# TsKsp.sys分析报告

该驱动为qq管家函数过滤驱动，跟TsFltMgr.sys搭配，TsFltMgr主要完成KiFastCallEntry的挂钩和提供设置过滤函数的接口，而该驱动主要用于配置拦截规则，用前者提供的接口设置真正的过滤函数，并在过滤函数中进行规则匹配以决定拦截行为。并提供一系列接口函数和控制码用于控制规则和内部数据。设备名[\Device\TSKSP](file:///\\Device\\TSKSP)，符号链接名[\DosDevices\TSKSP](file:///\\\\DosDevices\\\\TSKSP) ，加密手段：Rabbit算法、MD5算法。通过InlineHook KifastCallEntry实现挂钩。

目录

[TsKsp.sys分析报告 1](#_Toc434423723)

[一、 驱动入口DriverEntry 2](#_Toc434423724)

[1.1 监控模型 2](#_Toc434423725)

[1.2 派遣例程 2](#_Toc434423726)

[1.3 监控函数 3](#_Toc434423727)

[1.4重要回调的挂钩 10](#_Toc434423728)

[1.5 一些用到的数据 10](#_Toc434423729)

[1.6 OBJECT\_TYPE\_INITIALIZER 挂钩 11](#_Toc434423730)

[1.7 PROCINFO结构 12](#_Toc434423731)

[二、 控制信息 14](#_Toc434423732)

[2.1 ACL访问控制列表 14](#_Toc434423733)

[2.2 匹配树 14](#_Toc434423734)

[2.3 全局开关DriverSwitch 15](#_Toc434423735)

[2.4 自保开关影响的函数和功能 17](#_Toc434423736)

[2.5 规则判断 18](#_Toc434423737)

[2.6 黑白名单g\_StorageList 19](#_Toc434423738)

[2.7 上抛消息结构 24](#_Toc434423739)

[2.8 其他 25](#_Toc434423740)

[三、 接口和控制码 27](#_Toc434423741)

[3.1 导出接口 27](#_Toc434423742)

[3.2 控制码 27](#_Toc434423743)

[四、 基础库 33](#_Toc434423744)

[4.1 由进程Id获取文件对象 33](#_Toc434423745)

[4.2 由线程句柄获取进程对象 36](#_Toc434423746)

[4.3 由线程对象获取进程Id 36](#_Toc434423747)

[4.4 由Ntfs文件索引号获取文件对象 37](#_Toc434423748)

[4.5 长度反汇编引擎 39](#_Toc434423749)

## 驱动入口DriverEntry

* 获取TsFltMgr接口，初始化注册表信息、规则、操作系统版本等信息
* 创建[\Device\TSSysKit](file:///\\Device\\TSSysKit)设备和[\DosDevices\TSSysKit](file:///\\DosDevices\\TSSysKit)符号链接
* 初始化接口
* 注册IRP\_MJ\_CREATE、IRP\_MJ\_CLOSE、IRP\_MJ\_DEVICE\_CONTROL、IRP\_MJ\_SHUTDOWN派遣例程
* 保存Ntfs和Fastfat派遣函数
* 设置保护关键目录和注册表项
* 挂钩各个过滤函数
* 挂钩KeUserModeCallback和重要回调
* 开自保

### 1.1 监控模型

用户操作

TsFltMgr

过滤

拦截/放行

QMHIPSService

下发规则/ACL

TsKsp

下发黑白名单

上抛

返回结果

获取开关

设置开关

### 1.2 派遣例程

IRP\_MJ\_CREATE

检查驱动加载者是否有Ts签名

IRP\_MJ\_SHUTDOWN

设置\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\TSCPM\目录的LastShutdownFlag和LastShutdownTime

清空\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN\QQDisabled下的值

### 1.3 监控函数

Ts进程：位于Q管目录或签名为Q管文件签名的进程，按规则判断：上抛给主防主防进程判断；默认情况下若发起者为Ts进程的都放过

使用AddPrevFilter接口设置的过滤函数有：

NtAllocateVirtualMemory

若发起者为非TS进程而目标为TS进程，则设置PostFilter

若发起者为非TS进程而目标为非TS进程，则放行

若发起者为非TS进程而目标为未知进程，则查询虚拟内存监视链表，对比进程路径，按是否Ts进程分别处理

在PostFilter中记录得到的地址信息

NtAlpcCreatePort

若发起者为非Ts进程，且不是csrss, services, smss, svchost, lsass, lsm，且操作为修改DNS解析器，则上抛主防根据规则判断

NtAlpcSendWaitReceivePort

若发起者为非lsass非csrss非Ts进程，修改SAM信息，则上抛主防根据规则判断

若发起者为非Ts进程，触发csrss进程结束/winlogon关机，则上抛主防根据规则判断

若发起者为非Ts进程，操作驱动服务端口ntsvcs，则按服务号处理：

RcloseServiceHandle 执行功能，并将服务从监视列表中移除

RdeleteService, RsetServiceObjectSecurity, RchangeServiceConfigW,RchangeServiceConfigA,RChangeServiceConfig2A,

RChangeServiceConfig2W, RcontrolService,RcontrolServiceExA,RcontrolServiceExW 如果目标服务为QQPCRTP, TSKSP, TsFltMgr, QQSysMon, TSSysKit, TSSysFix则拒绝，否则执行功能

RopenServiceW,RopenServiceA 执行功能并上报主防

RcreateServiceW,RCreateServiceA 查注册表匹配树并根据结果选择执行功能且加入监控列表或上抛主防根据规则判断

RstartServiceW,RStartServiceA 查注册表匹配树，上抛主防加载驱动事件，并根据结果放行或不执行，从监控列表中删除

NtAssignProcessToJobObject

若发起者为非TS进程而目标为TS进程，则拒绝

NtCreateFile

穿透实现NtCreateFile

NtCreateMutant

添加PostFilter，在PostFilter中，若发起这为非Ts进程且目标Object处于g\_StorageList[14]中则上抛主防判断

NtCreateKey

若发起者为非主防进程，则在注册表键值匹配ACL树(RegMonTree)中查找，并根据结果上抛主防根据规则判断

NtCreateProcessEx

添加PostFilter，在PostFilter中，将进程添加到进程监控链表并上报给主防

NtCreatePort

若发起者为非Ts进程，且不是csrss, services, smss, svchost, lsass, lsm，且操作为修改DNS解析器，则上抛主防根据规则判断

NtCreateSection

满足DesiredAccess=SECTION\_ALL\_ACCESS,SectionPageProtection=PAGE\_EXECUTE,AllocationAttributes=SEC\_IMAGE时：

设置PostFilter；若目标文件为Ts文件，则检查应用层回溯栈，如果存在CreateProcessInternalW，则在启动参数增加CREATE\_PRESERVE\_CODE\_AUTHZ\_LEVEL位

NtCreateThread

若发起者为非主防进程，且该线程不是进程第一个线程，且目标进程为Ts进程，则拒绝

若发起者为非主防进程，且该线程不是进程第一个线程，且目标进程为非Ts进程，则上抛主防根据规则判断

NtCreateThreadEx

若发起者为非主防进程，且该线程不是进程第一个线程，且目标进程为Ts进程，则拒绝

若发起者为非主防进程，且该线程不是进程第一个线程，且目标进程为非Ts进程，则上抛主防根据规则判断

NtCreateSymbolicLinkObject

若目标对象为\Device\PhysicalMemory则拒绝

否则查询ACL表，若不符合则记录最近创建的4个符号名，若符合则上抛主防根据规则判断

NtCreateUserProcess

若目标文件为Ts文件，则检查应用层回溯栈，如果存在CreateProcessInternalW，则在启动参数增加CREATE\_PRESERVE\_CODE\_AUTHZ\_LEVEL位

NtDeleteKey

若目标在g\_StorageList[9]中则拒绝

若目标在注册表监控树中，若符合则上抛主防根据规则判断

NtDeleteValueKey

若目标在g\_StorageList[9]中则拒绝

若目标在注册表监控树中，若符合则上抛主防根据规则判断

NtDeviceIoControlFile

IoControlCode=0x8FFF23C8或0x8FFF23CC，目标驱动为\Driver\NDProxy，则上抛主防根据规则判断 IoControlCode=0x2A0000(IOCTL\_SWENUM\_INSTALL\_INTERFACE)则将该服务项添加到监视

IoControlCode=0x980C8(FSCTL\_SET\_ZERO\_DATA)

IoControlCode=0x2D1400(IOCTL\_STORAGE\_QUERY\_PROPERTY), 0x700A0(IOCTL\_DISK\_GET\_DRIVE\_GEOMETRY\_EX), 0x170002(IOCTL\_NDIS\_QUERY\_GLOBAL\_STATS), 0x4D008(IOCTL\_SCSI\_MINIPORT), 0x900c0(FSCTL\_CREATE\_OR\_GET\_OBJECT\_ID), 0x90073(FSCTL\_GET\_RETRIEVAL\_POINTERS), 0x7c088(SMART\_RCV\_DRIVE\_DATA), 0x74080(SMART\_GET\_VERSION) 则上抛主防根据规则判断

IoControlCode=0xA8730154, 0xA8730010 则采用不同解密密钥解密出PEPROCESS地址，若对应进程路径文件在内存操作监视链表中则拒绝

NtDuplicateObject (发起者，源进程，目标进程互不相同)

若源进程和目标进程为本进程，且Options=DUPLICATE\_SAME\_ACCESS则放行

若发起者为TS, csrss, services, smss, svchost, lsass, lsm进程则放行

源进程为Ts进程，且”发起进程-源进程”对处于监视列表中则拦截，否则放行

源进程为非Ts进程，目标进程为Ts进程，且非QQPCSoftGame, QQPCSoftMgr, QQPCClinic, QQPCExternal，则检查”发起进程-目标进程”对是否处于监视列表中，若存在则拦截，否则放行

源进程为非Ts进程，目标进程为普通进程或QQPCSoftGame, QQPCSoftMgr, QQPCClinic, QQPCExternal：

源进程和目标进程不同，或Options=DUPLICATE\_SAME\_ACCESS：

源进程为发起进程，且源句柄类型为Process/Thread则拦截

源进程不同于发起进程，则用本进程作为目标进程执行函数得到目标句柄：

若执行成功且目标句柄类型为Process/Thread则拦截

若执行返回STATUS\_INSUFFICIENT\_RESOURCES且获取进程句柄数无效则拦截

其余情况放行

NtEnumerateValueKey

执行函数，若KeyValueInformationClass=KeyValueFullInformation且发起进程为explorer：

则只能枚举出子项\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN\QQDISABLED

NtFreeVirtualMemory

若发起进程为非Ts进程且目标进程为Ts进程则设置PostFilter，在PostFilter中吧释放的内存从监视数据中去除

NtFsControlFile

若发起进程为非Ts进程，非lsass, csrss，则监视FsControlCode=0x11C017决定是否上抛主防

NtGetNextThread

若发起进程为非主防进程且目标进程为Ts进程则拒绝执行

若发起进程为非主防进程且目标进程为非Ts进程则将访问权限限制为SYNCHRONIZE|THREAD\_QUERY\_INFORMATION|THREAD\_GET\_CONTEXT

NtGetNextProcess

设置PostFilter

NtLoadDriver

若发起进程为非Ts进程，则将驱动文件信息上抛主防判断

NtMakeTemporaryObject

若目标对象为Section类型，且对象名路径在\knowndlls\下则拒绝，否则放行

NtOpenFile

穿透实现NtOpenFile

NtOpenProcess

若全局开关“监视打开进程”开启，则设置PostFilter，重置AccessMask参数并执行函数，在PostFilter中将发起线程加入监控链表中

若全局开关“监视打开进程”关闭：

若发起进程为非Ts进程：

若目标进程为QQPCFileSafe.exe, QQPCSoftGame.exe, QQPCSoftMgr.exe, QQPCExternal.exe, QQPCClinic.exe则上抛主防判断

若发起进程为svchost且目标进程为QQPCRtp.exe则上抛主防判断，若不符合条件则修改AccessMask标志位

若发起进程为lsass且目标进程为QQPCTray.exe则上抛主防判断，若不符合条件则修改AccessMask标志位

若发起进程为service且目标进程为Ts进程则上抛主防判断，若不符合条件则修改AccessMask标志位

若不满足上述条件，且不在g\_StorageList[15]中，则上抛主防判断，否则修改AccessMask

若发起进程为Ts进程，或发起进程与目标进程相同，则修改AccessMask

NtOpenThread

若全局开关“监控打开线程”开启，则设置PostFilter，重置AccessMask参数并执行函数，在PostFilter中将发起线程加入监控链表中

若全局开关“监视打开进程”关闭：

若发起进程为非Ts进程，且目标进程为Ts进程，则根据发起进程权限判断是否修改AccessMask

NtOpenSection

若发起进程为非主防进程，且DesiredAccess有写权限，且目标对象为\device\physicalmemory则上抛主防根据规则判断

NtProtectVirtualMemory

若发起者为非TS进程而目标为TS进程，则上抛主防根据规则进行判断

NtQueueApcThread

若发起者为非TS进程而目标为TS进程，则拒绝

若发起者为非TS进程而目标为非TS进程，则查询ACL表上抛主防根据规则判断

NtQueueApcThreadEx

若发起者为非TS进程而目标为TS进程，则拒绝

若发起者为非TS进程而目标为非TS进程，则查询ACL表上抛主防根据规则判断

NtReplaceKey

若发起者为非Ts进程：

目标注册表键路径匹配g\_StorageList[9]，则拒绝

目标注册表键路径匹配SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN，则拒绝

其他情况由注册表监控树获取权限并上抛主防根据规则判断

NtRequestWaitReplyPort

若发起者为非lsass非csrss非Ts进程，修改SAM信息，则上抛主防根据规则判断

若发起者为非Ts进程，触发csrss进程结束/winlogon关机，则上抛主防根据规则判断

若发起者为非Ts进程，操作驱动服务端口ntsvcs，则按服务号处理：

RcloseServiceHandle 执行功能，并将服务从监视列表中移除

RdeleteService, RsetServiceObjectSecurity, RchangeServiceConfigW,RchangeServiceConfigA,RChangeServiceConfig2A,

RChangeServiceConfig2W, RcontrolService,RcontrolServiceExA,RcontrolServiceExW 如果目标服务为QQPCRTP, TSKSP, TsFltMgr, QQSysMon, TSSysKit, TSSysFix则拒绝，否则执行功能

RopenServiceW,RopenServiceA 执行功能并上报主防

RcreateServiceW,RCreateServiceA 查注册表匹配树并根据结果选择执行功能且加入监控列表或上抛主防根据规则判断

RstartServiceW,RStartServiceA 查注册表匹配树，上抛主防加载驱动事件，并根据结果放行或不执行，从监控列表中删除

NtRestoreKey

若发起者为非Ts进程：

目标注册表键路径匹配g\_StorageList[9]，则拒绝

目标注册表键路径匹配SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN，则拒绝

其他情况由注册表监控树获取权限并上抛主防根据规则判断

NtSetContextThread

若发起者为非TS进程而目标为TS进程，则查询ACL表并上抛主防根据规则进行判断

NtSetInformationFile

若发起者为主防进程则放过

若FileInformationClass=FileDispositionInformation：

检查目标文件权限并上抛主防根据规则判断

若FileInformationClass= FileRenameInformation：

检查源文件和目标文件权限并上抛主防根据规则判断

若FileInformationClass= FileLinkInformation：

检查目标文件权限并上抛主防根据规则判断

若FileInformationClass= FileEndOfFileInformation：

检查目标文件权限并上抛主防根据规则判断

若FileInformationClass= FileAllocationInformation：

检查目标文件权限并上抛主防根据规则判断

其他FileInformationClass放行

NtSetSecurityObject

若发起者为非Ts进程，目标对象为以下之一则拒绝，否则放过：

\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\TSSysKit

\REGISTRY\MACHINE\SYSTEM\ControlSet002\services\TSSysKit

\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\services\TSSysKit

\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\TSKSP

\REGISTRY\MACHINE\SYSTEM\ControlSet002\services\TSKSP

\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\services\TSKSP

\REGISTRY\MACHINE\SYSTEM\ControlSet001\services\QQPCRTP

\REGISTRY\MACHINE\SYSTEM\ControlSet002\services\QQPCRTP

\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\services\QQPCRTP

NtSetSystemInformation

若SystemInformationClass=SystemExtendServiceTableInformation：

若发起者为非Ts进程且驱动文件存在则上抛主防根据规则进行判断

若SystemInformationClass= SystemRegistryAppendStringInformation：

若发起者为非Ts进程且目标注册表路径匹配g\_StorageList[9]，若键值路径匹配\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\CONTROL\SESSION MANAGER下的PendingFileRenameOperations,或PendingFileRenameOperations2，则根据文件监控树中的重命名前文件和重命名后文件的访问权限上抛主防判断，否则根据注册表监控树上抛主防判断

NtSetValueKey

若发起者为非Ts进程，且目标对象路径匹配\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSKSP\*或\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\QQPCRTP，且目标注册表键名为Start(服务启动类型)，则上抛主防根据规则进行判断，根据结果选择是否执行

若发起者为非Ts进程，且目标进程不是service，若匹配g\_StorageList[9]则根据注册表监控树的访问权限上抛主防判断

若键值路径匹配\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\CONTROL\SESSION MANAGER下的PendingFileRenameOperations,或PendingFileRenameOperations2，则根据文件监控树中的重命名前文件和重命名后文件的访问权限上抛主防判断，否则根据注册表监控树上抛主防判断

上抛判断前，会做清理以下键值以反调试：

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPCTray.exe

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPCRTP.EXE

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPCUPDATE.EXE

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPCAddWidget.exe

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPCMgr\_tz\_Setup.exe

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPCMgr.exe

\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\QQPConfig.exe

NtSuspendThread

若发起者为非Ts进程且目标进程为Ts进程则上抛主防根据规则判断

若发起者为非Ts进程且目标进程为非Ts进程则放过

NtSystemDebugControl

若Command=SysDbgWriteVirtual或SysDbgWritePhysical则上抛主防根据规则判断

NtTerminateProcess

若发起者为非Ts进程且目标进程为Ts进程：

若目标进程权限不足则放过

若发起进程在g\_StorageList[11]中则上抛主防根据规则判断

若发起者为非Ts进程且目标进程为非Ts进程：

若目标进程id在g\_StorageList[1]中或权限不足则上抛主防根据规则判断

若目标进程不是taskmgr则放过，否则上抛主防根据规则判断

NtTerminateThread

若发起者为非TS进程而目标为TS进程，则拒绝

若发起者为非TS进程而目标为非TS进程，则查询ACL表上抛主防根据规则判断

NtUnmapViewOfSection

若发起者为非主防进程，且目标进程为Ts进程，则拒绝

若发起者为非主防进程，且目标进程为非Ts进程，则上抛主防根据规则判断

NtUserClipCursor

若发起者为非主防进程，且激活窗口属于Ts进程，则匹配预设进程名，若匹配则跳过执行，否则放行

NtUserGetAsyncKeyState

若发起者为普通进程，执行功能，若键’0’-‘9’,’A’-‘Z’, 数字键盘’0’~’9’被按下，则上抛主防根据规则重置结果

NtUserGetKeyboardState

若发起者为普通进程，执行功能，若键’0’-‘9’,’A’-‘Z’, 数字键盘’0’~’9’被按下，则上抛主防根据规则重置结果

NtUserGetKeyState

若发起者为普通进程，执行功能，若键’0’-‘9’,’A’-‘Z’, 数字键盘’0’~’9’被按下，则上抛主防根据规则重置结果

NtUserGetRawInputBuffer

若发起者为普通进程，执行功能，若键’0’-‘9’,’A’-‘Z’, 数字键盘’0’~’9’被按下，则上抛主防根据规则重置结果

NtUserGetRawInputData

若发起者为普通进程，执行功能，若键’0’-‘9’,’A’-‘Z’, 数字键盘’0’~’9’被按下，则上抛主防根据规则重置结果

NtUserMessageCall

若dwType=FNID\_SENDMESSAGECALLPROC, FNID\_SENDMESSAGE, FNID\_SENDNOTIFYMESSAGE, FNID\_SENDMESSAGECALLBACK则跳过执行

若发起进程为普通进程，且uMsg=WM\_GETTEXT或WM\_SETTEXT则执行函数，若获取的字符串在列表中则上报主防

NtUserPostMessage

若发起进程不属于smss, lsass, lsm则上报主防进程

捕获uMsg= WM\_KEYFIRST, WM\_KEYUP, WM\_SYSKEYDOWN, WM\_SYSKEYUP, WM\_LBUTTONDBLCLK, WM\_LBUTTONDOWN, WM\_LBUTTONUP, WM\_MBUTTONDBLCLK, WM\_MBUTTONDOWN, WM\_MBUTTONUP, WM\_RBUTTONDBLCLK, WM\_RBUTTONDOWN, WM\_RBUTTONUP, BM\_CLICK, WM\_COMMAND, WM\_NOTIFY

若uMsg=WM\_COMMAND，则监视指定控件通知码

若发起进程不同于目标窗口所属进程则决定是否上报主防

若发起进程为非主防进程，且目标线程不同于发起线程，则关注uMsg=WM\_KEYDOWN, WM\_LBUTTONDOWN, BM\_CLICK, WM\_CLOSE, WM\_SYSCOMMAND, SC\_CLOSE, WM\_SETREDRAW, WM\_SHOWWINDOW, WM\_NCDESTROY, WM\_DESTROY, WM\_SETTEXT, IE\_DOCOMMAND, IE\_GETCOMMAND, IE\_GETCOUNT, WM\_COMMAND

若uMsg=WM\_SYSCOMMAND或SC\_CLOSE且发起者为explorer则放行，否则跳过执行

NtUserPostThreadMessage

若发起进程不属于smss, lsass, lsm则上报主防进程

捕获uMsg= WM\_KEYFIRST, WM\_KEYUP, WM\_SYSKEYDOWN, WM\_SYSKEYUP, WM\_LBUTTONDBLCLK, WM\_LBUTTONDOWN, WM\_LBUTTONUP, WM\_MBUTTONDBLCLK, WM\_MBUTTONDOWN, WM\_MBUTTONUP, WM\_RBUTTONDBLCLK, WM\_RBUTTONDOWN, WM\_RBUTTONUP, BM\_CLICK, WM\_COMMAND, WM\_NOTIFY

若uMsg=WM\_COMMAND，则监视指定控件通知码

若发起进程不同于目标窗口所属进程则决定是否上报主防

若发起进程为非主防进程，且目标线程不同于发起线程，则关注uMsg=WM\_KEYDOWN, WM\_LBUTTONDOWN, BM\_CLICK, WM\_CLOSE, WM\_SYSCOMMAND, SC\_CLOSE, WM\_SETREDRAW, WM\_SHOWWINDOW, WM\_NCDESTROY, WM\_DESTROY, WM\_SETTEXT, IE\_DOCOMMAND, IE\_GETCOMMAND, IE\_GETCOUNT, WM\_COMMAND，若匹配则根据情况跳过执行

NtUserSendInput

若发起进程不属于smss, lsass, lsm则上报主防进程

若发起进程为非主防进程则跳过执行

NtUserSetImeInfoEx

先执行函数以获取文件名

若输入法序号不存在于\Registry\Machine\SYSTEM\CurrentControlSet\Control\Keyboard Layouts\或子项为空，且发起进程为非Ts进程，则根据注册表监控树上抛主防根据规则决定是否放行

若输入法文件不为msctfime.ime/ msctf.dll，则根据注册表监控树上抛主防根据规则决定是否放行，否则放行

执行KeUserModeCallback前对该函数做还原inline hook处理

NtUserSetWindowsHookEx

若发起者为非Ts进程，ThreadId不为NULL，且目标进程为Ts进程，则拒绝

若发起者为非Ts进程，ThreadId不为NULL，且目标进程为System进程，若ModuleName为shell32.dll, msctf.dll, ieframe.dll, mshtml.dll, dinput8.dll, browseui.dll则放过，否则上抛主防根据规则判断

若发起者为非Ts进程，ThreadId为NULL，且目标进程为Ts进程，则按全局键盘钩子上抛主防根据规则判断

放行WH\_KEYBOARD\_LL类hook

NtUserSetWindowLong

若nCmdShow为SW\_SHOWMINNOACTIVE或SW\_FORCEMINIMIZE且目标窗口属于QQ.exe则上报给主防

若发起进程为主防进程则放过

若目标窗口属于Ts进程则跳过执行

NtUserSetWindowPos

若nCmdShow为SW\_SHOWMINNOACTIVE或SW\_FORCEMINIMIZE且目标窗口属于QQ.exe则上报给主防

若发起进程为主防进程则放过

若目标窗口属于Ts进程则跳过执行

NtUserSetWinEventHook

若发起进程为Ts进程则放过

若dwflags不包含WINEVENT\_INCONTEXT则放过

若发起进程等同目标进程则放过

若目标进程为Ts进程，则上抛主防根据规则判断

NtUserShowWindow

若nCmdShow为SW\_SHOWMINNOACTIVE或SW\_FORCEMINIMIZE且目标窗口属于QQ.exe则上报给主防

若发起进程为主防进程则放过

若nCmdShow不为SW\_HIDE, SW\_SHOWMINIMIZED, SW\_MINIMIZE, SW\_SHOWMINNOACTIVE, SW\_FORCEMINIMIZE则放过

若目标窗口属于Ts进程则跳过执行

NtUserShowWindowAsync

若发起进程为主防进程则放过

若nCmdShow不为SW\_HIDE, SW\_SHOWMINIMIZED, SW\_MINIMIZE, SW\_SHOWMINNOACTIVE, SW\_FORCEMINIMIZE则放过

若目标窗口属于Ts进程则跳过执行

NtWriteVirtualMemory

若发起者为普通进程，若目标进程为Ts进程，则拒绝；若目标进程为普通进程，则在用户态回溯栈中判定是否由CreateProcess发起 ，并根据ACL表上抛主防根据规则决定是否放行

KeUserModeCallback

若ApiNumber=\_\_ClientLoadLibrary：

若发起者为主防进程，跳过执行

若目标文件和事先传入(IoCtlCode=0x22E0E8)的Ts dll文件匹配则跳过执行

若目标文件在g\_StorageList[13]中则跳过执行

若开启白名单且目标文件不在g\_StorageList[16]中则跳过执行

若目标文件在g\_StorageList[5]中且发起进程为Ts进程则跳过执行，否则放过

若ApiNumber== \_\_fnHkINLPKBDLLHOOKSTRUCT：

若发起者为普通进程，执行功能，若键’0’-‘9’,’A’-‘Z’, 数字键盘’0’~’9’被按下，则上抛主防根据规则重置结果

若ApiNumber== \_\_ClientImmLoadLayout

先执行函数以获取文件名

若输入法序号不存在于\Registry\Machine\SYSTEM\CurrentControlSet\Control\Keyboard Layouts\或子项为空，且发起进程为非Ts进程，则根据注册表监控树上抛主防根据规则决定是否放行

若输入法文件不为msctfime.ime/ msctf.dll，则根据注册表监控树上抛主防根据规则决定是否放行，否则放行

执行KeUserModeCallback前对该函数做还原inline hook处理

### 1.4重要回调的挂钩

PsSetCreateProcessNotifyRoutine

将新增加的进程信息加入(ProcInfoList)链表

若为创建进程：

清除关键Ts文件映像劫持，即\Registry\Machine\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options下的QQPCTray.exe, QQPCRTP.EXE, QQPCUPDATE.EXE, QQPCAddWidget.exe, QQPCMgr\_tz\_Setup.exe, QQPCMgr.exe, QQPConfig.exe

若为删除进程：

清除进程相关信息

若进程为主防进程，则清除和关闭消息通信，进行清理工作

上报进程退出

PsSetCreateProcessNotifyRoutineEx

若CreateInfo为空则获取父进程id后交给CreateProcessNotify处理

将新增加的进程信息加入(ProcExInfoList)链表

PsSetLoadImageNotifyRoutine

若为系统模块，若为TesSafe.sys则记录该模块信息

若为普通模块，加载进程为若为Ts进程，且和目标模块exe相同，则清空PEB结构中的ShimData数据

### 1.5 一些用到的数据

BuildNumber[21] => UNKNOWN, 2195, 2600, 3790, 6000, 6001, 6002, 7000|7600, 7601, 8102, 8250, 8400, 8432|8441, 8520, 9200, 9600, 9841, 9860, 9926, 10041, 10049

ApiName[15] => NtUserFindWindowEx, NtUserBuildHwndList, NtUserQueryWindow, NtUserGetForegroundWindow, NtUserSetParent, NtUserSetWindowLong, NtUserMoveWindow, NtUserSetWindowPos, NtUserSetWindowPlaceMent, NtUserShowWindow, NtUserShowWindowAsync, NtUserSendInput, NtUserMessageCall, NtUserPostMessage, NtUserPostThreadMessage

Index[BuildNumber][ApiName]=

0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,0x000,

0x170,0x12e,0x1d2,0x189,0x1fe,0x20d,0x1c1,0x20f,0x20e,0x218,0x219,0x1e1,0x1bc,0x1cb,0x1cc,

0x17a,0x138,0x1e3,0x194,0x211,0x220,0x1d1,0x222,0x221,0x22b,0x22c,0x1f6,0x1cc,0x1db,0x1dc,

0x179,0x137,0x1e1,0x193,0x20e,0x21c,0x1d0,0x21e,0x21d,0x227,0x228,0x1f4,0x1cb,0x1da,0x1db,

0x187,0x142,0x1f8,0x1a2,0x226,0x236,0x1e4,0x238,0x237,0x243,0x244,0x20d,0x1df,0x1f1,0x1f2,

0x187,0x142,0x1f8,0x1a2,0x226,0x236,0x1e4,0x238,0x237,0x243,0x244,0x20d,0x1df,0x1f1,0x1f2,

0x187,0x142,0x1f8,0x1a2,0x226,0x236,0x1e4,0x238,0x237,0x243,0x244,0x20d,0x1df,0x1f1,0x1f2,

0x18c,0x143,0x203,0x1a7,0x230,0x242,0x1ef,0x244,0x243,0x24f,0x250,0x218,0x1ea,0x1fc,0x1fd,

0x18c,0x143,0x203,0x1a7,0x230,0x242,0x1ef,0x244,0x243,0x24f,0x250,0x218,0x1ea,0x1fc,0x1fd,

0x1c7,0x166,0x1de,0x1aa,0x246,0x230,0x1f3,0x22e,0x22f,0x223,0x222,0x25e,0x1f8,0x1e6,0x1e5,

0x1c9,0x167,0x1e0,0x1ac,0x249,0x232,0x1f5,0x230,0x231,0x225,0x224,0x261,0x1fa,0x1e8,0x1e7,

0x1ca,0x168,0x1e1,0x1ad,0x24b,0x234,0x1f6,0x232,0x233,0x227,0x226,0x263,0x1fb,0x1e9,0x1e8,

0x1cb,0x168,0x1e2,0x1ae,0x24d,0x236,0x1f7,0x234,0x235,0x229,0x228,0x265,0x1fc,0x1ea,0x1e9,

0x1cc,0x168,0x1e3,0x1ae,0x24f,0x237,0x1f8,0x235,0x236,0x22a,0x229,0x267,0x1fd,0x1eb,0x1ea,

0x1cb,0x168,0x1e2,0x1ad,0x24e,0x236,0x1f7,0x234,0x235,0x229,0x228,0x266,0x1fc,0x1ea,0x1e9,

0x1cc,0x16a,0x1e3,0x1ae,0x251,0x239,0x1f9,0x237,0x238,0x22c,0x22b,0x269,0x1fe,0x1ec,0x1eb,

0x1ce,0x16b,0x1e5,0x1af,0x253,0x23b,0x1fb,0x239,0x23a,0x22e,0x22d,0x26b,0x200,0x1ee,0x1ed,

0x1d2,0x16f,0x1e9,0x1b3,0x259,0x23f,0x1ff,0x23d,0x23e,0x232,0x231,0x271,0x204,0x1f2,0x1f1,

0x1d2,0x16f,0x1e9,0x1b3,0x25a,0x240,0x1ff,0x23e,0x23f,0x233,0x232,0x272,0x204,0x1f2,0x1f1,

0x1d2,0x16f,0x1e9,0x1b3,0x25b,0x240,0x1ff,0x23e,0x23f,0x233,0x232,0x273,0x204,0x1f2,0x1f1,

0x1d3,0x16f,0x1ea,0x1b3,0x25c,0x241,0x200,0x23f,0x240,0x234,0x233,0x274,0x205,0x1f3,0x1f2,

\_\_ClientLoadLibrary在KeUserModeCallBack中的ApiNumber ImmLoadLayoutIndex[BuildNumber] =

-1, -1, 84, -1, -1, -1, -1, 82, 82, -1, -1, -1, -1, -1, 84, 88, -1, -1, -1, -1, -1

KeyboardLL在KeUserModeCallBack中的ApiNumber HookKeyboardLLIndex[BuildNumber] =

-1, -1, 45, -1, -1, -1, -1, 45, 45, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1

对象例程相对于对象类型结构偏移 ObjectProcedureOffset [BuildNumber] =

-1, -1, 140, 140, 88, 88, 88, 88, 88, -1, -1, -1, -1, -1, 88, 88, 88, 88, 88, 88, -1

ShimData成员相对PEB结构偏移

-1, -1, 488, -1, -1, -1, -1, 488, 488, -1, 488, 488, -1, -1, 488, 488, -1, -1, 0, 0, 0

KeUserModeCallBack的ImmLoadLayout功能号中模块路径相对InputBuffer偏移 ClientLoadLibraryNameOffset[BuildNumber] =0, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 28, 48, 28, 0

### 1.6 OBJECT\_TYPE\_INITIALIZER 挂钩

和TsSyskit中类似，只不过Tsksp中，采用的是Proxy~Hooker的方式，将Proxy替换OpenProcedure，为每种Procedure预留5个函数槽用于存放过滤函数，只要有一个失败则返回失败。Proxy结构数组我在IDA中标记为ObjectProceduresProxy，依次为Event, File, Process, Thread, Mutant, Section的Proxy结构：

Struct ObjectProcedureStruct

{

ULONG ObType;//标识对象类型

/\*

0 ExEventObjectType

1 PsProcessType

2 PsThreadType

3 IoFileObjectType

4 ExMutantObjectType

5 MmSectionObjectType

\*/

ULONG ProcIndex;//标识函数类型

/\*

0 DumpProcedure

1 OpenProcedure

2 CloseProcedure

3 DeleteProcedure

4 ParseProcedure

5 SecurityProcedure

6 QueryNameProcedure

7 OkayToCloseProcedure

\*/

ULONG Proxy[21];//存放代理函数地址，分21个操作系统版本

}

再用一个数组存储对应的过滤函数：ULONG ObjectProceduresD(ynamic)Filter[6][8][5]， 6对应ObType，8对应ProcIndex，5为函数槽个数，数组中的数据，是从静态结构模板中导入的，该静态模板结构沿用了ObjectProcedureStruct结构。TsKsp中最终只对6种对象类型的OpenProcedure做了挂钩。

代理函数逻辑：

NTSTATUS \_\_stdcall ProcedureProxy(OB\_OPEN\_REASON OpenReason, PEPROCESS Process, PVOID Object, ACCESS\_MASK GrantedAccess, ULONG HandleCount)

{

NTSTATUS status = STATUS\_SUCCESS;

For(int i=0;i<5;i++)

{

If(ObjectProceduresDFilter[ObType][ProcType][i])

Status = ObjectProceduresDFilter[ObType][ProcType][i](OpenReason,Process,Object,GrantedAccess,HandleCount);

If(!NT\_SUCCESS(status))

Break;

}

Return status;

}

### 1.7 PROCINFO结构

用于存储当前系统进程信息

Struct PROCINFO

{

HANDLE ProcessId;

ULONG ProcDir;//标志目录属性

ULONG ProcType;//标志进程类型

ULONG Access;//标志访问权限

WCHAR Path[260];

ULONG IsSignMatch;

ULONG Index;

}

ProcDir域：

“System”, “Idle” 4

Ts进程 2

普通进程 1

ProcType域：

“System” -3

“Idle” -2

普通进程 -1

“csrss” 0

“services” 1

“smss” 2

“explorer” 3

“winlogon” 4

“svchost” 5

“lsass” 6

“lsm” 7

“taskmgr” 8

Access域：(访问权限)

0x1 通信

0x2 进程线程创建打开，驱动加载，内存操作

0x4 结束进程

0x8 注册表创建打开设置

0x10 设置文件属性

0x100 窗口交互类

“System”,”Idle”,Ts进程 0x1FF

“svchost”,”csrss”,“lsm”,”lsass” 0x107

“smss” 0x10

“services” 0x8

普通进程 0x0

对重要进程访问控制权限的设定：

PROCINFO信息是在进程创建时构造的，对于重要系统进程会单独进行初始化(InitCriticalProcessList)

name=csrss.exe,access=0x107,index=0

name=services.exe,access=0x8,index=1

name=smss.exe,access=0x10,index=2

name=Explorer.exe,access=0x0,index=3

name=winlogon.exe,access=0x0,index=4

name=svchost.exe,access=0x107,index=5

name=lsass.exe,access=0x107,index=6

name=lsm.exe,access=0x107,index=7

name=taskmgr.exe,access=0x2,index=8

具体如TsKsp Log.txt进程结构信息所示

## 控制信息

### 2.1 ACL访问控制列表

struct ACLTable 进程相关ProcessAclTable

+00h BYTE\* arr BYTE AclTable [dimen3][dimen2][dimen1]

+04h int dimen1 源进程类型相关 PROCINFO->ProcDir

普通 1

同目录 2

"Idle" 4

"System" 4

+08h int dimen2 目标进程类型相关 PROCINFO->ProcDir

普通 1

同目录 2

"Idle" 4

"System" 4

+0Ch int dimen3

进程相关：

ZwTerminateProcess 3

ZwCreateThread 6

ZwCreateThreadEx 6

ZwTerminateThread 7

ZwQueueApcThread 9

ZwQueueApcThreadEx 9

ZwSetContextThread 10

ZwAllocateVirtualMemory 11

ZwFreeVirtualMemory 11

ZwWriteVirtualMemory 11

+10h bool inited

文件也有类似的访问控制表

### 2.2 匹配树

TsKsp中存在注册表权限匹配树，和文件权限匹配树，结构为树+链表：

struct TreeData

{

PWCHAR MatchString;

BOOLEAN HasVal;

ULONG\* Val1;

ULONG Val2;

TreeData\* Next;

};

struct MatchTree

{

PWCHAR MatchString;

MatchTree\* Left;

MatchTree\* Right;

TreeData\* Data;

BOOLEAN HasVal;

};

void TranverseData(TreeData\* data,int depth)

{

while(data)

{

for(int i=0;i<depth;i++)

DbgPrint("\t");

DbgPrint("access=%x %x\n",\*(ULONG\*)data->Val1,data->Val2);

if(data->MatchString)

{

for(int i=0;i<depth;i++)

DbgPrint("\t");

DbgPrint("-%ws",data->MatchString);

}

data=data->Next;

}

}

void TranverseTree(MatchTree\* node,int depth)

{

if(node)

{

if(node->MatchString)

{

for(int i=0;i<depth;i++)

DbgPrint("\t");

DbgPrint("+%ws\n",node->MatchString);

}

TranverseData(node->Data,depth+1);

TranverseTree(node->Left,depth+1);

TranverseTree(node->Right,depth);

}

}

Main()

{

MatchTree\* node=\*(MatchTree\*\*)(Base+0x2C908);

TranverseTree(node,0);  
}

得到结果见TsKspLog.txt 匹配树结构——注册表 和 匹配树结构——文件

### 2.3 全局开关DriverSwitch

全局监视开关，共有128bit，预留128个开关，具体标志位如下：

DriverSwitch[0] & 1 监视消息和Section

DriverSwitch[0] & 2 监视窗口控件消息

DriverSwitch[0] & 4 监视硬件IOCTL

DriverSwitch[0] & 8 是否在创建/打开文件时将读权限设置为读|删除

DriverSwitch[0] & 0x100 拦截LPC某消息

DriverSwitch[0] & 0x200 在进程操作中通过文件号打开NTFS分区文件

DriverSwitch[1] & 4 是否允许任务管理器结束进程

DriverSwitch[1] & 8 监视键盘状态

DriverSwitch[1] & 0x40 监视加载输入法文件

DriverSwitch[1] & 0x80 监视访问系统文件保护 sfc

DriverSwitch[2] & 0x100 监视Ole LPC

DriverSwitch[1] & 0x200 打开进程和注册表时是否允许发起者为普通进程

DriverSwitch[1] & 0x400 监视csrss process shutdown

DriverSwitch[1] & 0x1000 监视服务操作的进程是否为敏感进程

DriverSwitch[2] & 0x2000 监视操作虚拟内存目标进程为系统线程

DriverSwitch[1] & 0x4000 是否使用回调式进程虚拟内存记录监视虚拟内存;是否监视创建非系统线程

DriverSwitch[1] & 0x8000 监视对象操作的总开关，已包括Process, Thread, Mutant, Section, File, Event对象

DriverSwitch[1] & 0x20000 是否启用创建进程监视链表MonCreateProcessList

DriverSwitch[1] & 0x80000 监视内存映射

DriverSwitch[1] & 0x100000 监视窗口消息

DriverSwitch[1] & 0x200000 监视打开文件对象

DriverSwitch[1] & 0x800000 监视打开进程

DriverSwitch[1] & 0x1000000 监视NDIS 0Day Attack

DriverSwitch[1] & 0x2000000 是否打开监视注册表、进程、线程、文件等函数的总开关，包括

ZwCreateKey,ZwDeleteKey,ZwDeleteValueKey,ZwCreateThread,

ZwDeviceIoControlFile,ZwQueueApcThread,ZwSetInformationFile,ZwSetSecurityObject,ZwSetSystemInformation,

ZwCreateFile,ZwOpenFile,ZwSetValueKey,ZwSuspendThread,ZwSystemDebugControl,ZwTerminateProcess,

ZwTerminateThread,ZwDuplicateObject,ZwEnumerateValueKey,ZwOpenProcess,ZwOpenSection,ZwQueueApcThreadEx

DriverSwitch[1] & 0x8000000 设置打开进程的PostFilter

DriverSwitch[1] & 0x10000000 监视打开线程对象

DriverSwitch[1] & 0x20000000 允许从存储的进程链表获取签名信息

DriverSwitch[2] & 0x10 监视打开Event, Mutant对象

DriverSwitch[2] & 0x80 是否检查权限时使用PROCESSINFO的签名位IsSignMatch

DriverSwitch[2] & 0x100 检测Ole LPC通信

DriverSwitch[2] & 0x200 检测父进程创建时间逻辑

DriverSwitch[2] & 0x400 监视虚拟内存的创建、写入和释放

DriverSwitch[2] & 0x800 监视winlogon shutdown

DriverSwitch[2] & 0x1000 监视打开Mutant对象

DriverSwitch[2] & 0x2000 是否允许操作System进程虚拟内存

DriverSwitch[2] & 0x80000 监视操作SAM

DriverSwitch[2] & 0x8000000 监视操作栈区虚拟内存

DriverSwitch[2] & 0x40000000 监视用户态模块加载

MaskForDriverSwitch[14] 用于生成特定组合的DriverSwitch，见IoCtlCode == 0x22E410，数据：

0,0x22b8

1,0x21fc

2,0x2199

3,0x26b8

4,0x22b7

5,0x22b5

6,0x22b6

7,0x26b9

8,0x219a

9,0x22b3

10,0x26ba

11,0x26bd

12,0x26bb

13,0x26be

0xx22E410控制码接收4字节数据mask，根据mask生成对应DriverSwitch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bit31 | Bit28~30 | Bit27~0 | | |
|  | 若Bit28~30=2 | |
| Bit6~5 | Bit4~0 |
| 0关闭开关  1打开开关 | 0根据MaskForDriverSwitch设置DriverSwitch[0]  2普通设置DriverSwitch[0-3] | 若Bit28~30=0，则这里为MaskForDriverSwitch的一项 | | |
|  | DriverSwitch序号 | 要设置的Bit位 |

BOOL SwitchControl(ULONG mask)

{

int bit;

int val;

int index;

BOOL set = FALSE;

switch((mask >> 28) & 7)// 28~30 bit

{

case 0:

for(bit=0;bit<14;bit++)

{

if(MaskForDriverSwitch[bit] == mask & 0xFFFFFFF)

break;

}

if(bit<14)

{

set = TRUE;

if((mask & 0x80000000) == 0)

DriverSwitch[0] &= ~(1<<bit);

else

DriverSwitch[0] |= 1<<bit;

}

case 2:

val = mask & 0xFFFFFFF;

if(val <= 0x65)

{

set = TRUE;

index = val >> 5;

bit = val & 0x1F;

if((mask & 0x80000000) == 0)

DriverSwitch[index] &= ~(1<<bit);

else

DriverSwitch[index] |= 1<<bit;

}

}

}

### 2.4 自保开关影响的函数和功能

在DriverEntry中，将\REGISTRY\MACHINE\SYSTEM\CurrentControlSet\Services\QQSysMon\spvalue的值若为0或1，则开启自保。另外可以在Q管界面常规设置选择自保护以手动方式开关自保，对应的程序QQPConfig.exe每次启动时会从TsKsp读取自保位(控制码0x22E0A0)，设置完毕时发送给TsKsp(控制码0x22E070)

NtSetInformationFile

影响Class=FileDispositionInformation, FileRenameInformation, FileLinkInformation, FileEndOfFileInformation, FileAllocationInformation的判断

NtCreateSymbolicLinkObject

NtDeviceIoControlFile

影响IoCtlCode=FSCTL\_SET\_ZERO\_DATA的判断

NtSetSystemInformation

影响Class=SystemRegistryAppendStringInformation重启替换/删除注册表键的判断

影响Class=SystemExtendServiceTableInformation 加载可执行映像的判断

NtSetValueKey

影响重启替换/删除注册表键的判断

NtRestoreKey

NtOpenProcess

NtOpenThread

NtUser\*

NtRequestWaitReplyPort/NtAlpcSendWaitReceivePort

影响RChangeServiceConfig, RDeleteService, RSetServiceObjectSecurity,

NtAssignProcessToJobObject

NtCreateMutant

NtCreateThread\NtCreateThreadEx

NtDeleteKey

NtDeleteValueKey

NtSetValueKey

NtDuplicateObject

NtGetNextProcess

NtOpenProcess

NtOpenThread

NtProtectVirtualMemory

NtQueueApcThread\NtQueueApcThreadEx

NtSetContextThread

NtSetSecurityObject

NtSuspendThread

NtTerminateProcess

NtUnmapViewOfSection

NtWriteVirtualMemory

KeUserModeCallback

影响ApiNumber=ClientLoadLibrary，若加载Ts文件的判断

CreateProcessNofity/ CreateProcessNotifyEx

影响每次创建进程是否清除注册表调试键

影响每次创建进程是否清除注册表调试键

LoadImageNotify

影响加载用户态映像的判断

Object-OpenProcedure

影响Event, File, Mutant, Process, Thread 对象的过滤

LPC通信中访问\RPC Control\IcaApi

### 2.5 规则判断

1.获取规则组号

int \_\_stdcall GetRuleGroup(int mIndex, int nIndex, wchar\_t \*chname, wchar\_t \*enname) 根据传递的中英文字符串(说明，路径等)参数匹配出组号

mIndex 一级序号

0 进程线程

1 文件

3 其他

nIndex 二级序号 通过FuncTypeToNum转换

mIndex=0时，为进程/线程操作

3 TerminateProcess

4 CreateProcess

5 QueryProcess

6 CreateThread

7 TerminateThread

8 SuspendThread

9 QueueApcThread

10 SetContextThread

mIndex=2时，为注册表操作

0 NtCreateKey

2 NtSetValueKey

3 NtDeleteValueKey

4 NtDeleteKey

5 NtRestoreKey

6 默认

mIndex=3时，为文件操作

0 FileEndOfFileInformation/FileAllocationInformation

1 FileDispositionInformation

2 FileLinkInformation

3 FileRenameInformation

5 重启删除/重命名文件

6 默认

mIndex=4

取消进程钩子

mIndex=5

1 QueueApcThread

2 RChangeServiceConfig

3 RDeleteService

7 NtUserSendInput

8 HardwareIoctl

10 Ole LPC

chname 中文类型说明

enname 英文类型说明

2.将规则id, 源进程id, 目标进程id, 源进程路径, 目标进程路径等信息序列化(SerialData)到上抛消息结构中

3.设置事件等待主防读取并取得判断结果

实例：

kd> dd tsksp+29960

b2cbd960 00000012 8149b648 00000140 8149c000

b2cbd970 00000000 00000000 0000001d 81487508

见TsKsp Log.txt 规则判断

### 2.6 黑白名单g\_StorageList

共有17个，!list -t \_LIST\_ENTRY.Flink -x "dd @$extret+8 " tsksp+252A8+index\*0x30

每种类型的链表对应不同Index，都用同样的数据结构存储

Struct MonitorDataHead

{

ULONG Index;//功能序号

SPIN\_LOCK Lock;//用于同步

PVOID PData;//存储下面17种继承于LIST\_ENTRY的数据结构

ULONG OffsetToHead;//每种结构List成员相对于PData的偏移

ULONG DataSize;//每种结构数据大小

ULONG InsertRoutine;//插入元素例程

ULONG DeleteRoutine;//删除元素例程

ULONG FindRoutine;//查找元素例程

ULONG FindWrapperRoutine;//互斥查找元素例程

ULONG DeleteAllRoutine;//清空数据例程

ULONG CompareRoutine;//比较元素例程

}

Index=0 sizeof=528 结束进程名黑名单 关自保或进程文件为普通文件生效

+00 LIST\_ENTRY List

+08 WCHAR ProcessName[260]

Index=1 sizeof=12 结束进程ID黑名单 关自保或进程文件为普通文件生效

+00 LIST\_ENTRY List

+08 HANDLE ProcessId 进程Id

Index=2 sizeof=528 未知

+00 LIST\_ENTRY List

+08 WCHAR ProcessName[260] 进程名

Index=3 sizeof=12 未知

+00 LIST\_ENTRY List

+08 HANDLE ProcessId 进程Id

Index=4 sizeof=536 未知

+00 LIST\_ENTRY List

+08 WCHAR ProcessName[260] 进程名

+210 IsWild//是否为通配符

Index=5 sizeof=528 ClientLoadLibrary加载模块白名单 SelfProcInjectAllow

+00 LIST\_ENTRY List

+08 WCHAR ImagePath[260] 映像路径

C:\Program Files\Tencent\QQPCMgr\11.1.16892.209\QMForbiddenWinKey.dll

C:\WINDOWS\system32\mshtml.dll

C:\WINDOWS\system32\IEUI.dll

C:\WINDOWS\system32\ieframe.dll

C:\WINDOWS\system32\uxtheme.dll

C:\WINDOWS\system32\browseui.dll

Index=6 sizeof=528 未知

+00 LIST\_ENTRY List

+08 WCHAR [260]

Index=7 sizeof=1048 未知

+00 LIST\_ENTRY List

+08 WCHAR [260]

+210 WCHAR [260]

Index=8 sizeof=528 操作虚拟内存，创建进程，创建内存映射文件 目标黑名单

+00 LIST\_ENTRY List

+08 WCHAR FileName[260] 文件名

Index=9 sizeof=12 注册表操作黑名单

LIST\_ENTRY

WCHAR[260] KeyPathPattern 键路径模式

WCHAR[260] ValueNamePattern 键值模式

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSKSP\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\QQPCRTP\*, \*

\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN, QQPCTRAY

\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN\QQDISABLED, \*

\REGISTRY\USER\S-\*\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\RUN\QQDISABLED, \*

\REGISTRY\MACHINE\SOFTWARE\TENCENT\QQPCMGR\SYSTEMOPTIMIZE\DISABLED, \*

\REGISTRY\MACHINE\SOFTWARE\TENCENT\QQPCMGR\SYSTEMOPTIMIZE\DISABLEDSVC, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSDEFENSEBT\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\QQSYSMON\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TFSFLT\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSSYSKIT\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSFLTMGR\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSSK\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSNETMON\*, \*

\REGISTRY\USER\QMCONFIG\*, \*

\REGISTRY\MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\UNINSTALL\QQPCMGR\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\TSSYSFIX\*, \*

\REGISTRY\MACHINE\SYSTEM\\*CONTROLSET\*\SERVICES\ANTIRK\*, \*

Index=10 sizeof=536 文件

+00 LIST\_ENTRY List

+08 WCHAR FileName[260] 文件/路径名

+210 BOOL IsDir

+214

Index=11 sizeof=528 结束进程查询名单 CTSKspWrap::AddSelfProcTeminateQuery

+00 LIST\_ENTRY List

+08 WCHAR FilePath [260] 文件名

C:\Program Files\Tencent\QQPCMgr\11.1.16892.209

Index=12 sizeof=528 未知

+00 LIST\_ENTRY List

+08 WCHAR [260]

Index=13 sizeof=528 \_\_ClientLoadLibrary \_\_ClientLoadLibrary发起进程黑名单

+00 LIST\_ENTRY List

+08 FileName [260] 文件名

TASLogin.exe

Client.exe

SoapUI\_4\_6\_4.exe

QQPCLeakScan.exe

navicat.exe

NativeWeb.exe

phpstorm.exe

ugraf.exe

League of Legends.exe

dnf.exe

tgp\_daemon.exe

my.exe

pycharm.exe

bugreport.exe

QQPetBear.exe

TXPlatform.exe

xmind.exe

tencentdl.exe

\_INS5576.\_MP

Index=14 sizeof=536 打开对象目标黑名单 SyncObjProtect

LIST\_ENTRY List

POBJECT\_NAME\_INFORMATION ObjInfo 对象路径

WCHAR ObjectPath[260]

04CE0CB6-CDF3-4a4b-8B9D-292A455FAF5B

04CE0CB6-CDF3-4a4b-8B9D-292A455F

AF5B

QDOCTOR\_2

QDOCTOR\_1

QDOCTOR\_0

qpcmgr\10002\_0\_80

ENCENT\_QMTAV\_TSCAN\_HANG

Index=15 sizeof=1072 打开进程/复制句柄发起进程白名单 SelfProcAllow

LIST\_ENTRY List

ULONG Enable??

ULONG SourceFilePathLen

WCHAR[261] SourceProcessFilePath 源进程路径

ULONG Target FilePathLen

WCHAR[261] TargetProcessFilePath 目标进程路径  
 ACCESS\_MASK GrantedAccess

Index=16 528 ImageName \_\_ClientLoadLibrary发起进程白名单

+00 LIST\_ENTRY List

+08 FileName [260] 模块文件名

wuauclt.exe

MiniThunderPlatform.exe

RtxLite.exe

DriveTheLife.exe

115chrome.exe

adownloader.exe

2345Explorer.exe

MyIE9.exe

krbrowser.exe

Ruiying.exe

csbrowser.exe

114Web.exe

114IE.exe

WebGamegt.exe

Coral.exe

TangoWeb.exe

tango3.exe

TaoBrowser.exe

YY.exe

Android PC Suite.exe

wandoujia2.exe

DriverGenius.exe

DTLSoftManage.exe

sdDown.exe

VDisk.exe

klive.exe

Kanbox.exe

sedown.exe

Alibrowser.exe

BaiduHi.exe

LiveUpdate360.exe

flashgetmini.exe

flashget3.exe

idman.exe

Explorer.exe

QQBrowserExtHost.exe

QBDownloader.exe

FoxMail.exe

ieuser.exe

TTraveler.exe

rtx.exe

tm.exe

fetion.exe

msnmsgr.exe

outlook.exe

UDownSrv.exe

UDown.exe

WebThunder.exe

MiniThunder.exe

Thunder5.exe

ThunderMini.exe

DUTool.exe

RaySource.exe

peer.exe

Thunder.exe

ThunderService.exe

thunderplatform.exe

minimule.exe

emule.exe

QQDownload.exe

baidubrowser.exe

liebao.exe

QQBrowser.exe

360chrome.exe

360se.exe

webkit2webprocess.exe

safari.exe

huaer.exe

saayaa.exe

twchrome.exe

firefox.exe

thooe.exe

theworld.exe

myiq.exe

greenbrowser.exe

ybrowser.exe

115br.exe

opera.exe

chrome.exe

maxthon.exe

sogouexplorer.exe

### 2.7 上抛消息结构

分析出来的上抛数据类型如下：

Type Id

ProcessId 0 进程Id

ChildProcessId 1 子进程Id

SrcFilePath 4 源文件路径

ChildFilePath 5 子进程路径

RegPathName 6 注册表路径

RegKeyName 7 注册表键

RegData 8 注册表数据

ObjFilePath 9 目标文件路径

RuleId 10 规则id

ChName 11 中文描述

EnName 12 英文描述

?? 13

ProcessData 14

ChildData 15

ProcessIndex 17 进程序号

RegKeyType 18 注册表数据类型

RegDataSize 19 注册表数据大小

过滤函数中，根据条件进行筛选以后，遇到符合条件的消息，先根据特征获取消息序号，然后在GroupAccessList中查找进程对应该消息号的权限，之后根据捕获类型构造消息结构， SerialData函数将必要的数据按上述类型序列化到缓冲区，之后将消息添加进消息队列中，并设置写信号，等待主防进程读取，上抛过程可以是同步或者异步方式。

SENDMSG sizeof=0x1000

+000 ULONG Size//0x1000

+004 SENDMSG\* self

+008 struct REPLYMSG\* sizeof=0x1000

+000

+004 SENDMSG\* notifymsg;

+00C UCHAR[6] MsgTypeString;//”REPLY”

+018 PKEVENT Event

+01C HANDLE ThreadId

+00C UCHAR[7] MsgTypeString;//”NOTIFY”

+01C ULONG nIndex//事件类型标识

+020 ULONG pIndex//事件类型标识

+024 PKEVENT Event

+028 HANDLE ThreadId//待检测线程id

+02C ULONG 请求结果类型//=0 不请求

+030 BOOL 主防是否需要给出结果

+034 ULONG MsgSize//串行化后总长度

+038 PBYTE data[0xFC8] 串行化数据 格式“类型-大小-数据[]”

### 2.8 其他

读取ini文件

标记为和应用层api的名字相同(GetPrivateProfile\*)的函数，函数调用类型基本相同

GetPrivateProfileInt, 从Ini文件中获取整型值

GetPrivateProfileIntEncrypt从Ini文件中解密并获取整型值

GetPrivateProfileString从Ini文件中获取字符串

GetPrivateProfileStringEncrypt 从Ini文件中解密并获取字符串

主防QMHIPSService通信日志

可以打印出QMHIPSService与Tsksp.sys进行DeviceIoControl通信日志

在HKEY\_LOCAL\_MACHINE\Software\Tencent\QQPCMgr下新建EnableLogToView的Dword键，设置为0xFFFFFFFF，即可打开Magic debuge，xp下主防的日志在\Documents and Settings\All Users\Application Data\Tencent\QQPCMgr\TrojanLog\qqpcrtp\_qmhipsservice.log

对Q管签名的破解

Q管签名验证，存在于除TsFltMgr之外的所有驱动中(boot start)，代码已经在TsSysKit分析文中给出，过程如下：

1.读取PE，以IMAGE\_DOS\_HEADER->e\_res2[0-1] 的DWORD值作为文件偏移，取该处128字节，通常在文件尾

2.用这128字节密文解密出24字节信息：

+00 被加密数据的文件偏移，通常是代码段起始

+04 被加密数据的信息长度，通常到加密前的文件结束位置

+08 BYTE[16] 文件内容经过MD5变种算法加密出的16字节密钥

3.对被加密数据重新MD5校验，将结果和16字节密钥对比

由于算法可逆性未知，128字节密文不易从自己构造的24字节原始信息恢复，因此我的破解方法是取最小的具有Ts签名的文件尾项数据，插在PE开头，原先PE段相应移位，这样便可以通过Ts签名校验。代码为SignAsTsFile.cpp

影响到PROCINFO的IsSignMatch成员，该标志为Ts文件签名通过标志，由DriverSwitch开关控制，访问权限没有“Q管目录文件”级别高，影响到的函数有：

NtDuplicateObject 放行执行者

虚拟内存操作 保护目标

允许输入法注入

NtCreateThreadEx放行执行者

NtSuspendThread放行执行者

NtGetNextThread放行执行者

NtTerminateProcess 放行执行者

NtTerminateThread 放行执行者

NtQueueApcThread放行执行者

NtQueueApcThreadEx放行执行者

NtUserSetWinEventHook放行执行者

NtUserSetWinEventHookEx放行执行者 保护目标

SetSystemInformation 放行执行者

NtSetContextThread放行执行者

NRestoreKey放行执行者

NtSetValueKey 放行执行者

NtCreateThread保护目标

NtUnmapViewOfSection放行执行者

NtCreateSymbolicLinkObject放行执行者

OpenObject放行执行者

NtSetSecurityObject放行执行者

NtAlpcCreatePort 放行执行者

NtCreatePort 放行执行者

NtAssignProcessToJobObject放行执行者

NtLoadDriver放行执行者

## 接口和控制码

### 3.1 导出接口

ULONG\_PTR \_\_stdcall Interface(int Index)

{

Swtich(index)

{

Case 1:

Return GetProcInfoById;// 函数指针，用于通过进程Id拷贝PROCINFO结构，如前所述

Case 2:

Return &EnableUpThrow;//用于控制是否将驱动事件上抛给主防消息并按规则处理，影响大部分过滤函数

Case 3:

Return &SelfProtectSwitch;//自保开关，影响的过滤函数见“自保开关影响的函数和功能”一节

Case 4:

Return &MonitorSwitch;

/\*

ULONG[0]:进程,线程,钩子操作开关，影响的过滤函数有：NtTerminateProcess, NtCreateThread, NtTerminateThread, NtSetContextThread, NtCreateThreadEx, NtQueueApcThread, NtQueueApcThreadEx, NtUserSetWinEventHook, NtUserSetWindowsHookEx

ULONG[1]: 注册表操作开关，影响的过滤函数有：NtRestoreKey, NtSetValueKey, NtCreateKey, NtDeleteKey, NtDeleteValueKey, NtLoadDriver, NtSystemDebugControl, NtSetSystemInformation

ULONG[2]: 文件,设备操作，影响的过滤函数有：NtSetInformationFile, NtCreateSymbolinkObject, NtDeviceIoControlFile

\*/

Case 5:

Return &SetFileSwitch;//文件操作开关，若关闭则用Tsksp内置规则，否则交给文件过滤驱动分析逻辑

Case 6:

Return GetProcDir;//函数指针，用于通过进程Id获取ProcDir属性，如前所述

Default:

Return NULL;

}

}

### 3.2 控制码

0x22E004 挂钩KeUserModeCallBack并返回结果

Buffer= sizeof=4

+00 BOOL ret

0x22E008 未实现

0x22E010 增加结束进程名黑名单元素 g\_StorageList[0]

Buffer= sizeof=0x208

+00 WCHAR ProcessName[260]

0x22E014 删除结束进程名黑名单元素 g\_StorageList[0]

Buffer= sizeof=0x208

+00 WCHAR ProcessName[260]

0x22E01C 设置进程线程钩子操作开关MonitorSwitch 如前所述

Buffer= sizeof=0xC

+00 ULONG Data[3]

0x22E020 未知

0x22E028 添加结束进程ID黑名单元素

Buffer= sizeof=4

+00 HANDLE ProcessId

0x22E02C 删除结束进程ID黑名单元素

Buffer= sizeof=4

+00 HANDLE ProcessId

0x22E030 添加g\_StorageList[2]元素

0x22E034 删除g\_StorageList[2]元素

0x22E038 增加允许自注入名单g\_StorageList[5] CTSKspWrap::AddSelfProcInjectAllow

Buffer= sizeof=0x208

+00 WCHAR ImagePath[260]

0x22E03C 删除ClientLoadLibrary加载模块白名单元素

Buffer= sizeof=0x208

+00 WCHAR FilePath[260]

0x22E040 挂钩KeUserModeCallback

0x22E044 添加g\_StorageList[4]元素

0x22E048 删除g\_StorageList[4]元素

0x22E04C 添加g\_StorageList[6]元素

0x22E064 DriverEntry中若KeUserModeCallback挂钩失败，则IRP\_MJ\_ DEVICE\_CONTROL派遣例程只接受该控制码，用于获取挂钩信息

Buffer= sizeof=4

+00 ULONG //0挂钩成功 2挂钩失败

0x22E050 删除g\_StorageList[6]元素

0x22E054 添加g\_StorageList[7]元素

0x22E058 删除g\_StorageList[7]元素

0x22E05C 重置关闭自保状态

Buffer= sizeof=4

+00 ULONG 1重置 2关闭

0x22E060 设置检测发送消息开关SendInputSwitch CTSKspWrap::SetSendInputSwitch

Buffer= sizeof=4

+00 ULONG => SendInputSwitch

该标志位影响的过滤函数有：NtUserSendInput, NtUserMessageCall, NtUserPostmessage, NtUserPostThreadMessage

0x22E064

Buffer= sizeof=4

+00 ULONG status;// KeUserModeCallback挂钩结果

0x22E06C 添加g\_StorageList[12]元素

0x22E070 设置自保开关SelfProtectSwitch CTSKspWrap::SetSelfProtectState

Buffer= sizeof=4

+00 ULONG => SelfProtectSwitch

0x22E074 传入%SystemRoot%\system32\advapi32.dll CreateServiceA地址 CTSKspWrap::AddApiAddress

Buffer= sizeof=4

+00 ULONG Addr

0x22E078 传入%SystemRoot%\system32\advapi32.dll CreateServiceW地址 CTSKspWrap::AddApiAddress

Buffer= sizeof=4

+00 ULONG Addr

0x22E07C 传入%SystemRoot%\system32\rpcrt4.dll NdrClientCall2地址 CTSKspWrap::AddApiAddress

Buffer= sizeof=4

+00 ULONG Addr

0x22E080 增加自结束进程查询名单 CTSKspWrap::AddSelfProcTeminateQuery

Buffer= sizeof=0x208

+00 WCHAR FilePath[260]

0x22E084 增加注册表监控项 CTSKspWrap::AddRegMonitor

Buffer= sizeof=0x418

+000h WCHAR RegPath[260]

+208h WCHAR KeyName[260]

+410h HANDLE ProcessHandle

+414h DWORD Access

0x22E08C 根据id设置PROCINFO的ProcDir域

Buffer= sizeof=8

+0 HANDLE ProcessId

+4 ULONG ProcDir

0x22E090 CTSKspWrap::AddFileGroup

Buffer= sizeof=0x212

+00 ULONG Access

+08 WCHAR [260] 匹配路径

0x22E094 CTSKspWrap::AddProcPrivilege

0x22E09C 初始化驱动同步 CTSKspWrap::InitDriverSync

Buffer= sizeof=4

+00 ULONG => InitDriverSync

0x22E0A0 获取自保开关状态 CTSKspWrap::GetSelfProtectState

Buffer= sizeof=4

+00 ULONG Data <= SelfProtectSwitch

0x22E0C4 和驱动建立连接的过程中，发给驱动的用于通知主防驱动已经写完消息等待读取的同步信号

Buffer= sizeof=4

+00 PKSEMAPHORE MsgWriteLock

0x22E0C8 主防向驱动索取消息结构 CTSKspWrap::DriverGetMessage

Buffer= sizeof=0x1000

0x22E0CC 驱动返回消息 CTSKspWrap::DriverReplyMessage

0x22E0D0 通知驱动主防进程退出，做清理工作 CTSKspWrap::CloseDriverEvent

0x22E0D8 穿透创建服务加载驱动

buffer= sizeof=0x91C

+000 WCHAR ImagePath[260] 驱动文件路径

+208 DWORD Type 驱动注册表Type项

+20C DWORD Start 驱动注册表项Start类型

+210 DWORD flag (决定是否设置注册表Tag和Group信息)

+214 ？？？

+468 DWORD Tag 驱动注册表Tag项

+46C WCHAR DisplayName[300] 驱动注册表项DisplayName

+6C4 WCHAR ServiceName[300] 驱动服务名

0x22E0DC 下发要监控的线程Id给驱动，存储于ThreadIdSlot，该结构在创建/打开文件操作中生效

Buffer= sizeof=4

+00 HANDLE ThreadId

0x22E0E0 取消要监控的线程Id，修改ThreadIdSlot

Buffer= sizeof=4

+00 HANDLE ThreadId

0x22E0E4 重置全局访问控制表ACL =>AclTable 三维数组，详述见全局访问控制表章节

Buffer= sizeof=0x10

+00 ULONG dimen1//第一维大小

+04 ULONG dimen2//第二维大小

+08 ULONG dimen3//第三维大小

+0C PBYTE data//数据基址

0x22E0E8 添加\_\_ClientLoadLibrary发起进程黑名单元素 g\_StorageList[13] CTSKspWrap::SetIoControl

Buffer= sizeof=0x208

+00 FileName [260] 模块文件名

0x22E0EC 删除\_\_ClientLoadLibrary发起进程黑名单元素 g\_StorageList[13]

Buffer= sizeof=0x208

+00 FileName [260] 模块文件名

0x22E0F0 设置要监控的模块路径 CTSKspWrap::SetIoControl

Buffer= sizeof=0x208

+00 DllPath [260] 模块文件名

0x22E100 设置进程信息

Buffer= sizeof=0xC

+00 HANDLE ProcessId in

+04 ULONG ProcDir out

+08 ULONG out

0x22E104 是否开启验证父进程和子进程创建时间逻辑

Buffer= sizeof=4

+00 BOOLEAN VerifyTime//0不开启 1开启

0x22E108 添加\_\_ClientLoadLibrary发起进程白名单项g\_StorageList[16] CTSKspWrap::SetIoControl

Buffer= sizeof=0x208

+00 FileName [260] 模块文件名

0x22E10C 开启\_\_ClientLoadLibrary发起进程白名单项g\_StorageList[16]的验证

0x22E110 删除\_\_ClientLoadLibrary发起进程白名单项g\_StorageList[16] CTSKspWrap::SetIoControl

Buffer= sizeof=0x208

+00 FileName [260] 模块文件名

0x22E114 关闭\_\_ClientLoadLibrary发起进程白名单项g\_StorageList[16]的验证

0x22E400 增加监控条目 信息添加到数组，做第一次规则匹配 CTSKspWrap::AddMonitorItem

Buffer= sizeof>0x10

+00 USHORT cbSize

+02 USHORT 0/1

+04 USHORT 2

+06 USHORT GroupInfoSize//组信息结构大小

+08 USHORT Type //组类型 0进程线程 1文件 2?? 3其他

+10 UBYTE[] 组信息

0x22E404 增加”规则组进程访问权限”信息 信息添加到二级链表，做第二次规则匹配

Buffer= sizeof>0x10

+00 USHORT cbSize

+02 USHORT 0/1

+04 USHORT 2

+06 USHORT GroupInfoSize//进程相关的权限信息结构大小

+10 UBYTE[] 组信息

内部存储结构：

+00 LIST\_ENTRY ListEntry

+08 LIST\_ENTRY ChildListEntry

+00 LIST\_ENTRY ListEntry

+08 ULONG uMinRuleNum

+0C ULONG uMaxruleNum

+10 ULONG Access;//0放过 非0拦截

+10 HANDLE ProcessId

保存在内存中的结构为RuleGroupInfo[4] 每个结构由成员个数(=n)和基地址2个ULONG组成，每个基地址存放着n个下列子结构：

00 ULONG ruleid

04 ULONG mid

08 ULONG subid

0C ULONG val4

10 WCHAR\* matchfirst

14 WCHAR\* matchsecond

详细情况后面有详述

0x22E410 设置全局开关DriverSwitch DriverSwitch见全局开关章节 CTSKspWrap::SetDriverSwitch

Buffer= sizeof=4

+00 ULONG mask

0x22E414 监视操作窗口标题，将新标题字符串加入监视列表白名单

Buffer= sizeof>0

+00 WCHAR[??] 窗口标题

0x22E418 获取在NtRequestWaitReplyPort和NtAlpcSendWaitReceivePort通信中使用IWbemInterface通信的进程

Bufer= sizeof=8

+00 HANDLE ProcessId

+04 0

0x22E41C 增加打开对象目标黑名单项g\_StorageList[14]，见“黑白名单”一节 CTSKspWrap::AddSyncObjProtect

Buffer= sizeof=0x208

+00 WCHAR ObjectPath[260]

0x22E420 增加打开进程/复制句柄发起进程白名单项 g\_StorageList[15] ，见“黑白名单”一节 CTSKspWrap::AddSelfProcAllow

Buffer= sizeof=0x430

+000 LIST\_ENTRY List

+008 ULONG Enable??

+00C ULONG SourceFilePathLen

+010 WCHAR SourceProcessFilePath[261]

+21C ULONG TargetFilePathLen

+220 WCHAR TargetProcessFilePat[261]

+42C ACCESS\_MASK GrantedAccess

0x22E420 由进程Id获取进程加载序号，PROCESSINFO的Index域

Buffer= sizeof=8

+0 HANDLE ProcessId <=> ULONG Index

0x22E424 由进程Id获取进程文件全路径

Buffer= sizeof= in 4 out 0x104

+0 HANDLE ProcessId <=>

## 基础库

### 4.1 由进程Id获取文件对象

BOOLEAN GetSectionObjectOffset()

{

UCHAR Inst1[]={0x8B, 0xFF, 0x55, 0x8B, 0xEC, 0x8B, 0x45, 0x08, 0x8B, 0x80};

UCHAR Inst2[]={0x8B, 0x44, 0x24, 0x04, 0x8B, 0x80};

BOOLEAN result = FALSE;

ULONG\_PTR SectionBaseOffset = 0;

PsGetProcessSectionBaseAddress = MmGetSystemRoutineAddress(&uName);

if(PsGetProcessSectionBaseAddress && MmIsAddressValid(PsGetProcessSectionBaseAddress))

{

if(MmIsAddressValid((PVOID)((char\*)PsGetProcessSectionBaseAddress + 14)) &&

RtlCompareMemory((PVOID)Inst1, PsGetProcessSectionBaseAddress, sizeof(Inst1)) == sizeof(Inst1))

{

/\*

nt!PsGetProcessSectionBaseAddress:

805287da 8bff mov edi,edi

805287dc 55 push ebp

805287dd 8bec mov ebp,esp

805287df 8b4508 mov eax,dword ptr [ebp+8]

805287e2 8b803c010000 mov eax,dword ptr [eax+13Ch]

805287e8 5d pop ebp

\*/

SectionBaseOffset = \*(ULONG\*)((char\*)PsGetProcessSectionBaseAddress + 10);

}

else if(MmIsAddressValid((PVOID)((char\*)PsGetProcessSectionBaseAddress + 10)) &&

RtlCompareMemory((PVOID)Inst2, PsGetProcessSectionBaseAddress, sizeof(Inst2)) == sizeof(Inst2))

//and BuildNumber==2600

{

SectionObjectOffset = \*(ULONG\*)((char\*)PsGetProcessSectionBaseAddress + 6);

}

if(SectionBaseOffset >= 276)

{

SectionObjectOffset = SectionBaseOffset - 4;

result = TRUE;

}

}

return result;

}

NTSTATUS GetFileObjectByProcessId(HANDLE ProcessId, PFILE\_OBJECT\* pFileObject)

{//法一 借助PEPROCESS结构

NTSTATUS status = STATUS\_UNSUCCESSFUL;

PEPROCESS Process = NULL;

if(!pFileObject)

return status;

if(NT\_SUCCESS(PsLookupProcessByProcessId(ProcessId,&Process)))

{

if(MajorVersion == 1 || MajorVersion == 2)

{

PSECTION Section = \*(PSECTION\*)((char\*)Process + SectionObjectOffset);

if(Section && MmIsAddressValid(Section) && Section->Segment && MmIsAddressValid(Section->Segment) &&

Section->Segment->ControlArea && MmIsAddressValid(Section->Segment->ControlArea))

{

\*pFileObject = Section->Segment->ControlArea->FilePointer;

if(\*pFileObject)

{

ObReferenceObject(\*pFileObject);

status = STATUS\_SUCCESS;

}

}

}

else if(MajorVersion == 5)

{

if(!PsReferenceProcessFilePointer)

PsReferenceProcessFilePointer = (ULONG)MmGetSystemRoutineAddress(&uName);

if(PsReferenceProcessFilePointer)

status = ((NTSTATUS (\_\_stdcall\*)(PEPROCESS,PFILE\_OBJECT\*))PsReferenceProcessFilePointer)(Process,pFileObject);

}

}

if(Process)

ObDereferenceObject(Process);

return status;

}

NTSTATUS GetPebBaseByProcessObject(PEPROCESS Process, PVOID \*PebBaseAddr)

{

NTSTATUS status = STATUS\_UNSUCCESSFUL;

HANDLE ProcessHandle = NULL;

PROCESS\_BASIC\_INFORMATION ProcessInformation;

if(NT\_SUCCESS(ObOpenObjectByPointer(Process, OBJ\_KERNEL\_HANDLE, NULL,0, NULL, KernelMode, &ProcessHandle)) &&

ZwQueryInformationProcess(ProcessHandle, ProcessBasicInformation, &ProcessInformation, sizeof(ProcessInformation), NULL) &&

ProcessInformation.PebBaseAddress)

{

\*PebBaseAddr = ProcessInformation.PebBaseAddress;

status = STATUS\_SUCCESS;

}

if(ProcessHandle)

ZwClose(ProcessHandle);

return status;

}

PFILE\_OBJECT GetFileObjectByProcessId(HANDLE ProcessId)

{//法二 借助PEB命令行

PEPROCESS Process = NULL;

PPEB PebBaseAddr = NULL;

BOOLEAN Attached = FALSE;

HANDLE FileHandle = NULL;

PVOID Buffer = NULL;

PFILE\_OBJECT FileObject = NULL;

UNICODE\_STRING ImagePathName;

UNICODE\_STRING FullPath;

OBJECT\_ATTRIBUTES Oa;

UNICODE\_STRING Prefix = RTL\_CONST\_STRING(L"\??\");

IO\_STATUS\_BLOCK IoStatus;

if(KeGetCurrentIrql() == PASSIVE\_LEVEL &&

NT\_SUCCESS(PsLookupProcessByProcessId(ProcessId, &Process)) &&

NT\_SUCCESS(GetPebBaseByProcessObject(Process, (PVOID\*)&PebBaseAddr)))

{

KeAttachProcess(Process);

Attached = TRUE;

\_\_try

{

ProbeForRead(PebBaseAddr,0x1D8,1);

ProbeForRead(PebBaseAddr->ProcessParameters,0x90,1);

ImagePathName = PebBaseAddr->ProcessParameters->ImagePathName;

if(ImagePathName.Length != 0 && ImagePathName.Length < 0xFFF8)

{

if((char\*)ImagePathName.Buffer < (char\*)PebBaseAddr->ProcessParameters)// 如果该成员是偏移而不是指针

ImagePathName.Buffer = (PWSTR)((ULONG)ImagePathName.Buffer + PebBaseAddr->ProcessParameters);

ProbeForRead(ImagePathName.Buffer,ImagePathName.Length,1);

ULONG FullLen = ImagePathName.Length;

if(!RtlPrefixUnicodeString(&Prefix, &ImagePathName, TRUE))

FullLen += 8;

Buffer = ExAllocatePool(NonPagedPool, FullLen);

if(Buffer)

{

RtlZeroMemory(Buffer,FullLen);

FullPath.Buffer = (PWCH)Buffer;

FullPath.MaximumLength = FullLen;

if(FullLen != ImagePathName.Length)

RtlAppendUnicodeStringToString(&FullPath,&Prefix);

RtlAppendUnicodeStringToString(&FullPath,&ImagePathName);

}

}

}

\_\_except(0)

{

Attached = FALSE;

}

InitializeObjectAttributes(&Oa,&FullPath,OBJ\_KERNEL\_HANDLE,NULL,NULL);

if(NT\_SUCCESS(ZwOpenFile(&FileHandle, SYNCHRONIZE | FILE\_READ\_ATTRIBUTES, &Oa, &Ios,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE, FILE\_SYNCHRONOUS\_IO\_NONALERT | FILE\_NON\_DIRECTORY\_FILE)))

ObReferenceObjectByHandle(FileHandle, 0, \*IoFileObjectType, KernelMode, (PVOID\*)&FileObject, NULL);

}

if(Attached)

KeDetachProcess();

if(FileHandle)

{

ZwClose(FileHandle);

FileHandle = NULL;

}

if(Process)

{

ObDereferenceObject(Process);

Process = NULL;

}

if(Buffer)

ExFreePool(Buffer);

return FileObject;

}

### 4.2 由线程句柄获取进程对象

PEPROCESS GetProcessObjectFromThreadHandle(HANDLE ThreadHandle)

{

PETHREAD Thread = NULL;

PEPROCESS Process = NULL;

if(ThreadHandle)

{

if(NT\_SUCCESS(ObReferenceObjectByHandle(ThreadHandle, 0, \*PsThreadType,

IsKernelHandle(ThreadHandle)?KernelMode:UserMode, (PVOID\*)&Thread, NULL)))

{

Process = IoThreadToProcess(Thread);

ObDereferenceObject(Thread);

}

}

return Process;

}

### 4.3 由线程对象获取进程Id

NTSTATUS GetProcIdFromProcessObject(PEPROCESS Process,PHANDLE pProcessId)

{

NTSTATUS status = STATUS\_UNSUCCESSFUL;

HANDLE ProcessHandle = NULL;

PROCESS\_BASIC\_INFORMATION ProcessInformation;

if(NT\_SUCCESS(ObOpenObjectByPointer(Process, OBJ\_KERNEL\_HANDLE, NULL,0, NULL, KernelMode, &ProcessHandle)) &&

ZwQueryInformationProcess(ProcessHandle, ProcessBasicInformation, &ProcessInformation, sizeof(ProcessInformation), NULL))

{

\*pProcessId = ProcessInformation.UniqueProcessId;

status = STATUS\_SUCCESS;

}

if(ProcessHandle)

ZwClose(ProcessHandle);

return status;

}

NTSTATUS GetThreadProcessId(PETHREAD Thread,PHANDLE pProcessId)

{

NTSTATUS status = STATUS\_UNSUCCESSFUL;

PVOID PsGetThreadProcessId = NULL;

ULONG OffsetEprocessToThreadObject = 0x22C;

if(MmIsAddressValid(Thread))

{

PsGetThreadProcessId = MmGetSystemRoutineAddress(&uName);

if(PsGetThreadProcessId)

{

\*pProcessId = ((HANDLE (\_\_stdcall\*)(PETHREAD))PsGetThreadProcessId)(Thread);

status = STATUS\_SUCCESS;

}

if(!NT\_SUCCESS(status))

{

ULONG Addr = (ULONG)Thread + OffsetEprocessToThreadObject;

if(Addr && MmIsAddressValid((PVOID)Addr))

{

PEPROCESS Process = \*(PEPROCESS\*)Addr;

if(Process && MmIsAddressValid(Process))

{

if(NT\_SUCCESS(GetProcIdFromProcessObject(Process, pProcessId)))

{

status = STATUS\_SUCCESS;

}

}

}

}

}

return status;

}

### 4.4 由Ntfs文件索引号获取文件对象

PFILE\_OBJECT GetRealFileObject(HANDLE ProcessId,PFILE\_OBJECT FileObject)

{

UNICODE\_STRING uNtfs = RTL\_CONSTANT\_STRING(L"\\Ntfs");

PDRIVER\_OBJECT NtfsDrvObj = NULL;

PDEVICE\_OBJECT NtfsDevObj = NULL,fsDevObj = NULL;

PFILE\_OBJECT NtfsFileObj = NULL,ObjFileObj = NULL,RealFileObj = NULL;

NTSTATUS status;

PFILE\_OBJECT Ntfs;

BOOLEAN Real=FALSE;

//检查是否文件属于NTFS文件系统

status = IoGetDeviceObjectPointer(&uNtfs,0,&NtfsFileObj,&NtfsDevObj);

if(NT\_SUCCESS(status) && NtfsFileObj && MmIsAddressValid(NtfsFileObj) && MmIsAddressValid(NtfsFileObj->DeviceObject))

NtfsDrvObj = NtfsFileObj->DeviceObject->DriverObject;

fsDevObj = IoGetBaseFileSystemDeviceObject(FileObject);//FileSystem\Ntfs

if(fsDevObj && MmIsAddressValid(fsDevObj) && fsDevObj->DriverObject == NtfsDevObj)

{

FILE\_STANDARD\_INFORMATION StandardInfo;

ULONG RetLen;

status = IoQueryFileInformation(FileObject,FileStandardInformation,sizeof(StandardInfo),&StandardInfo,&RetLen);

if(NT\_SUCCESS(status))

{

if(StandardInfo.NumberOfLinks > 1)

{

ObjFileObj = GetFileObjectByProcessId(ProcessId);

if(ObjFileObj && ObjFileObj->FsContext != FileObj->FsContext)

{

//2种方式的FsContext不同，说明可能被拦截，下面采用Ntfs文件号获取

Real = TRUE;

ObDereferenceObject(ObjFileObj);

}

}

}

}

if(!Real)

{

//获取该文件Ntfs文件号

FILE\_INTERNAL\_INFORMATION InternalInfo;

ULONG RetLen;

PVOID Buf = ExAllocatePool(PagedPool,1024);

status = IoQueryFileInformation(FileObj,FileInternalInformation,sizeof(InternalInfo),&InternalInfo,&RetLen);

if(NT\_SUCCESS(status) && Buf)

{

//获取父目录信息

RtlZeroMemory(Buf,1024);

POBJECT\_NAME\_INFORMATION DeviceName = (POBJECT\_NAME\_INFORMATION)Buf;

OBJECT\_ATTRIBUTES Oa;

IO\_STATUS\_BLOCK IoStatus;

HANDLE DeviceHandle = NULL;

HANDLE FileHandle = NULL;

status = ObQueryNameString(FileObj->DeviceObject,Buf,1024,RetLen);

if(NT\_SUCCESS(status) && DeviceName->Name.Buffer)

{

InitializeObjectAttributes(&Oa,&DeviceName->Name,OBJ\_CASE\_INSENSITIVE | OBJ\_KERNEL\_HANDLE, NULL, NULL);

status = ZwOpenFile(&DeviceHandle , 0, &Oa, &IoStatus, 0, FILE\_NON\_DIRECTORY\_FILE);

if(NT\_SUCCESS(status))

{

UNICODE\_STRING InnerFileName = {sizeof(InternalInfo), sizeof(InternalInfo), &InternalInfo};

InitializeObjectAttributes(&Oa, &InnerFileName, OBJ\_CASE\_INSENSITIVE | OBJ\_KERNEL\_HANDLE, DeviceHandle, NULL);

status = ZwOpenFile(FileHandle, SYNCHRONIZE | FILE\_READ\_ATTRIBUTES, &Oa, &IoStatus,

FILE\_SHARE\_READ | FILE\_SHARE\_WRITE | FILE\_SHARE\_DELETE, FILE\_OPEN\_BY\_FILE\_ID | FILE\_NON\_DIRECTORY\_FILE | FILE\_SYNCHRONOUS\_IO\_NONALERT);

if(NT\_SUCCESS(status))

{

//用文件索引号打开文件成功

ObReferenceObjectByHandle(FileHandle, 0, \*IoFileObjectType, KernelMode, &RealFileObj);

}

}

}

if(FileHandle)

ZwClose(FileHandle);

if(DeviceHandle)

ZwClose(DeviceHandle);

}

if(Buf)

ExFreePool(Buf);

}

return RealFileObj;

}

### 4.5 长度反汇编引擎

用于获取指令长度

int DisasmLen(unsigned char\* bytecode)

{

unsigned long decode1[256][7]=

{

{0x00,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x01,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x02,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x03,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x04,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x05,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x06,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x07,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x08,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x09,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x0a,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x0b,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x0c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x0d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x0e,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x0f,0x03,0x03,0x02,0x00,0x00,0x00,},

{0x10,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x11,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x12,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x13,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x14,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x15,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x16,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x17,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x18,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x19,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1a,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1b,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x20,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x21,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x22,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x23,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x24,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x25,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x26,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x27,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x28,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x29,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2c,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x30,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x31,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x32,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x33,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x34,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x35,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x36,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x37,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x38,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x39,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3a,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3b,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x40,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x41,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x42,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x43,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x44,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x45,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x46,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x47,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x48,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x49,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x4a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x4b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x4c,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x4d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x4e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x4f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x50,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x51,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x52,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x53,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x54,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x55,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x56,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x57,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x58,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x59,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x5a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x5b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x5c,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x5d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x5e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x5f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x60,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x61,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x62,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x63,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x64,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x65,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x66,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x67,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x68,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x69,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x6a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x6b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x6c,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x6d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x6e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x6f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x70,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x71,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x72,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x73,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x74,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x75,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x76,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x77,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x78,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x79,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x7a,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x7b,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x7c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x7d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x7e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x7f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x80,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x81,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x82,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x83,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x84,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x85,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x86,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x87,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x88,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x89,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x8a,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x8b,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x8c,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x8d,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x8e,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x8f,0x05,0x03,0x00,0x01,0x00,0x00,},

{0x90,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x91,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x92,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x93,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x94,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x95,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x96,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x97,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x98,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x99,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x9a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x9b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x9c,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x9d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x9e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x9f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xa0,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xa1,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xa2,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xa3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xa4,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xa5,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xa6,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa7,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa8,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xa9,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xaa,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xab,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xac,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xad,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xae,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xaf,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb0,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb1,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb2,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb4,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb5,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb6,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb7,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xb8,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xb9,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xba,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xbb,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xbc,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xbd,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xbe,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xbf,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc0,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc1,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc2,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc4,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xc5,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xc6,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xc7,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc8,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xc9,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xca,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xcb,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xcc,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xcd,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xce,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xcf,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xd0,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xd1,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd2,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd4,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd5,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd6,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd7,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd8,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd9,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xda,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdb,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdc,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdd,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xde,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdf,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe0,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe1,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe2,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe4,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe5,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe6,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe7,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe8,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe9,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xea,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xeb,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xec,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xed,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xee,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xef,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf0,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf1,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf2,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf4,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf5,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf6,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf7,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf8,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xf9,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xfa,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xfb,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xfc,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xfd,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xfe,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xff,0x01,0x01,0x00,0x00,0x00,0x00,},

};

unsigned long decode2[256][7]=

{

{0x00,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x01,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x02,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x03,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x04,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x05,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x06,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x07,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x08,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x09,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x0a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x0b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x0c,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x0d,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x0e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x0f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x10,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x11,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x12,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x13,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x14,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x15,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x16,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x17,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x18,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x19,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x1a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x1b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x1c,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x1d,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x1e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x1f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x20,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x21,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x22,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x23,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x24,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x25,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x26,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x27,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x28,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x29,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x2c,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x2d,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x2e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x2f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x30,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x31,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x32,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x33,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x34,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x35,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x36,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x37,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x38,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x39,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x3a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x3b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x3c,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x3d,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x3e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x3f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x40,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x41,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x42,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x43,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x44,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x45,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x46,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x47,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x48,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x49,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x4a,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x4b,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x4c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x4d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x4e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x4f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x50,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x51,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x52,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x53,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x54,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x55,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x56,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x57,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x58,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x59,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x5a,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x5b,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x5c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x5d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x5e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x5f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x60,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x61,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x62,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x63,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x64,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x65,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x66,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x67,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x68,0x05,0x03,0x00,0x00,0x00,0x00,},

{0x69,0x06,0x04,0x01,0x00,0x04,0x00,},

{0x6a,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x6b,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x6c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x6d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x6e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x6f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x70,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x71,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x72,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x73,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x74,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x75,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x76,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x77,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x78,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x79,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x7a,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x7b,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x7c,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x7d,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x7e,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x7f,0x02,0x02,0x00,0x01,0x00,0x00,},

{0x80,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x81,0x06,0x04,0x01,0x00,0x04,0x00,},

{0x82,0x02,0x02,0x00,0x00,0x00,0x00,},

{0x83,0x03,0x03,0x01,0x00,0x01,0x00,},

{0x84,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x85,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x86,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x87,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x88,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x89,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x8a,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x8b,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x8c,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x8d,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x8e,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x8f,0x02,0x02,0x01,0x00,0x00,0x00,},

{0x90,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x91,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x92,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x93,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x94,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x95,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x96,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x97,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x98,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x99,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x9a,0x07,0x05,0x00,0x00,0x00,0x01,},

{0x9b,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x9c,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x9d,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x9e,0x01,0x01,0x00,0x00,0x00,0x00,},

{0x9f,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa0,0x05,0x03,0x00,0x00,0x00,0x02,},

{0xa1,0x05,0x03,0x00,0x00,0x00,0x02,},

{0xa2,0x05,0x03,0x00,0x00,0x00,0x02,},

{0xa3,0x05,0x03,0x00,0x00,0x00,0x02,},

{0xa4,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa5,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa6,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa7,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xa8,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xa9,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xaa,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xab,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xac,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xad,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xae,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xaf,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xb0,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb1,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb2,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb3,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb4,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb5,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb6,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb7,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xb8,0x05,0x03,0x00,0x00,0x00,0x08,},

{0xb9,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xba,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xbb,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xbc,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xbd,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xbe,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xbf,0x05,0x03,0x00,0x00,0x00,0x00,},

{0xc0,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xc1,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xc2,0x03,0x03,0x00,0x00,0x00,0x00,},

{0xc3,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xc4,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc5,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xc6,0x03,0x03,0x01,0x00,0x01,0x00,},

{0xc7,0x06,0x04,0x01,0x00,0x04,0x00,},

{0xc8,0x04,0x04,0x00,0x00,0x00,0x00,},

{0xc9,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xca,0x03,0x03,0x00,0x00,0x00,0x01,},

{0xcb,0x01,0x01,0x00,0x00,0x00,0x01,},

{0xcc,0x01,0x01,0x00,0x00,0x00,0x01,},

{0xcd,0x02,0x02,0x00,0x00,0x00,0x01,},

{0xce,0x01,0x01,0x00,0x00,0x00,0x01,},

{0xcf,0x01,0x01,0x00,0x00,0x00,0x01,},

{0xd0,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd1,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd2,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd3,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd4,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xd5,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xd6,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xd7,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xd8,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xd9,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xda,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdb,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdc,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdd,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xde,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xdf,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xe0,0x02,0x02,0x00,0x01,0x00,0x04,},

{0xe1,0x02,0x02,0x00,0x01,0x00,0x04,},

{0xe2,0x02,0x02,0x00,0x01,0x00,0x04,},

{0xe3,0x02,0x02,0x00,0x01,0x00,0x00,},

{0xe4,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xe5,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xe6,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xe7,0x02,0x02,0x00,0x00,0x00,0x00,},

{0xe8,0x05,0x03,0x00,0x01,0x00,0x00,},

{0xe9,0x05,0x03,0x00,0x01,0x00,0x00,},

{0xea,0x07,0x05,0x00,0x00,0x00,0x01,},

{0xeb,0x02,0x02,0x00,0x01,0x00,0x00,},

{0xec,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xed,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xee,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xef,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf0,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf1,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf2,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf3,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf4,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf5,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf6,0x00,0x00,0x00,0x00,0x00,0x00,},

{0xf7,0x00,0x00,0x00,0x00,0x00,0x00,},

{0xf8,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xf9,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xfa,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xfb,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xfc,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xfd,0x01,0x01,0x00,0x00,0x00,0x00,},

{0xfe,0x02,0x02,0x01,0x00,0x00,0x00,},

{0xff,0x02,0x02,0x01,0x00,0x00,0x00,},

};

unsigned char decode3[256]=

{

0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,

0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,

0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,

0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,0x00,0x00,0x00,0x00,0x11,0x24,0x00,0x00,

0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,

0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,

0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,

0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x02,0x01,0x01,0x01,

0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,

0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,

0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,

0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x05,0x04,0x04,0x04,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,

};

unsigned char\* ptr = bytecode;

unsigned long len = 0,var2 = 0,var3 = 0, decodel[7]={0};

unsigned long\* pdecode = 0;

switch(\*ptr)

{

case 0xF:

ptr++;

len = 1;

pdecode = &decode1[\*ptr][0];

break;

case 0x26:

case 0x2E:

case 0x36:

case 0x3E:

case 0x64:

case 0x65:

len = 1;

ptr++;

break;

case 0x66:

len = 1;

var3 = 1;

ptr++;

break;

case 0x67:

len = 1;

var2 = 1;

ptr++;

break;

case 0xF0:

case 0xF2:

case 0xF3:

len = 1;

ptr++;

break;

case 0xF6:

decodel[0] = 0xF6;

if(\*(ptr+1) & 0x38)

{

decodel[1] = 2;

decodel[2] = 2;

decodel[3] = 1;

decodel[5] = 0;

}

else

{

decodel[1] = 3;

decodel[2] = 3;

decodel[3] = 1;

decodel[5] = 1;

}

pdecode = decodel;

break;

case 0xF7:

decodel[0] = 0xF6;

decodel[3] = 1;

if(\*(ptr+1) & 0x38)

{

decodel[1] = 6;

decodel[2] = 4;

decodel[5] = 4;

}

else

{

decodel[1] = 2;

decodel[2] = 2;

decodel[5] = 0;

}

pdecode = decodel;

break;

default:

break;

}

if(!pdecode)

pdecode = decode2[\*ptr];

if(pdecode[6] & 2)

{

if(var2 == 0)

len += pdecode[1];

else

len += pdecode[2];

}

else

{

if(var3 == 0)

len += pdecode[1];

else

len += pdecode[2];

}

if(pdecode[3])

{

unsigned char var4 = ptr[pdecode[3]];

len += decode3[var4] & 0xF;

if((decode3[var4] & 0x10) && (ptr[pdecode[3] + 1] & 7) == 5)

{

switch(var4 & 0xC0)

{

case 0x40:

len++;

break;

case 0x00:

case 0x80:

len += 4;

break;

default:

break;

}

}

}

return len;

}