

与断链隐藏进程功能类似，关于断链进程隐藏可参考《DKOM 实现进程隐藏》这一篇文章，断链隐藏驱动自身则用于隐藏自身SYS驱动文件，当驱动加载后那么使用ARK工具扫描将看不到自身驱动模块，此方法可能会触发PG会蓝屏，在某些驱动辅助中也会使用这种方法隐藏自己。

要是实现隐层，具体步骤可被概述为如下几步；

DriverEntry函数是驱动程序的入口函数，当驱动程序被加载时，该函数会被操作系统自动调用。该函数接收两个参数：

- Driver: 指向 DRIVER_OBJECT 结构的指针，用于描述驱动程序的属性和行为；
- RegistryPath: 指向一个 UNICODE_STRING 结构的指针，用于指示驱动程序的注册表路径。

功能是从驱动程序的 `DriverSection` 中移除自身模块的信息，并启动一个新的系统线程来修改 `DRIVER_OBJECT` 结构的属性。具体来说，该函数执行以下步骤：

- 1.从 `Driver->DriverSection` 中获取自身模块的 `LIST_ENTRY` 结构指针，即 `pModuleList`；
- 2.将 `pModuleList` 从双向链表中移除，即将其前一个节点的 `Flink` 指向其后一个节点的地址，将其后一个节点的 `Blink` 指向其前一个节点的地址；
- 3.调用 `PsCreateSystemThread` 创建一个新的系统线程，线程入口函数为 `ThreadRun`，传递参数 `Driver`；
- 4.设置 `Driver->DriverUnload` 为 `UnDriver`，即卸载函数的地址并返回 `STATUS_SUCCESS` 表示初始化成功。

`ThreadRun` 函数是一个系统线程的入口函数，该函数会在 `PsCreateSystemThread` 中被调用。该函数接收一个参数：

- `StartContext`: 线程入口函数传递的参数，在本例中即为 `Driver`。

函数的主要功能是延时一段时间，然后修改传递进来的 `DRIVER_OBJECT` 结构的属性。具体来说，该函数执行以下步骤：

- 延时3秒钟，以等待驱动程序的初始化完成；
- 将 `Driver` 转换为 `PDRIVER_OBJECT` 类型；
- 将 `Driver` 的各个属性设置为 `NULL` 或 `0`，以清空驱动程序的信息；
- 关闭创建该线程的句柄。

上方流程，驱动实现代码总结起来如下所示：

```
#include <ntifs.h>

HANDLE hThread;

VOID ThreadRun(PVOID StartContext)
{
    LARGE_INTEGER times;
    PDRIVER_OBJECT pDriverObject;

    // 等待3秒 单位是纳秒
    times.QuadPart = -30 * 1000 * 1000;

    KeDelayExecutionThread(KernelMode, FALSE, &times);
    pDriverObject = (PDRIVER_OBJECT)StartContext;

    // 修改模块信息
    pDriverObject->DriverSize = 0;
```

```

pDriverObject->DriverSection = NULL;
pDriverObject->DriverExtension = NULL;
pDriverObject->DriverStart = NULL;
pDriverObject->DriverInit = NULL;
pDriverObject->FastIoDispatch = NULL;
pDriverObject->DriverStartIo = NULL;

ZwClose(hThread);
}

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint(("Uninstall Driver Is OK \n"));
}

NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
{
    DbgPrint(("hello lyshark \n"));

    PLIST_ENTRY pModuleList;
    pModuleList = Driver->DriverSection;

    // 前一个模块的Flink=本模块的Flink
    pModuleList->Blink->Flink = pModuleList->Flink;

    // 前一个模块的Blink=本模块的Blink
    pModuleList->Flink->Blink = pModuleList->Blink;
    PsCreateSystemThread(&hThread, GENERIC_ALL, NULL, NULL, NULL, ThreadRun, Driver);

    Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}

```

输出效果如下，驱动每隔3秒执行一次模块修改：

#	Time	Debug Print
8	3.37202597	2932500000 - STORMINI: StorNVMe - POWER: ACTIVE
9	3.39019132	hello lyshark
10	4.37266254	2942500000 - STORMINI: StorNVMe - POWER: IDLE
11	4.37641191	2942500000 - STORMINI: StorNVMe - POWER: ACTIVE
12	5.37726450	2952500000 - STORMINI: StorNVMe - POWER: IDLE
13	5.38341284	2952656250 - STORMINI: StorNVMe - POWER: ACTIVE
14	6.38431406	2962656250 - STORMINI: StorNVMe - POWER: IDLE
15	6.39120293	hello lyshark
16	6.73498964	2966093750 - STORMINI: StorNVMe - POWER: ACTIVE
17	7.74737120	2976250000 - STORMINI: StorNVMe - POWER: IDLE
18	8.54790783	2984218750 - STORMINI: StorNVMe - POWER: ACTIVE
19	8.54837704	2984218750 - STORMINI: StorNVMe - POWER: IDLE
20	8.54850674	2984218750 - STORMINI: StorNVMe - POWER: ACTIVE
21	9.55587006	2994375000 - STORMINI: StorNVMe - POWER: IDLE
22	9.90620899	2997812500 - STORMINI: StorNVMe - POWER: ACTIVE