

说到实现监控进程创建，相信不少人的第一反应是 Hook `NtCreateProcessEx` 或 `NtCreateUserProcess`。不过从今以后，我们完全可以摒弃这一思路，使用在 WINDOWS VISTA 之后才有的一个新 API：

`PsSetCreateProcessNotifyRoutineEx`。有些人乍一看，觉得这个函数很眼熟。没错，在 NT5 系列系统上，有一个函数名字叫

`PsSetCreateProcessNotifyRoutine`，这个函数能创建一个回调，当有进程被创建或者死亡时，会调用你的回调函数通知你的驱动做出相应的反应。不过由于 `PsSetCreateProcessNotifyRoutine` 的回调函数的可用信息太少了（仅有『新进程 PID』、『父进程 PID』和『进程创建或退出』这三个标志），所以有了一个新函数：`PsSetCreateProcessNotifyRoutineEx`。此函数的原型为：

```
NTSTATUS PsSetCreateProcessNotifyRoutineEx(
    __in PCREATE_PROCESS_NOTIFY_ROUTINE_EX NotifyRoutine,
    __in BOOLEAN Remove
);
```

接下来看看回调函数 `NotifyRoutine` 的原型：

```
VOID CreateProcessNotifyEx(
    __inout PEPROCESS Process,    //新进程 EPROCESS
    __in HANDLE ProcessId,        //新进程 PID
    __in_opt PPS_CREATE_NOTIFY_INFO CreateInfo //新进程详细信息（仅在创建进程时有效）
);
```

接下来看看 `PS_CREATE_NOTIFY_INFO` 结构体的定义：

```
typedef struct _PS_CREATE_NOTIFY_INFO {
    SIZE_T          Size;
    union {
        ULONG       Flags;
        struct {
            ULONG FileOpenNameAvailable :1;
            ULONG Reserved :31;
        };
    };
    HANDLE          ParentProcessId;
    CLIENT_ID       CreatingThreadId;
    struct _FILE_OBJECT *FileObject;
    PCUNICODE_STRING ImageFileName;
    PCUNICODE_STRING CommandLine;
    NTSTATUS        CreationStatus;
} PS_CREATE_NOTIFY_INFO, *PPS_CREATE_NOTIFY_INFO;
```

可见在这个结构体里，包含了很多的信息，不仅包含父进程 ID，父线程 ID、甚至直接包括程序的路径（不用通过 `FileObject` 来取，路径直接包含在了 `ImageFileName` 里）和命令行参数！最绝的是，如果要阻止进程创建，直接把这个结构体的 `CreationStatus` 成员改为 `STATUS_UNSUCCESSFUL` 即可。用这个回调

里弹框实现进程创建防御，绝对比 HOOK NtCreateUserProcess 方便百倍！以下的代码，实现了监视进程的创建和退出事件，并禁止“计算器”进程创建。

```
//创建回调:
st=PsSetCreateProcessNotifyRoutineEx((PCREATE_PROCESS_NOTIFY_ROUTINE_EX)MyCreateProcessNotifyEx, FALSE);

//删除回调:
PsSetCreateProcessNotifyRoutineEx((PCREATE_PROCESS_NOTIFY_ROUTINE_EX)MyCreateProcessNotifyEx, TRUE);

//回调处理
NTKERNELAPI PCHAR PsGetProcessImageFileName(PEPROCESS Process);
NTKERNELAPI NTSTATUS PsLookupProcessByProcessId(HANDLE ProcessId, PEPROCESS *Process);
//通过PID获得进程名
PCHAR GetProcessNameByProcessId(HANDLE ProcessId)
{
    NTSTATUS st=STATUS_UNSUCCESSFUL;
    PEPROCESS ProcessObj=NULL;
    PCHAR string=NULL;
    st = PsLookupProcessByProcessId(ProcessId, &ProcessObj);
    if (NT_SUCCESS(st))
    {
        string = PsGetProcessImageFileName(ProcessObj);
        ObfDereferenceObject(ProcessObj);
    }
    return string;
}

//回调本体
VOID MyCreateProcessNotifyEx
(
    __inout PEPROCESS Process,
    __in HANDLE ProcessId,
    __in_opt PPS_CREATE_NOTIFY_INFO CreateInfo
)
{
    NTSTATUS st=0;
    HANDLE hProcess=NULL;
    OBJECT_ATTRIBUTES oa={0};
    CLIENT_ID ClientId={0};
    char xxx[16]={0};
    if (CreateInfo!=NULL) //进程创建事件
    {
        DbgPrint("[monitor_create_process_x64][%ld]创建进程: %wZ",
            CreateInfo->ParentProcessId,
            GetProcessNameByProcessId(CreateInfo->ParentProcessId),
            CreateInfo->ImageFileName);
    }
}
```

```

strcpy(xxx, PsGetProcessImageFileName(Process));
if(!_stricmp(xxx, "calc.exe"))
{
    DbgPrint("禁止创建计算器进程!");
    CreateInfo->CreationStatus=STATUS_UNSUCCESSFUL;    //禁止创建进程
}
}
else
{
    DbgPrint("[monitor_create_process_x64]进程退出: %s", PsGetProcessImageFileName(Process));
}
}
}

```

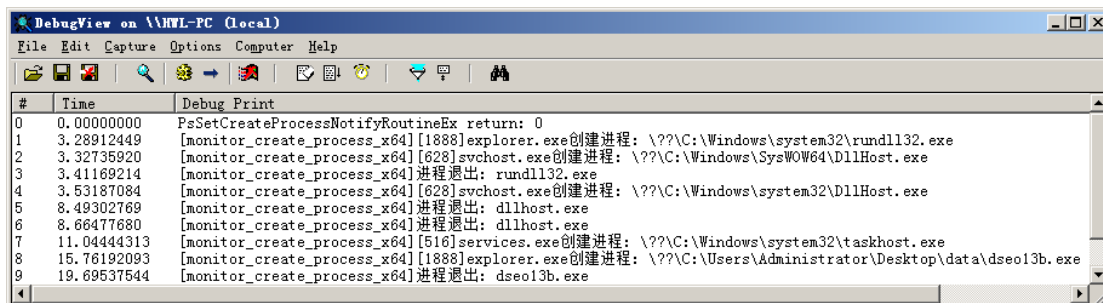
另外需要特别说明的是，直接编译这份代码，加载驱动后创建回调会失败。因为 MSDN 上有这么一个限制：

The image that contains the callback routine pointer must have IMAGE_DLLCHARACTERISTICS_FORCE_INTEGRITY set in its image header.

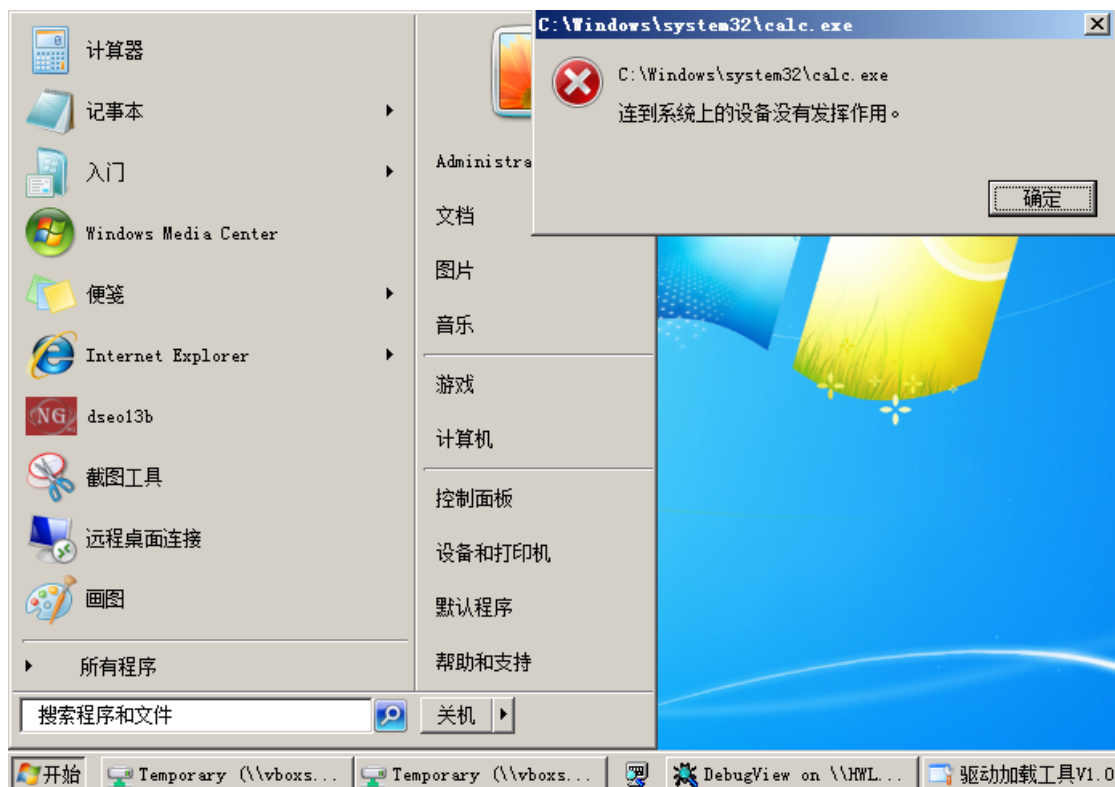
解决这个限制是在 source 文件里加上这么一行：

LINKER_FLAGS=/INTEGRITYCHECK

加载驱动后效果如下：



创建计算器进程时的效果：



接下来说说线程创建和退出的监视。监视线程的创建和退出是使用 `PsSetCreateThreadNotifyRoutine` 来创建回调函数，而取消回调函数则使用 `PsRemoveCreateThreadNotifyRoutine`。创建函数、取消函数与回调函数的原型如下：

```
//创建回调的函数
NTSTATUS PsSetCreateThreadNotifyRoutine
(
    _In_    PCREATE_THREAD_NOTIFY_ROUTINE NotifyRoutine
);

//取消回调的函数
NTSTATUS PsRemoveCreateThreadNotifyRoutine
(
    _In_    PCREATE_THREAD_NOTIFY_ROUTINE NotifyRoutine
);

//回调函数原型
VOID (*PCREATE_THREAD_NOTIFY_ROUTINE)
(
    IN HANDLE ProcessId,
    IN HANDLE ThreadId,
    IN BOOLEAN Create
);
```

类似于上面的进程监控，当有线程创建/退出时，系统会调用一次回调函数告知你，此时可以采取相应的动作。不过郁闷的是，在 `CreateThreadNotify` 里，

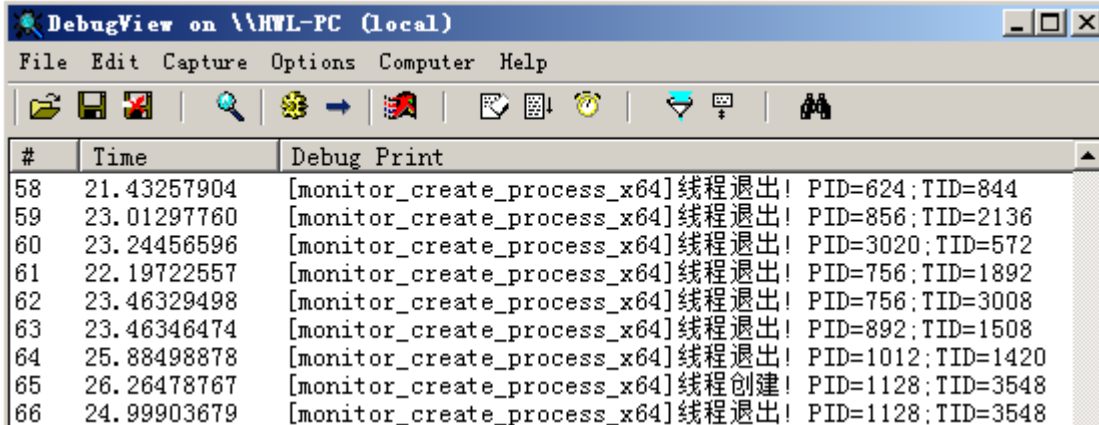
只能监视线程的创建和退出，却不能阻止线程创建。经我测试，在通知回调函数的时候，线程尚未初始化完毕，使用 PsLookupThreadByThreadId 获得 ETHREAD 有时会失败，即使侥幸成功，也不能调用 PspTerminateThreadByPointer 来结束新线程，否则会蓝屏。代码如下：

```
//设置回调：
st=PsSetCreateThreadNotifyRoutine(MyCreateThreadNotify);

//删除回调：
PsRemoveCreateThreadNotifyRoutine(MyCreateThreadNotify);

//回调函数：
VOID MyCreateThreadNotify
(
    IN HANDLE ProcessId,
    IN HANDLE ThreadId,
    IN BOOLEAN Create
)
{
    if(Create)
        DbgPrint("线程创建! PID=%ld;TID=%ld",ProcessId,ThreadId);
    else
        DbgPrint("线程退出! PID=%ld;TID=%ld",ProcessId,ThreadId);
}
```

效果图：



#	Time	Debug Print
58	21.43257904	[monitor_create_process_x64] 线程退出! PID=624;TID=844
59	23.01297760	[monitor_create_process_x64] 线程退出! PID=856;TID=2136
60	23.24456596	[monitor_create_process_x64] 线程退出! PID=3020;TID=572
61	22.19722557	[monitor_create_process_x64] 线程退出! PID=756;TID=1892
62	23.46329498	[monitor_create_process_x64] 线程退出! PID=756;TID=3008
63	23.46346474	[monitor_create_process_x64] 线程退出! PID=892;TID=1508
64	25.88498878	[monitor_create_process_x64] 线程退出! PID=1012;TID=1420
65	26.26478767	[monitor_create_process_x64] 线程创建! PID=1128;TID=3548
66	24.99903679	[monitor_create_process_x64] 线程退出! PID=1128;TID=3548

课后作业：想办法阻止指定进程新建线程。