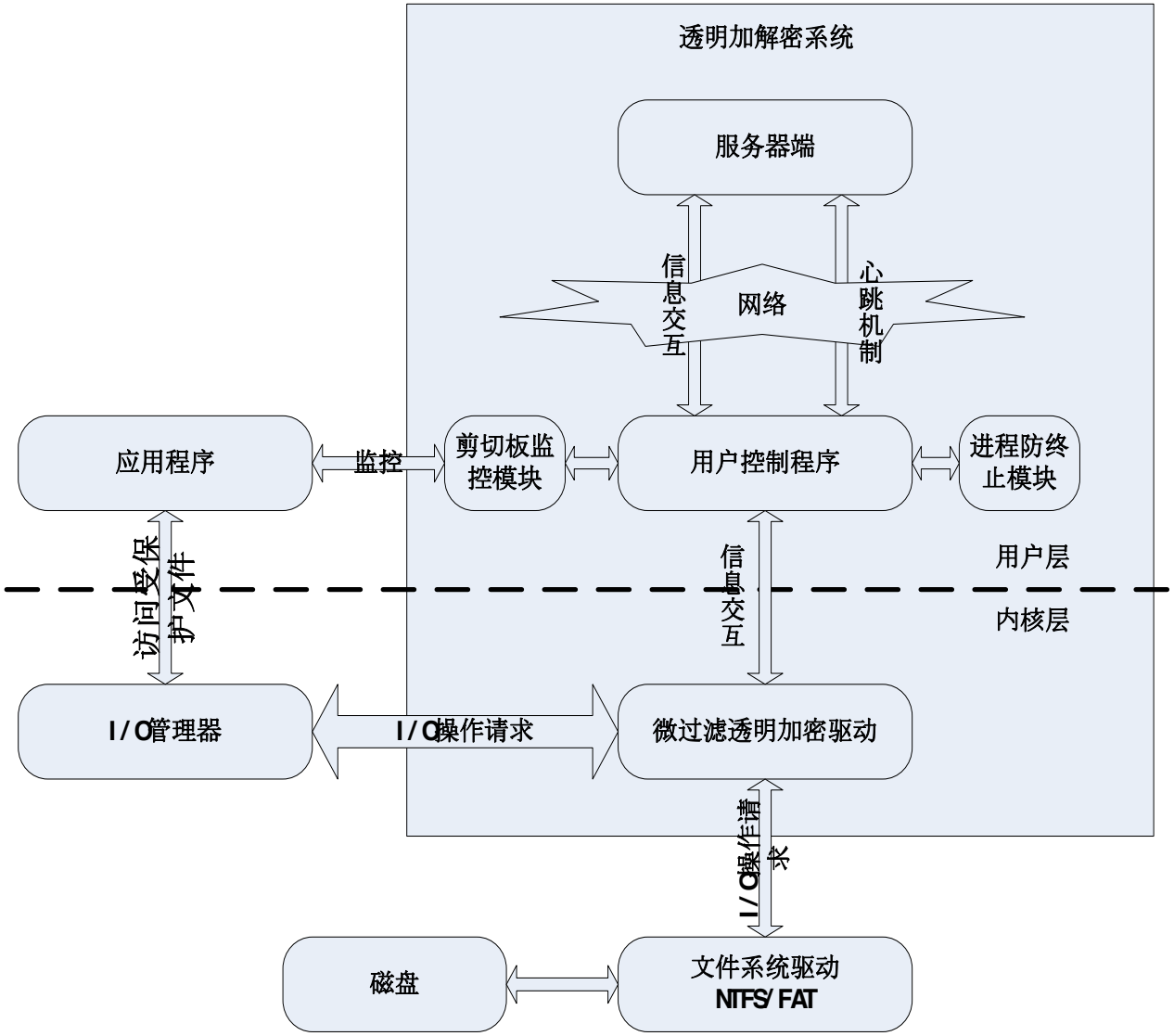
IRP Stack Location：I/O管理器为每个IRP创建一个I/O Stack Location 数组（Stack Locations），存放将要处理该IRP请求的所有驱动对应的IO_STACK_LOCATION结构，该结构中封装了与该驱动相关的参数。

Minifilter框架图



目录

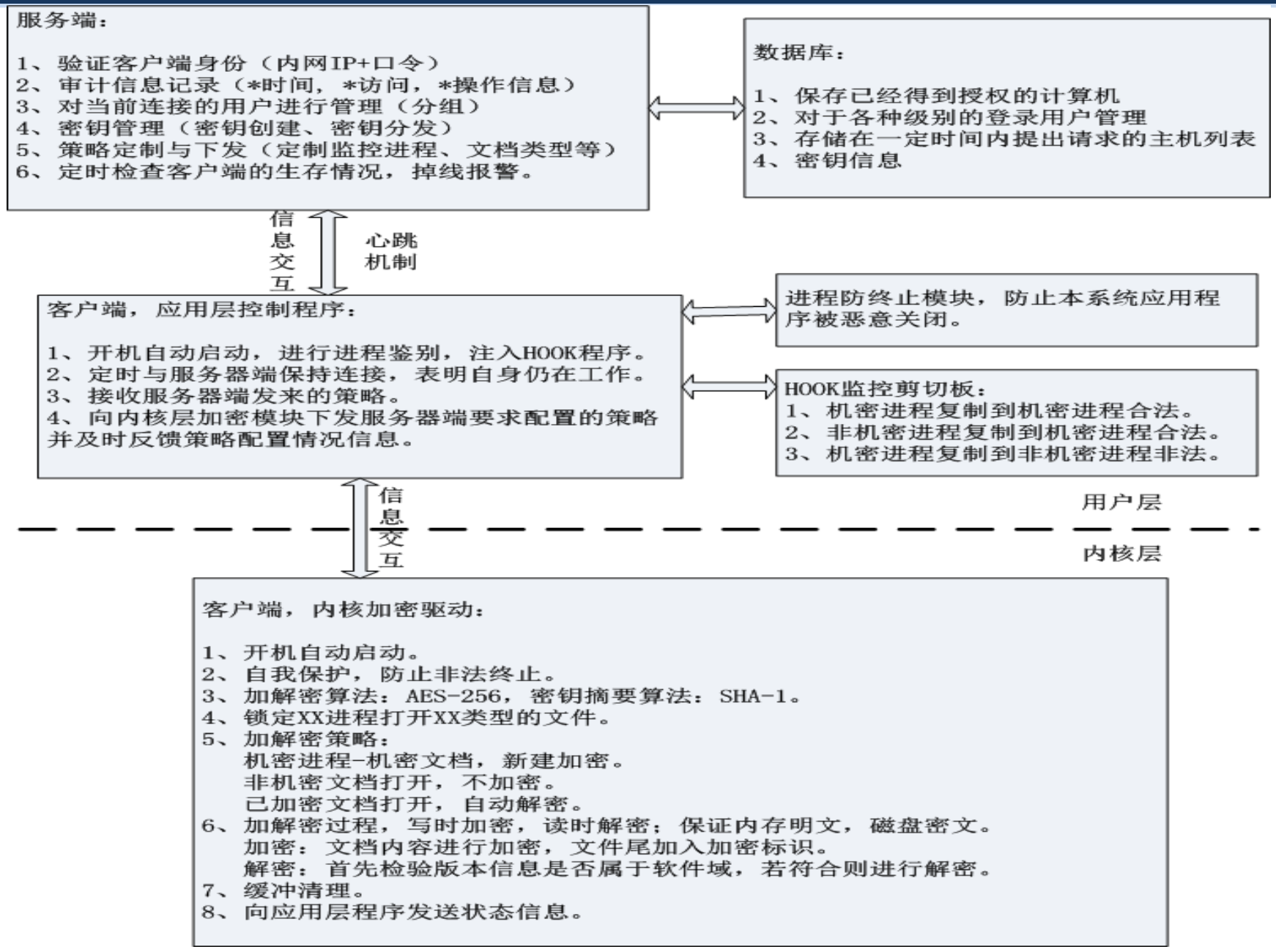
系统总体设计图



透明——是指在不改变用户使用习惯、应用程序和电子文件格式的前提下，对指定文档类型进行实时监控，实现对受保护文档的自动加解密功能。文档一旦脱离加解密环境，将无法正常解密，从而对电子文档起到保护的作用。

文档在磁盘中是密文，在内存中是明文。

模块划分图


服务端：
1、验证客户端身份（内网IP+口令）
2、审计信息记录（*时间，*访问，*操作信息）
3、对当前连接的用户进行管理（分组）
4、密钥管理（密钥创建、密钥分发）
5、策略定制与下发（定制监控进程、文档类型等）
6、定时检查客户端的生存情况，掉线报警。
数据库：
1、保存已经得到授权的计算机
2、对于各种级别的登录用户管理
3、存储在一定时间内提出请求的主机列表
4、密钥信息
信息交互
心跳机制
客户端，应用层控制程序：
1、开机自动启动，进行进程鉴别，注入HOOK程序。
2、定时与服务器端保持连接，表明自身仍在工作。
3、接收服务器端发来的策略。
4、向内核层加密模块下发服务器端要求配置的策略并及时反馈策略配置情况信息。
进程防终止模块，防止本系统应用程序被恶意关闭。
HOOK监控剪切板：
1、机密进程复制到机密进程合法。
2、非机密进程复制到机密进程合法。
3、机密进程复制到非机密进程非法。
信息交互
用户层
内核层
客户端，内核加密驱动：
1、开机自动启动。
2、自我保护，防止非法终止。
3、加解密算法：AES-256，密钥摘要算法：SHA-1。
4、锁定XX进程打开XX类型的文件。
5、加解密策略：
机密进程-机密文档，新建加密。
非机密文档打开，不加密。
已加密文档打开，自动解密。
6、加解密过程，写时加密，读时解密；保证内存明文，磁盘密文。
加密：文档内容进行加密，文件尾加入加密标识。
解密：首先检验版本信息是否属于软件域，若符合则进行解密。
7、缓冲清理。
8、向应用层程序发送状态信息。

系统开发主要工具

软件	作用
Visual Studio 2010	开发平台
WinDDK7600.16385.1	驱动开发包
Windebug	调试工具
VMware Workstation11	双机调试环境、测试环境
SRVINSTW.EXE	安装传统型驱动，Minifilter使用INF文件进行安装。卸载指定服务。
DebugView.exe	调试工具，查看内核输出

内核透明加密过滤驱动——关键结构定义

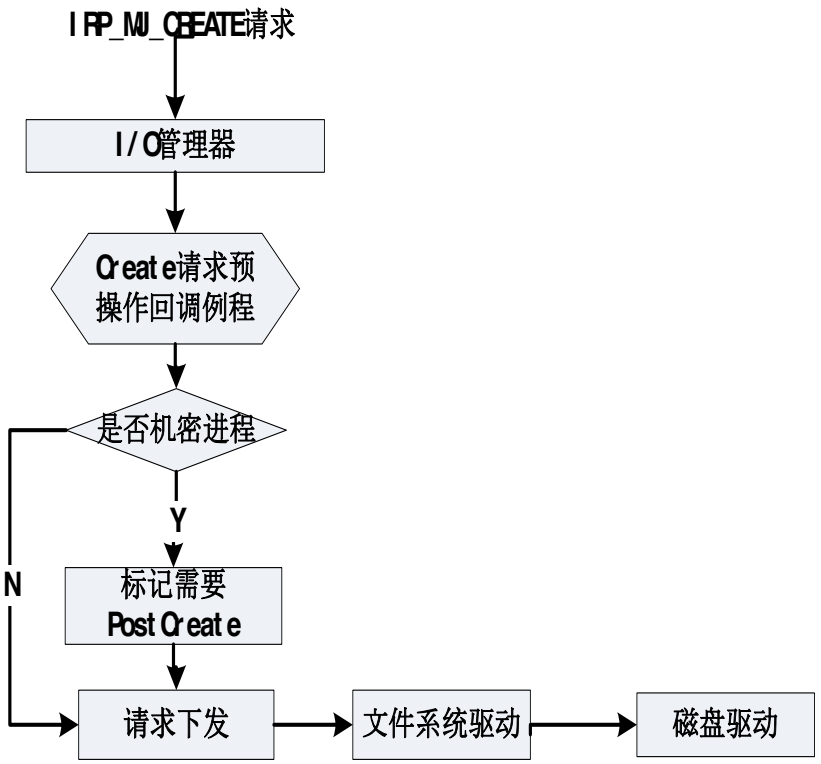
- 加密标识结构定义。加密标识用于标识受保护文档，其中存放着与文档相关的必要参数，在受保护文档关闭时被写入文件尾。并且本文采用了适当的数据隐藏手段，对该加密标识进行隐藏处理，使其对用户层应用程序来说是不可见的。

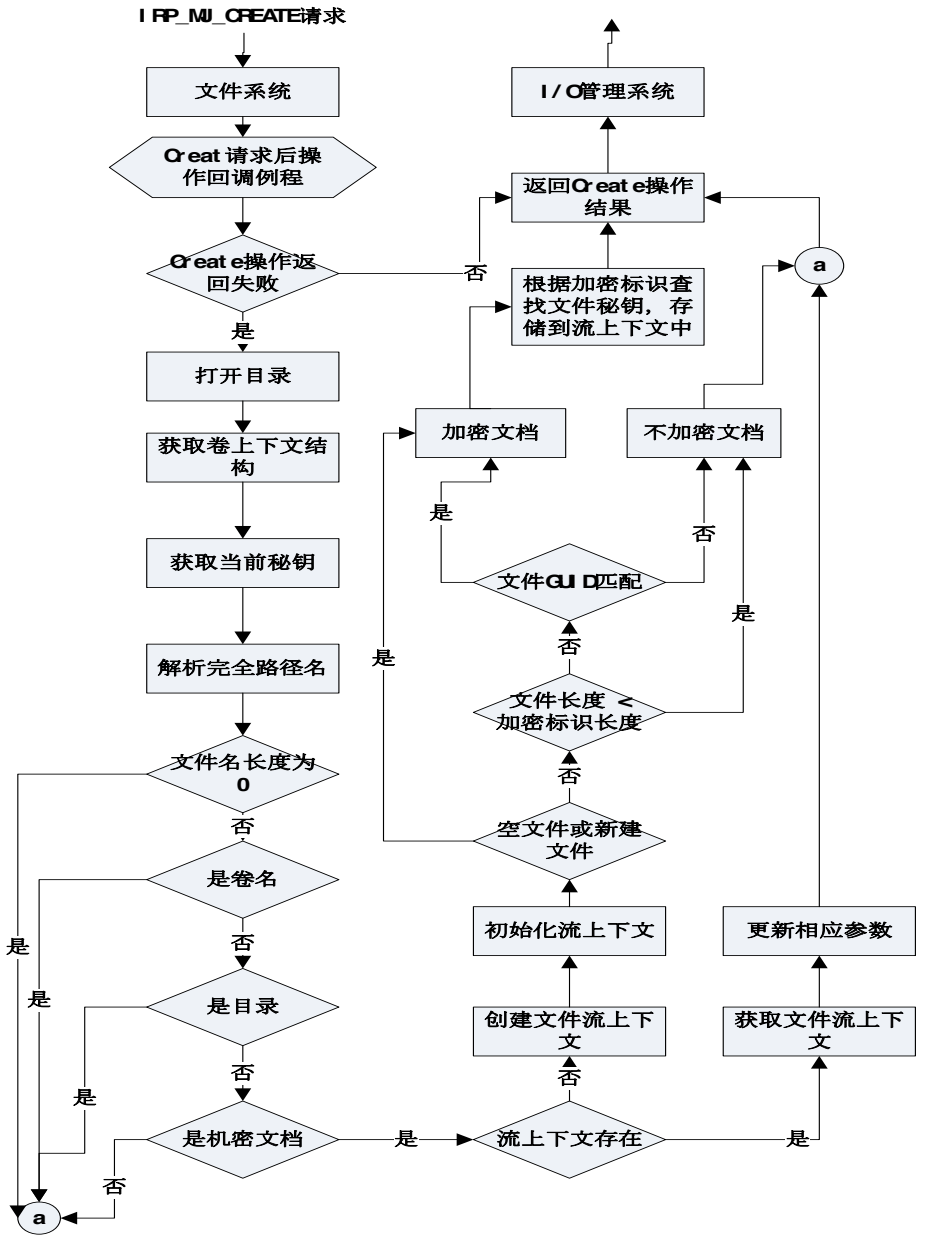
- 卷上下文结构定义。在卷挂载时将本文定义的卷上下文挂载带卷上，来记录必要的参数。必要时，可在操作回调例程中读取。

- 流上下文结构定义。流上下文与文件对应，用于记录与文件相关的参数。

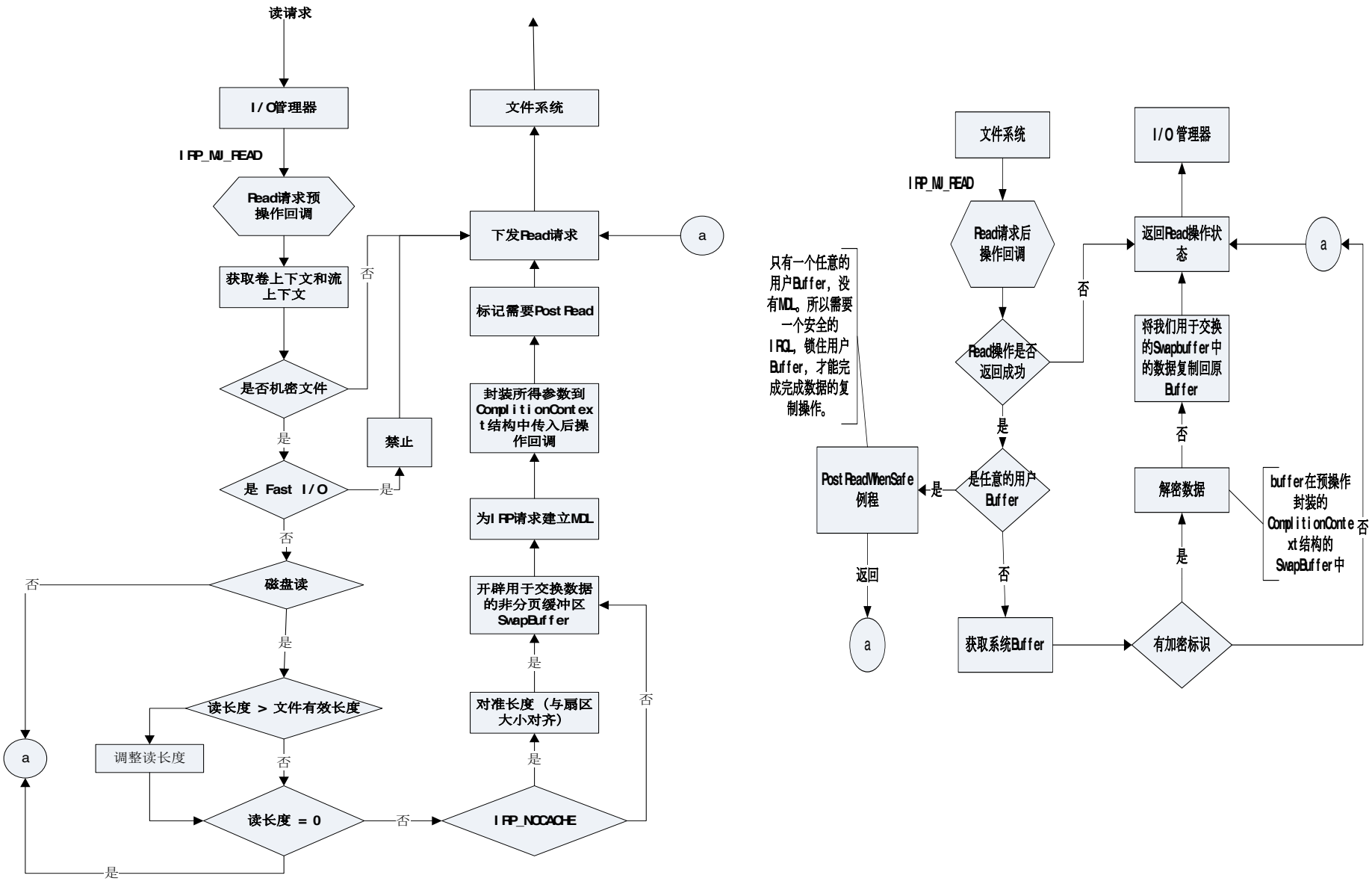
- 预操作回调例程到后操作回调例程的上下文结构定义。有些内核对象不能再DPC中断级中被获取，但在DPC中断级中可安全释放。因此通常会在预操作回调中取得这些对象，再传给后操作回调例程，并由后操作回调例程完成释放工作。这个过程可通过操作回调例程中的Completion参数来实现。

内核透明加密过滤驱动——Create回调例程

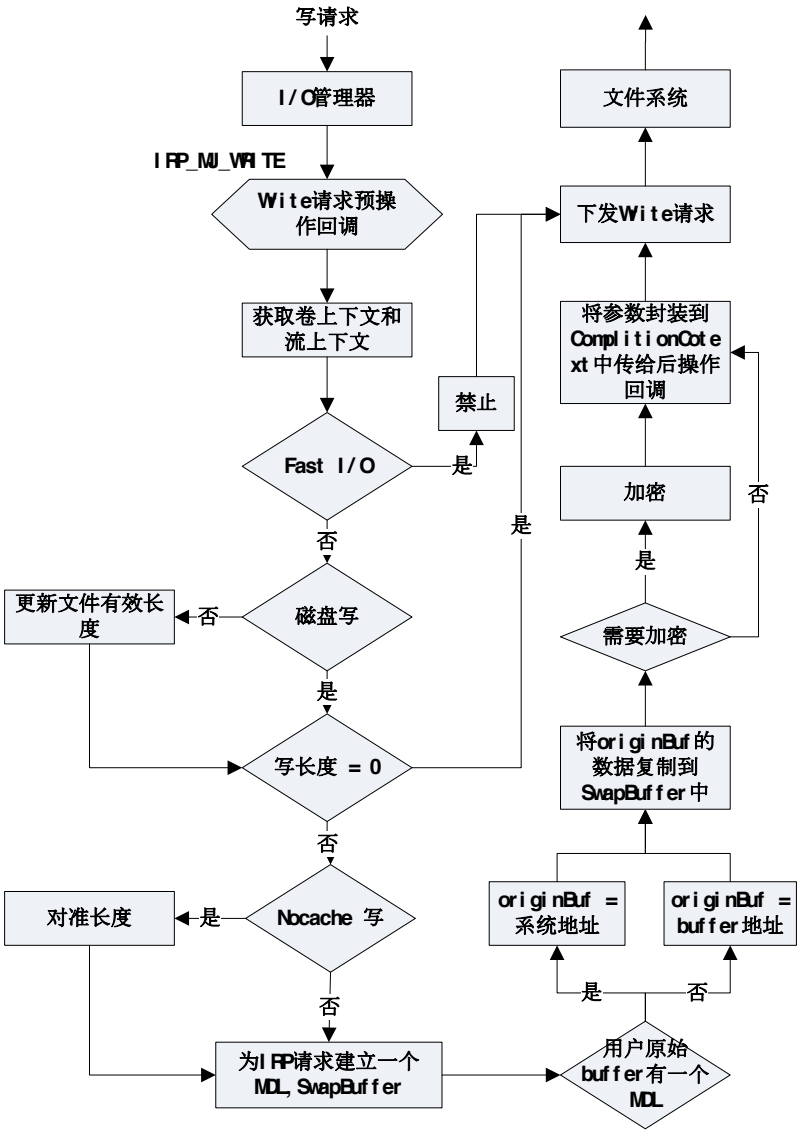


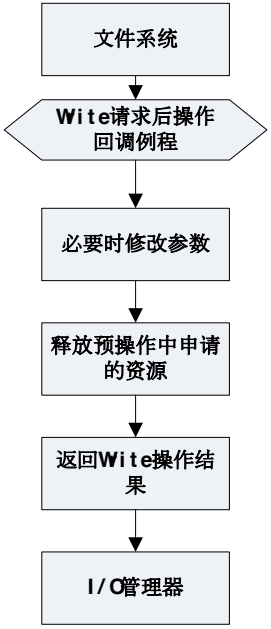


内核透明加密过滤驱动——Read回调例程



内核透明加密过滤驱动——Write回调例程





内核透明加密过滤驱动——其他回调例程

- **IRP_MJ_CLOSE**：本系统为该请求注册预操作回调例程EncrytingFilter_PreClose。当该回调例程被触发时，必须要减少流上下文的引用。若流上下文的引用减少到0，文档被关闭，若该文档为受保护文档，需要将加密标识写入文件尾，并且清除缓存。

- **IRP_MJ_QUERY_INFORMATION**：注册了EncrytingFilter_PreQueryInfo预操作回调例程和EncrytingFilter_PostQueryInfo后操作回调例程，实现对受保护文档的文件标识进行隐藏的工作。当用户需要查询文件信息时，要重新计算，返回除了填充和文件标识之外的，真正的文本内容长度与偏移。

- **IRP_MJ_SET_INFORMATION**：注册了EncrytingFilter_PreSetInfo预操作回调例程和EncrytingFilter_PostSetInfo后操作回调例程。当需要设置文件信息时，必须要经过重新计算，以达到对文件标识以及额外填充内容进行隐藏的目的。

- **IRP_MJ_CLEANUP**：注册了EncrytingFilter_PreCleanup预操作回调例程。当IRP_MJ_CLEANUP请求到来时，表示文件对象句柄的应用计数值已经减少到0。换句话说，就是文件对象的所有句柄都已经被关闭。此时，需要对机密文档的缓冲进行清除，以防机密内容泄露。

内核透明加密过滤驱动——内核与应用层通信

```
NTSTATUS Msg_CreateCommunicationPort( IN PFLT_FILTER pFilter )
{
    ......
    /*   建立安全描述符 */
    status = FltBuildDefaultSecurityDescriptor(&pSecurityDescriptor,FLT_PORT_ALL_ACCESS) ;
    /* 初始化服务端口 */
    RtlInitUnicodeString(&uPortName, SERVER_PORTNAME) ;
    InitializeObjectAttributes(&objectAttributes, &uPortName,
        OBJ_CASE_INSENSITIVE|OBJ_KERNEL_HANDLE, NULL,pSecurityDescriptor) ;
    /* 创建新的服务端口 */
    status = FltCreateCommunicationPort(
        pFilter,                          //   过滤驱动对象句柄
        &g_pServerPort,                   //   服务端口
        &objectAttributes,  NULL,
        Msg_ConnectNotifyCallback,        //   连接回调例程
        Msg_DisconnectNotifyCallback,     //   断开连接回调例程
        Msg_MessageNotifyCallback,        //   发送信息回调例程
        MAX_CONNECTIONS                   //   最大连接数
        ) ;
    ...... ......
    return status ;
}
```

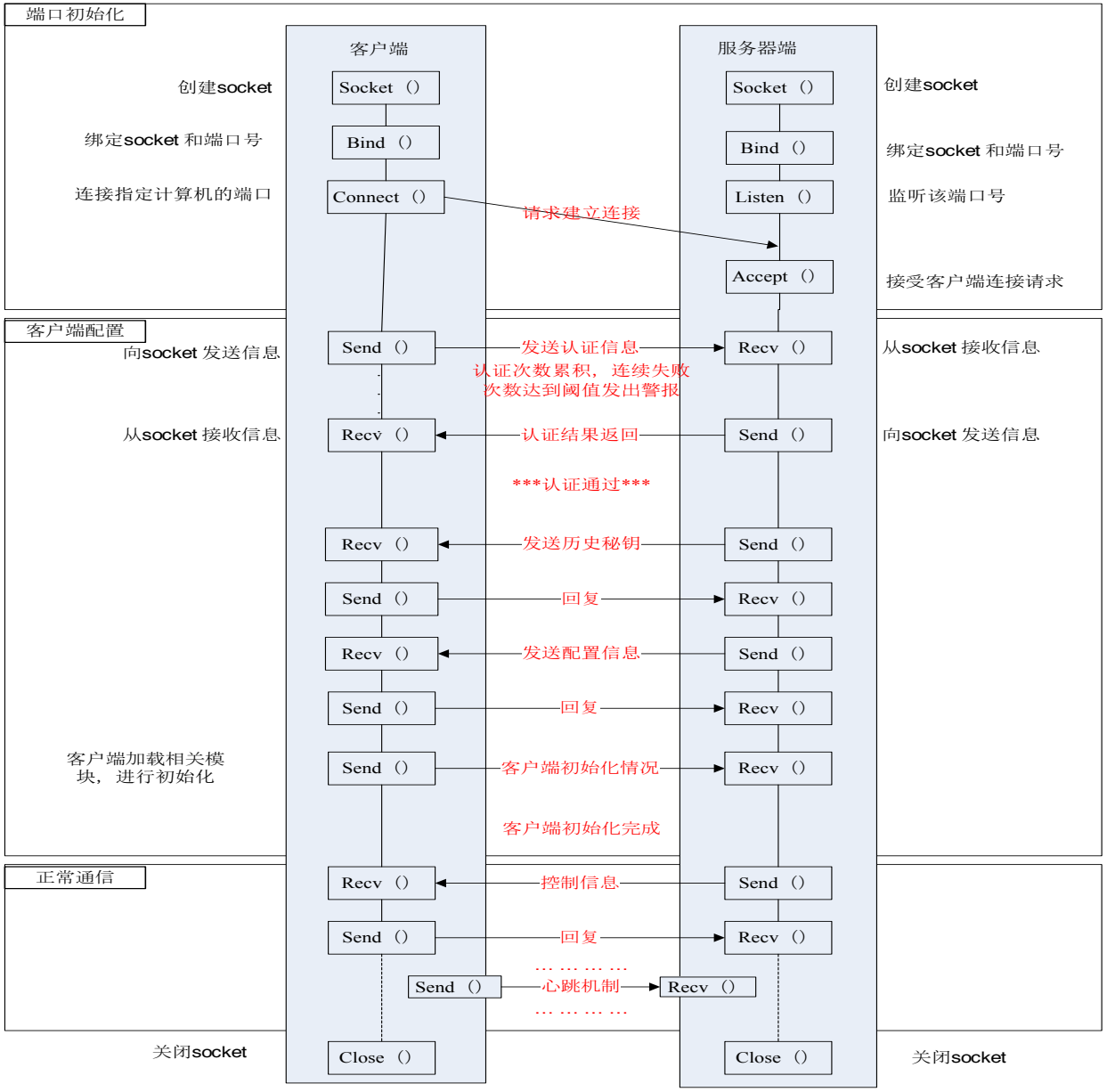
剪切板监控模块与进程防终止模块

HOOK API的实现过程主要分为两个步奏：其一是实现一个与原API一样的函数接口，其中加入自己的控制代码，用新API的地址替换原API的地址；二是将实现HOOK API功能的DLL注入到目标进程中。DLL可跟随系统钩子一起被注入到目标进程中，本系统利用鼠标钩子，实现自定义HOOK DLL的注入。这样一来只要HOOK的API被调用就会被重定向到我们的替换函数中。

- 剪切板
- User32.dll
- GetClipboardData()
- SetClipboardData()

- 防终止
- Kernel32.dll
- OpenProcess()
- TerminateProcess()

客户端-服务器通信模块


端口初始化
客户端
服务器端
创建socket
Socket ()
Socket ()
创建socket
绑定socket 和端口号
Bind ()
Bind ()
绑定socket 和端口号
连接指定计算机的端口
Connect ()
请求建立连接
Listen ()
监听该端口号
Accept ()
接受客户端连接请求
客户端配置
向socket 发送信息
Send ()
发送认证信息
Recv ()
从socket 接收信息
认证次数累积，连续失败
次数达到阈值发出警报
从socket 接收信息
Recv ()
认证结果返回
Send ()
向socket 发送信息
认证通过
Recv ()
发送历史秘钥
Send ()
Send ()
回复
Recv ()
Recv ()
发送配置信息
Send ()
Send ()
回复
Recv ()
客户端加载相关模块，进行初始化
Send ()
客户端初始化情况
Recv ()
客户端初始化完成
正常通信
Recv ()
控制信息
Send ()
Send ()
回复
Recv ()
… … … …
Send ()
心跳机制
Recv ()
… … … …
关闭socket
Close ()
Close ()
关闭socket

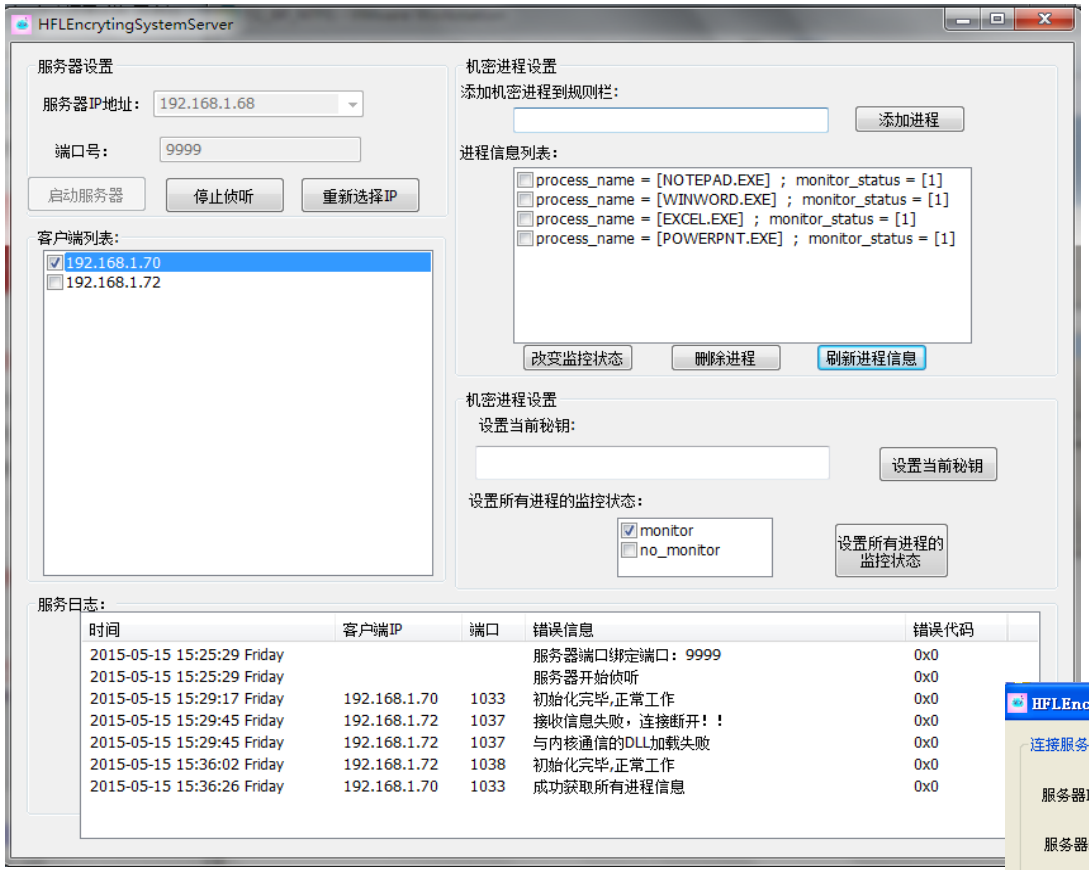
目录

测试环境

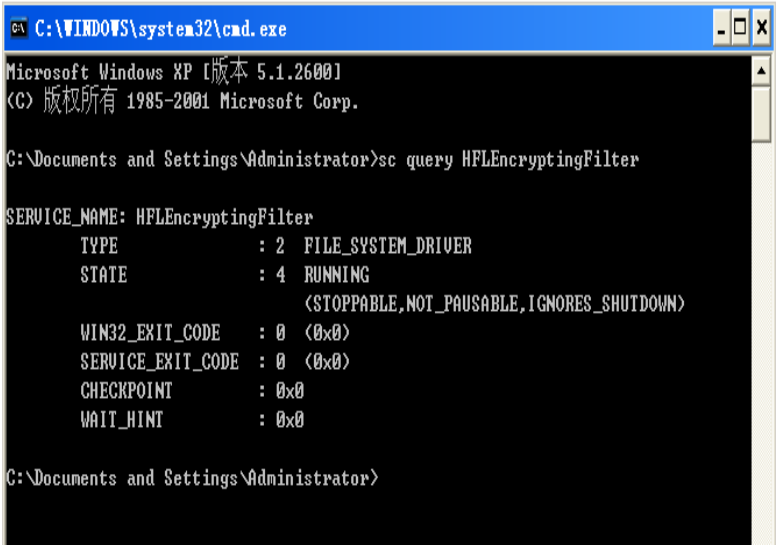
操作系统	Windows XP、Windows 7
处理器	AMD A6-4455M APU with Radeon(tm) HD Graphics ， MMX, 2.1GHz
内存	1024MB RAM
文件系统类型	FAT、NTFS

接下来仅给出在Windows XP SP3中测试的结果。在Windows7操作系统中的测试结果相同。本系统的兼容性主要取决于内核模块，由Minifilter的框架的优点，理论上本系统可兼容Windows系列操作系统系统，但由于时间原因，除以上两种Windows操作系统版本之外，其他版本未经过严格的测试。

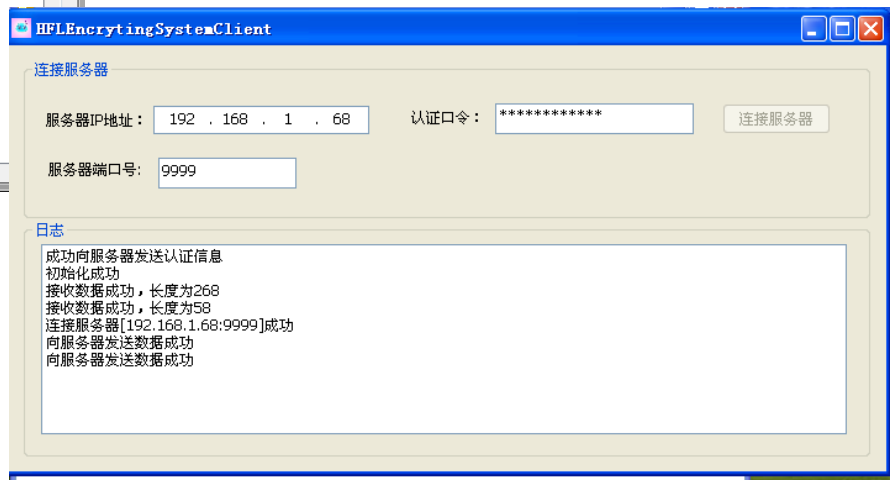
服务器-客户端建立连接



服务器端

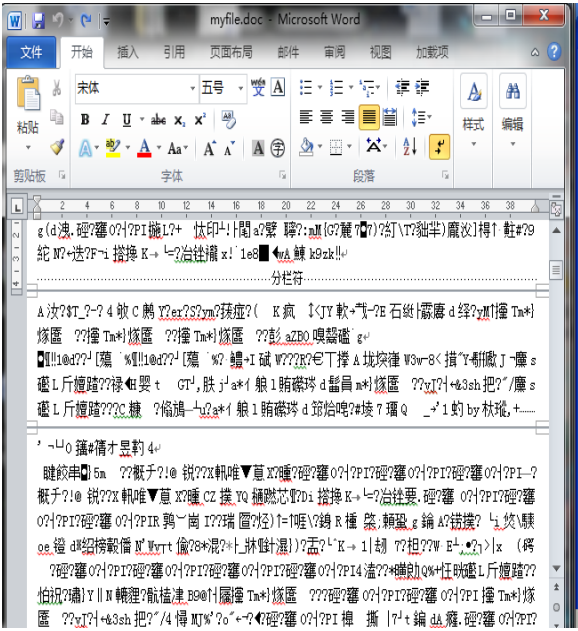


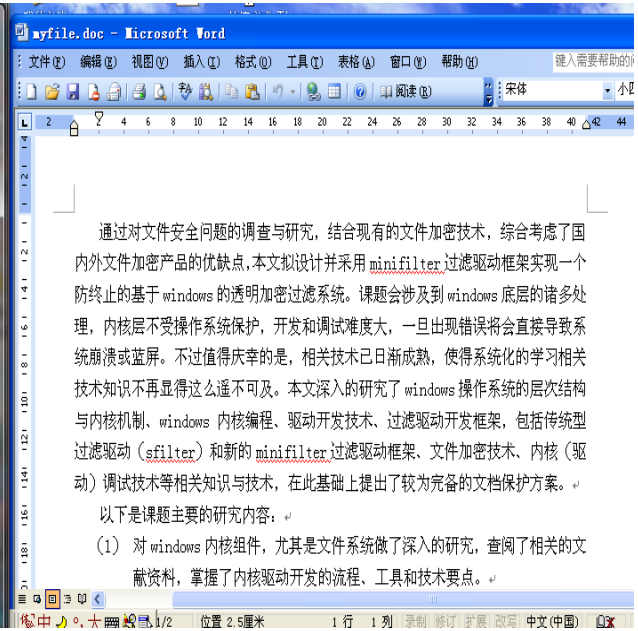
内核模块加载

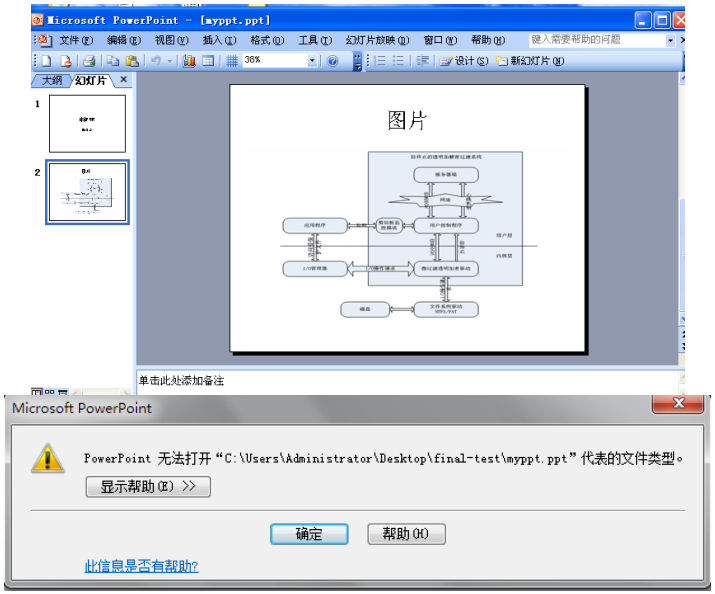


客户端连接

测试

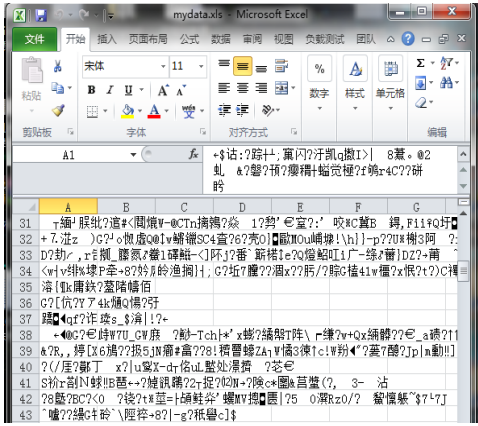


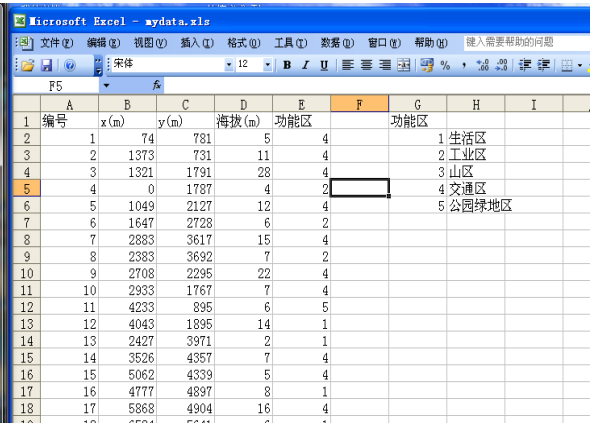


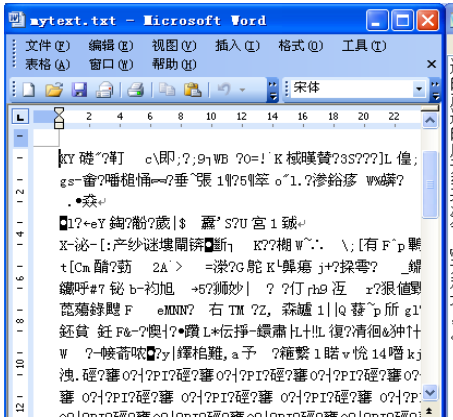


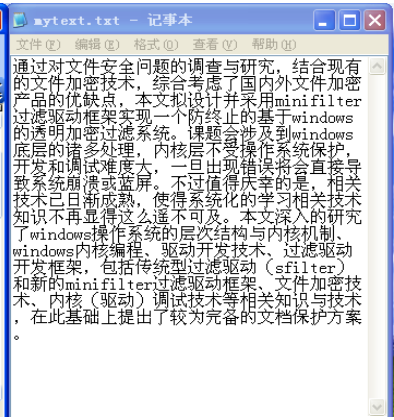
主机-虚拟机word文档

主机-虚拟机ppt文档









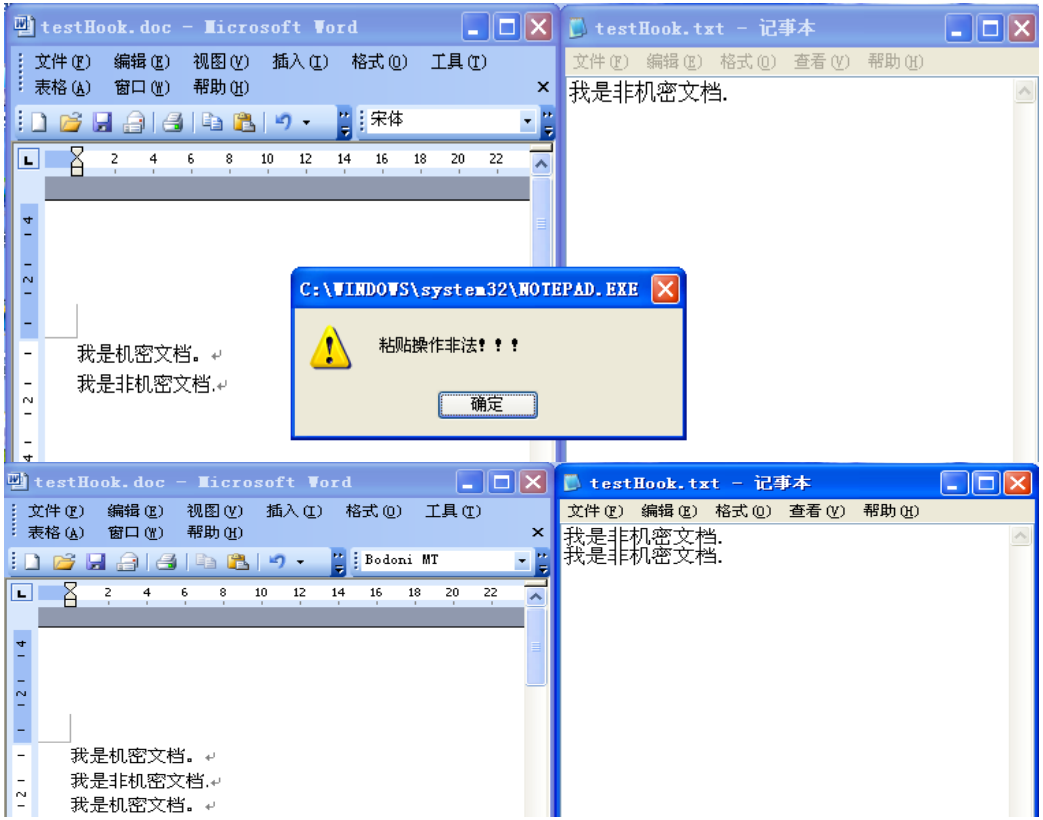
主机-虚拟机excel文档

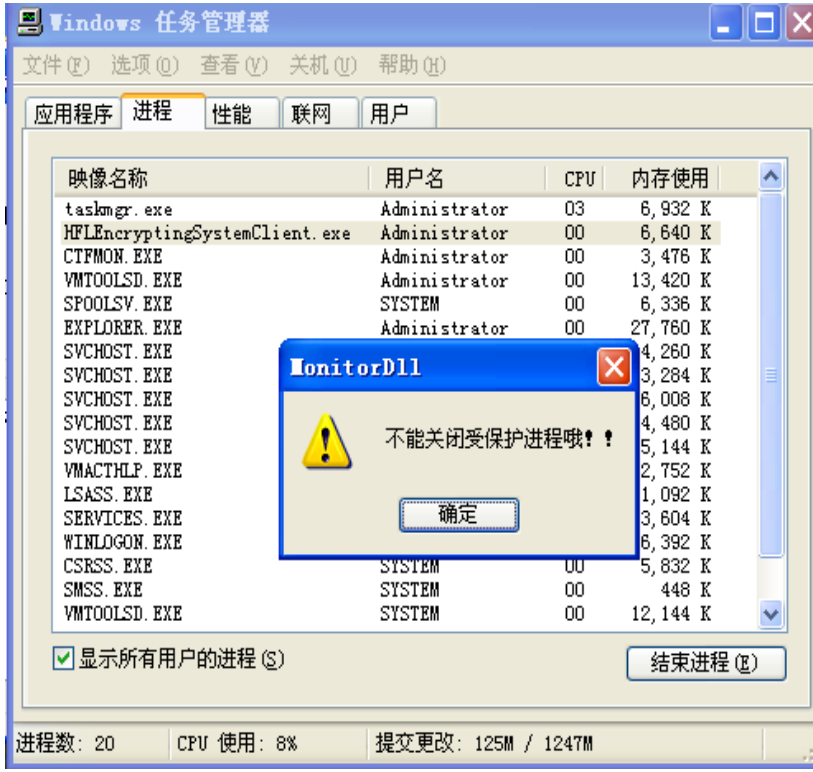
主机-虚拟机txt文档

测试

2015-05-15 16:03:59 Friday　192.168.1.72　1038　连接断开！！！　0x0

心跳机制





进程防终止

剪切板监控，禁止机密进程到非机密进程的粘贴

目录

本系统功能特点

- 实现了对受保护文档的透明加解密功能。
- 系统由服务器端、客户端应用层控制程序、客户端内核层透明加密过滤驱动三个主要模块构成，各模块之间相互制约，相互监控，为保证文档安全提供了很好的支持。
- 采用多个文件密钥，这可由管理员进行设置，并且将密钥的摘要存放到受保护文档的加密标识中，这相对于单一密钥的安全性要高。
- 深入的研究了进程防终止技术，对本系统的进程进行了保护，降低系统被恶意关闭的危险，提高了系统的安全性。在服务器端与客户端之间加入心跳机制，实现对客户端在线情况的监控。
- 结合用户层的安全技术，实现对剪切板的监控，防止用户通过剪切板将机密信息拷贝带走，进一步增强系统的安全性。
- 系统对用户的行为进行较为合理的跟踪审计，此外保证系统具有较好的稳定性、灵活性、方便性、健壮性。

需要改进与完善的不足

- **USB Key提高认证灵活性**。本系统基本上只为内网文档安全提供了保障，内网通过IP地址加口令进行认证。但这对于存在出差办公需求的用户显得不是很方便。虽然在出差情况下也可以通过网络进行口令认证，启动移动设备的加解密环境，但在没有网络的情况下，这个问题就无法解决。针对这样一个问题，本文拟定解决方案是，指定文档授权机制，在授权范围和时间内，用户可以通过单位分发的USB Key启动系统加解密环境，进行正常的工作。

- **防止进程欺骗**。在编写驱动代码时，主要是在PEB中获取进程名来判断当前进程是否为机密进程。但如果有恶意用户将非机密进程名改为机密进程名，系统就无法鉴别，而将其当做机密进程来处理，这将可能导致受保护文档外泄。针对此问题，本文拟定解决方案为：计算机密进程版本信息的摘要值，若企业使用的应用程序是由管理员统一版本的，则这是一个唯一值，然后根据对可执行文件的完整性验证（HASH验证）来判断当前进程是否为真实的机密进程。

- **系统休眠处理**。本文拟定解决方案为：系统进入休眠时，关闭所有打开着的机密文件对象并清除系统缓冲

Thank you

2015.06.05