破解的时候多在系统函数中下断，分析逆向

命令行参数用来执行文件

Ex扩展函数

Win函数需要传入的参数 结构指针

前缀 pv 空指针 dw fdw

Ntapi

Ntdll的api IRP请求，设备对象，驱动对象

IDT IRP

PHD函数库

程序错误处理 函数失败时返回值类型 bool 句柄或Null pvoid long

检测到错误会调用本地 线程存储TLS 错误代码号 getlasterror（错误代码号）

Windows中函数返回值包含的信息太少，错误代码号则有数千，是调试主要手段

线程最后的错误代码 formatmessage作用很多 可以编码错误代码

2、定义错误代码

二、unicode 1、单字节和双字节 2、unicode函数 W 枚举资源 输出文字

3、窗口创建4、字符串函数 5、资源 6、字符串转换

1. 内核对象

创建 打开 操作 符号对象 事件 文件 文件映射 io 作业 互斥 管道 进程 信标 线程 计时器 对象的数据只能在内核访问，外面只能调用api接口

对象句柄 对象属于内核 而不属于进程，利用使用计数来撤销对象 安全描述符 对象的使用者（服务器中用） 可以自己编辑描述符

2、进程内核句柄表 一个数组包含对象指针 访问屏蔽和标志

创建内核对象 文件 线程 文件映射 创建错误可能返回0也可能返回-1，要注意

关闭内核对象 closeHandle 3、内核对象共享 文件映射 邮箱 管道 互斥同步

(1)父子进程继承 可继承句柄声明CreateProcess 传入继承参数 改变句柄的继承标志SetHandleInformation(2)命名对象 create时输入名字 可以用继承性或者duplicateHandle对象句柄复制 （3）Open函数带上名字

Getlasterr 名字空间

四、进程 管理进程的内核对象+进程地址空间 数据+代码+dll+堆栈+堆 线程时间片

1、 cui 和gui的连接不同 入口点winmian 或者main wWinmain

启动函数 检索命令行 环境变量 全局变量初始化 初始化栈和堆 调用构造函数

getStartupInfo() winmain() 2、进程实例句柄 winmain第一参

getmoudlehandle() 返回模块（）或者dll的内存地址

亲缘性 错误模式:如何处理错误 setErrorMode() 驱动器目录getcurrentdirectory 当前目录getfullpathname 系统版本 getversion

3、创建进程 4、startupinfo 包含大量窗口信息

获取模块文件名 getmoudlefilename() 返回exe或者dll模块的 地址 getmoudlehandle

进程命令行createProcess() getcommandline() 访问命令行各部分 commandlinetoArg

进程的环境变量 getenviromentvariable setenviromentvariable

进程亲缘性

终止自己的进程 exitprocess 终止别人的进程 terminateprocess() 返回、

终止时 线程全部终止。Gdi释放，用户对象释放，内核对象关闭，退出代码，内核对象收到通知 内核对象引用计数减一

获取进程退出代码 getexitcodeprocess()

Closehandle 关闭内核对象

2、子进程 线程同步有问题 共享数据，进程间通信 内存映射文件

CreateProcess() 传入 Process\_information 结构 pi &pi

Pi.hThread pi.hprocess

Waitfor singleobject

5、进程枚举

Toolhelp系列 快照函数 pe 对象 pe={sizeof(pe)}是初始化结构的语法，把结构指针

传入指针参数 (&pe)

enumProcess函数

五、作业 一组进程当成实体 创建多个exe进程 同时关闭 服务器多进程

Handle createJobObject

Jobobject\_basic\_limit\_information jobli={0}

OpenJobObject

限制功能 SetInformationJobObject 需要一个Jobobject\_basic\_limit\_information结构

用户界面ui限制

安全限制

2、进程放入作业 AssignProcessToJobObject(j,p) 放入的进程生成进程时，也会自动加入作业 3、终止进程 TerminateJobObject() 4、获取作业统计信息QueryInformationJobObject

传入空的 JobObjectAccountingInformation 结构 \_Io\_Counters\_

GetProcessIoCounters() 已执行读写操作的数量

各种结构 明细 5、作业通知 事件 进程运行的时间 退出的时间 分配的cpu时间是否用完

StartRestrictedProcess

通知状态，未通知状态

六、线程 线程=内核对象+堆栈 内核对象是用来存放统计信息的地方和操作管理

线程堆栈：参数和局部

1 、线程函数 需要一个返回值 作为退出代码 使用参数和局部变量的话，只会在线程堆栈 CreateThread()

不应该使用CreateThread,应该使用编译器的该语法\_\_beginthreadex

Exitthread terminaterhread 获取退出代码 getexitcodethread

TLSgetvalue tiddata块

获取线程或者进程对象 getcurrentprocess getcurrentthread 句柄属于伪句柄，不创建新的，不能用来关闭对象

Getprocesstimes getthreadtimes 查询进程线程id :getcurrentprocessid getcurrentthreadid

伪句柄变实句柄 duplicatehandle

7.线程调度，优先级、亲缘性 上下文结构context 上下文切换 20ms调度一次

抢占式 可以自行控制挂起

暂停：createprocess thread时就直接挂起 恢复 resumeThread 挂起suspendThread()

暂停或者恢复进程（所有线程） 1、利用父进程必须使用调试程序 waitfordebug 2、快照遍历

3、睡眠Sleep() 0 ,INFINITY

4、转换到另外的线程 SwitchToThread

5、线程运行时间 Gettickcount 这个函数本身就可能被中断

Getthreadtimes 返回创建时间 退出时间 内核时间 用户时间 获取所有线程GetProcessTimes 6、线程环境 CONTEXT 寄存器和栈 指令指针 堆栈指针 标志 返回地址 整数 浮点寄存器 段寄存器 调试 扩展 GETThreadContext 需要先挂起

只能返回用户环境 内核环境无法获取

获取寄存器后可以写内联汇编，前提是线程在挂起 也可以setthreadcontext

设置contextflags避免访问错误 7、优先级 0-31（最高） 优先调度31 高优先线程使用大量时间导致低优先无法运行 0页线程

进程优先级类 实时（抢先于操作系统） 高 高于正常 正常 低于正常 空闲

Explorer优先级高，线程不做事，不占时间

相对线程优先级 CreateProcess传入优先级类 改变优先级 get/set priority class

设置线程相对优线级 setthreadpriority

动态提高优先级 IO事件会被系统临时提高 可以取消setProcessPriorityBoost

前台进程，系统自动调整优先级 10、亲缘性 控制哪个CPU进行哪个进程

八、用户线程同步 （不用内核对象）

另一个线程需要了解情况 1、访问共享资源不破坏 2、线程完成任务需要通知另外线程

1、原子访问 不允许其他线程访问同一资源 全局变量 文件io 需要把时间错开

原子操作递增InterlockedExchageAdd() 修改变量递增(互锁函数)，硬件信号，防止cpu访问同一个内存地址

循环锁

InterlockedExchage InterlockedExchagePointer

高速缓存行

2、高级线程同步 互锁只能处理简单数据（循环锁控制处理器，很费资源）

使用同步变量

Volicate

关键代码段EnterCriticalSection(cs) LeaveCriticalSection() 其中传入了一个CRITICAL\_SECTION结构 不用时删除DeleteCriticalSection()

关键代码的循环锁

九、线程和内核对象同步 （慢，需要转到内核方式）

线程结束时变已通知

线程等待对象变为**已通知**

有通知的对象 进程线程 作业 事件 文件修改 定时器 文件 信标 控制台输入 互斥

已通知 未通知状态 线程等待任何对象通知WaitForSingleObject（对象，INFINITE），等待中线程挂起，返回值WAIT\_OBJECT\_0 WAIT\_TIMEOUT WAIT\_FAILED（获取上次错误）

参数可以是任何对象

Waitformultipleobject 可以看多个内核对象的状态（等一个或等所有）

Waitfor 函数返回时还可能修改内核对象的状态，比如把已通知再改回未通知

多线程等待一个对象的时候:公平。

事件对象 使用计数（所有对象都有） 自动重置还是人工重置

创建事件createevent 有个句柄，可以duplicatehandle 或者openevent

Setevent 设置为通知 resetevemt 重置为未通知

Waitforobject(event,)

Pulseevent 变已通知，然后很快变为未通知 closehandle

等待定时器CreateWaitableTimer() OpenWaitableTimer 自动重置还是人工重置

SetWaitableTimer 设置通知时间，初次时间，报时间隔 使用SYSTEMTIME 系统时间结构

取消定时器

Filetimetolocalfiletime filetimetosystemtime 获取日期getdateformate 获取时间 gettimeformate

异步过程调用排队（APC）SetWaitableTimer传入APC函数指针

和用户定时器Settimer的区别:线程安全，多线程

信标对象: 包含最大资源量 当前资源数量 资源大于0发出信标信号

CreateSemphore OpenSemphore Release

互斥对象 保证对资源的访问权 互斥对象的线程id（被谁拥有） 拥有的次数

CreateMutex OpenMutex 等待通知

线程释放互斥需要比较是否匹配线程 id

异步设备IO

WaitforInputIdle

等待自己的消息MsgWaitForMultipleObjects

**等待调试事件 WaitforDebugEvent**

发出通知并且等待另一个对象，SignalObjectAndWait (通知一个对象的同时等待另一个)

多个线程读，一个线程写

十一、线程池的使用

异步函数调用 间隔调用函数 已通知时调用函数 异步io完成时调用函数 （回调）

队列

QueueUsetrWorkItem 自动创建线程 调用传入的函数

定时器队列 CreateTimerQueue CreateTimerQueueTimer 队列中创建定时器 放入Apc

删除定时器DeleteTimerQueueTimer 改变定时器时间ChangeTimerQueueTimer 删除队列DeleteTimerQueue

等待通知 RegisterWaitForSingleObject（） 排队 超时

等待异步io BindIoCompletionCallBack

十二、纤程

线程转纤程 ConvertThreadTiFiber 线程上创建更多纤程 CreateFiber 内存栈，执行环境

单线程上一次只能执行一个纤程 纤程切换 SwithchToFiber 删除纤程DeleteFiber

获取当前纤程环境GetCurrentFiber GetFiberData

十三、内存结构

虚拟地址空间 4G ffffffff 32位最大4g寻址 系统空间是隐藏的无法访问，剩下的分配给应用

分区:null指针区 64k 禁止进入区 共享内存 用户方式（exe和dll） 文件（mmf，仅98） 内核方式 （系统代码，**访问违规**）

分配空间VitualAlloc (称为保留) 64k为分配粒度 区域大小是页面大小倍数4k 进程环境块FEB 线程环境块TEB 释放空间 VitualFree

提交物理存储器

访问:如果数据在ram中，直接从映射地址读取物理地址的数据

如果不在ram中（未读入或者物理内存被其他进程数据占用），页面失效 寻找一个空页，映射过去，没有空页就交换一些页出来（使用页面文件）

实际上数据不是都在页文件中，复制加载会非常慢 而是在硬盘的exe，只有需要时才交换到页

页面保护属性 读写违规 写违规 执行违规

多个进程共享单数据块 CopyOnWrite保护，**写时复制**

停用页面缓存 写入合并 PAGE\_GUARD 写入时发送通知

空闲（未保留） 私有（页文件） 映像（exe dll 写时复制） 映射文件

数据对齐：需要2的幂 EFLAGS AC标志 当AC为1，访问未对齐数据。Int 17h中断（但是window中是不会改ac的） seterrormode(允许未对齐异常)

十四、虚拟内存

1、系统信息GetSystemInfo(传一个指针) 2、虚拟内存状态GlobalMemoryStatus() 传递结构指针 3、地址空间状态VitualQuery 虚拟内存表 查询另一个进程VitualQueryEx MEMORY\_BASE\_INFORMATION结构

十五，使用虚拟内存 虚拟内存 内存映射文件 内存堆栈

1. 保留区域VitualAlloc（） 返回NULL 调用失败

MEM\_RESERVE 为区域赋予一个**保护属性 只读**

2、提交存储器 ： 再次调用VitualAlloc 传入MEM\_COMMIT标志 也可以传入两个标志同时保留和提交3、提交时机 始终提交 VitualQuery检查提交状态 自己保留记录 Seh的处理程序 4、回收内存 VitualFree 自己设计标志 或者检验方式来回收

5、**改变保护属性** VitualProtect 6、清除存储器 VitualAlloc MEM\_RESET

7、地址窗口扩展 大于32位 AWE 访问ram大于进程空间AllocateUserPhisicalPages

MapUserPhysicalPages

十六、 线程堆栈 默认1M可修改 链接/STACK选项 createThread可修改

Excption\_stack\_overflow 堆栈溢出异常，seh异常处理程序 访问违规异常

堆栈开始64k 溢出条件 最后64k下溢条件

堆栈检查函数

十七、内存映射文件

1. 加载exe和dll 2、访问磁盘数据，避免Io和缓存 3、共享数据

1、createProcess 创建虚拟空间 保留地址 基地址400000可链接时重载/BASE

文件映射对象 新进程对象 线程对象 使用内存映射文件共享代码和数据

为了防止exe文件（因为是内存映射）本身被改变，用了**写时拷贝**，创建新内存块，改变写到新内存块 重复打开的进程exe文件互不干扰

进程共享数据，节 :变量放入自己的节中 编译 /SECTION 共享节

2、内存映射文件 打开文件内核对象 创建文件映射内核对象 CreateFile CreateFileMapping 读/写 无缓存 pe 保留地址空间 MapViewOfFile()

撤销内存映像 UnmapViewOfFile 强制修改写入磁盘 FlushViewOfFile（全部或部分）

Closehandle

内存映像处理超大文件 如8G

可以映射同一文件 数据段 的多个视图，所有视图会同步更新

MapViewOfFileEx 设定基地址

进程间共享数据 多个进程映射同一段 文件视图

页文件的内存映射文件 不需要已有的file 实现进程数据共享 CreateFilleMapping，INVALID\_HANDLE\_VALIE （创建来自页文件的文件映像）

稀疏文件提交 CreateFileMapping SEC\_RESERVE VitualAlloc 创建稀疏文件 稀疏映射

十八、 堆栈（这里其实是指堆heap）

进程默认堆栈 获取默认堆栈句柄 GetProcessHeap

1. 辅助堆栈

保护组件 内存管理 本地访问 减少线程同步开销 迅速释放

创建HeapCreate 分配内存块HeapAlloc 成功分配返回地址 改变大小HeapReAlloc

内存实际大小HeapSize 释放内存HeapFree 撤销堆栈HeapDestory

C++中使用 malloc delete new

3、帮助函数 快照 Heap32First Next Heap32ListFirst Heap32ListNext

获取进程堆 GetProcessHeaps 返回句柄数组 验证堆完整性 HeapVallidate 堆合并空闲块HeapCompact

堆锁 HeapLock HeapUnlock 遍历堆HeapWalk PROCESS\_HEAP\_ENTRY结构

十九、dll user32.dll用户界面 窗口消息 kernel32.dll内存管理、进程和线程 gdi32.dll显示文本画图 对象安全 窗口控件

Dll可以内存共享 c++运行时库 dll可以包含资源 本地化 使用dll挂钩

Dll链接 隐式：编译时链接，有符号，函数名不变形 显式Loaddll

Dll提供函数 为了避免dll文件映射被修改，使用了**写时复制**

运行时库 输入函数和变量 输出函数和变量 **dll模块能包含其他dll的输出函数和变量 或者类？**

创建dll模块 \_\_declspec(dllexport) 函数名或者序号

.def文件或者#pragma comment(linker,”/exprot:新函数名=原函数名”)

搜索dll顺序:exe文件夹、进程当前目录、系统目录。Windows目录.path目录 找不到dll会弹警告

加载：确定输入的引用，查看输入表，然后在dll输出表找，找到rva，然后算出虚拟空间地址保存到输入表

1. 显式加载 符号链接 LoadLiberary

加载错误 获取上次错误 DON\_T\_RESOLVE\_DLL\_PEFERENCE 不调用dllmain

只包括资源的dll 卸载dll FreeLiberary FreeLiberaryAndExitThread 立刻卸载线程，到kernel32.dll

GetProcAddress 获取函数地址 参数是函数名（被重编码过的，或者序号）

Dllmain不是必须的

通知 dll\_process\_attach detach thread\_attach

C++运行期库

延迟加载dll 链接开关:/delayload:a.dll

函数转发器 转发到另一个dll的函数 比如调用kernel32.dll自动转发给ntdll.dll

改变模块位置 dll首选基地址 1000000 多个dll 改变模块地址 实际映射和首选地址的差 与机器指令地址相加

移位节

二十一、线程本地存储器 TLS 位标志 是个数组

动态tls TLSAlloc 找空标志 放入值 TlsSetValue 获取值TlsGetValue 释放TlsFree

静态Tls不需要函数 \_\_declspec(thread)声明线程变量 放入.tls节

二十二、插入dll和挂接api

把dll插入另一个进程

为其他进程窗口创建子类 setwindowlongptr 窗口过程指向自己的

注册表插入

Windows挂钩

Setwindowshookex 给进程安装消息钩子 findwindow

Getwindowthreadprocessid 发送消息postthreadmessage

远程线程 在目标线程创建新线程 CreateRemoteThread 让该线程调用LoadLiberrary 直接用loadliberary代替 threadfunc

分配另一个进程内存VirtualAllocEx 读写其他进程内存 ReadProcessMemory WriteProcessMemory

劫持dll 创建自己的同名dll，中途转发一道

调试操作

挂接windowsapi jmp到自己的内存再返回

修改输入表 改变输入表的函数地址就行了

内存映射文件插入iama

二十三、SEH 结构化异常处理 异常出现 异常的信息 异常的回调处理函数

硬件异常 软件异常 try except 异常过滤器error

GetExceptionCode() 异常类别 读写异常 页故障 堆栈异常 线程异常

非法操作 无效指令

调试异常：断点 单步陷阱 无效句柄

除0 异常 超范围异常 浮点异常

错误代码

获取异常信息GetExceptionInformation 三个结构 异常记录 线程环境 异常指针

异常记录：代码，另一个异常，cpu指令地址 异常附加信息

软件异常：强制引发， 用try 处理

二十五、未处理异常 UnhandledExceptionFilter 调用调试或者退出进

可以关掉异常提示框seterrormode

对异常进行处理 try

包装所有线程函数（设置顶层异常处理处理）S etUnhandledExceptionFilter

6、c++异常处理 本身就生成了对SEH的调用（RaiseException）

二十六、窗口 窗口消息 线程的消息队列

PostMessage postthreadmessage getwindowsthreadprocessid

发送退出消息 postquitmessage

Sendmessage() 同步，必须等待消息处理

Sendmessagetimeout sendmessagecallback sendnotifymessage replymessage

确认是线程内还是线程间 insendmessage

查询队列状态getqueuestatus 获取消息getmessage peekmessage

利用消息发送数据 进程间消息

6、窗口过程的字符 ansi unicode iswindowunicode() registerclassw

二十七、硬件输入模型 局部输入状态

原始输入线程RIT 系统硬件输入队列SHIQ

输入状态

焦点，光标 setfocus setactivewindow getactivewindow 注意调用函数的线程要和创建线程一致

BringWindowToTop

SetForegroundWindow

确定是否按下键 GetAsyncKeyStatus 光标剪切ClipCursor

ShowCursor SetCursor

鼠标捕获 SetCapture ReleaseCapture

挂接到局部输入状态 AttachThreadInput 多个线程共享虚拟输入队列

硬件异常 软件异常