# 在驱动开发中我们有时需要得到驱动自身是否被加载成功的状态，这个功能看似没啥用实际上在某些特 殊场景中还是需要的，如下代码实现了判断当前驱动是否加载成功，如果加载成功, 则输出该驱动的详细路径信息。

该功能实现的核心函数是 NtQuerySystemInformation 这是一个微软未公开的函数，也没有文档化，不过我们仍然可以通过动态指针的方式调用到它，该函数可以查询到很多系统信息状态，首先需要定义一 个指针。



typedef NTSTATUS( NTQUERYSYSTEMINFORMATION)(

IN ULONG SystemInformationClass, OUT PVOID SystemInformation,

IN ULONG\_PTR SystemInformationLength, OUT PULONG\_PTR ReturnLength OPTIONAL);

其次还需要一个 SYSTEM\_MODULE\_INFORMATION 该结构内可以得到模块入口信息模块名称等，调用

NtQuerySystemInformation 数据会被格式化为 SYSTEM\_MODULE\_INFORMATION 方便调用。



typedef struct \_SYSTEM\_MODULE\_INFORMATION { HANDLE Section;

PVOID MappedBase; PVOID Base;

ULONG Size; ULONG Flags;

USHORT LoadOrderIndex;

USHORT InitOrderIndex; USHORT LoadCount; USHORT PathLength; CHAR ImageName[256];

} SYSTEM\_MODULE\_INFORMATION, PSYSTEM\_MODULE\_INFORMATION;

最后是 SYSTEM\_INFORMATION\_CLASS 该结构同样是一个未文档化的结构体，本此代码中需要用到的枚举类型是 SystemModuleInformation 其他类型也放这里后期做参考用。

typedef enum \_SYSTEM\_INFORMATION\_CLASS

{

SystemBasicInformation = 0x0, SystemProcessorInformation = 0x1, SystemPerformanceInformation = 0x2, SystemTimeOfDayInformation = 0x3, SystemPathInformation = 0x4, SystemProcessInformation = 0x5, SystemCallCountInformation = 0x6, SystemDeviceInformation = 0x7, SystemProcessorPerformanceInformation = 0x8, SystemFlagsInformation = 0x9, SystemCallTimeInformation = 0xa, SystemModuleInformation = 0xb, SystemLocksInformation = 0xc, SystemStackTraceInformation = 0xd, SystemPagedPoolInformation = 0xe, SystemNonPagedPoolInformation = 0xf,

SystemHandleInformation = 0x10, SystemObjectInformation = 0x11, SystemPageFileInformation = 0x12, SystemVdmInstemulInformation = 0x13, SystemVdmBopInformation = 0x14, SystemFileCacheInformation = 0x15, SystemPoolTagInformation = 0x16, SystemInterruptInformation = 0x17, SystemDpcBehaviorInformation = 0x18, SystemFullMemoryInformation = 0x19, SystemLoadGdiDriverInformation = 0x1a, SystemUnloadGdiDriverInformation = 0x1b, SystemTimeAdjustmentInformation = 0x1c, SystemSummaryMemoryInformation = 0x1d, SystemMirrorMemoryInformation = 0x1e, SystemPerformanceTraceInformation = 0x1f, SystemObsolete0 = 0x20, SystemExceptionInformation = 0x21, SystemCrashDumpStateInformation = 0x22, SystemKernelDebuggerInformation = 0x23, SystemContextSwitchInformation = 0x24, SystemRegistryQuotaInformation = 0x25, SystemExtendServiceTableInformation = 0x26, SystemPrioritySeperation = 0x27, SystemVerifierAddDriverInformation = 0x28, SystemVerifierRemoveDriverInformation = 0x29, SystemProcessorIdleInformation = 0x2a, SystemLegacyDriverInformation = 0x2b, SystemCurrentTimeZoneInformation = 0x2c, SystemLookasideInformation = 0x2d, SystemTimeSlipNotification = 0x2e, SystemSessionCreate = 0x2f, SystemSessionDetach = 0x30, SystemSessionInformation = 0x31, SystemRangeStartInformation = 0x32, SystemVerifierInformation = 0x33, SystemVerifierThunkExtend = 0x34, SystemSessionProcessInformation = 0x35, SystemLoadGdiDriverInSystemSpace = 0x36, SystemNumaProcessorMap = 0x37, SystemPrefetcherInformation = 0x38, SystemExtendedProcessInformation = 0x39, SystemRecommendedSharedDataAlignment = 0x3a, SystemComPlusPackage = 0x3b, SystemNumaAvailableMemory = 0x3c, SystemProcessorPowerInformation = 0x3d, SystemEmulationBasicInformation = 0x3e, SystemEmulationProcessorInformation = 0x3f, SystemExtendedHandleInformation = 0x40, SystemLostDelayedWriteInformation = 0x41, SystemBigPoolInformation = 0x42, SystemSessionPoolTagInformation = 0x43, SystemSessionMappedViewInformation = 0x44, SystemHotpatchInformation = 0x45, SystemObjectSecurityMode = 0x46,

SystemWatchdogTimerHandler = 0x47, SystemWatchdogTimerInformation = 0x48, SystemLogicalProcessorInformation = 0x49, SystemWow64SharedInformationObsolete = 0x4a, SystemRegisterFirmwareTableInformationHandler = 0x4b, SystemFirmwareTableInformation = 0x4c, SystemModuleInformationEx = 0x4d, SystemVerifierTriageInformation = 0x4e, SystemSuperfetchInformation = 0x4f, SystemMemoryListInformation = 0x50, SystemFileCacheInformationEx = 0x51, SystemThreadPriorityClientIdInformation = 0x52, SystemProcessorIdleCycleTimeInformation = 0x53, SystemVerifierCancellationInformation = 0x54, SystemProcessorPowerInformationEx = 0x55, SystemRefTraceInformation = 0x56, SystemSpecialPoolInformation = 0x57, SystemProcessIdInformation = 0x58, SystemErrorPortInformation = 0x59, SystemBootEnvironmentInformation = 0x5a, SystemHypervisorInformation = 0x5b, SystemVerifierInformationEx = 0x5c, SystemTimeZoneInformation = 0x5d, SystemImageFileExecutionOptionsInformation = 0x5e, SystemCoverageInformation = 0x5f, SystemPrefetchPatchInformation = 0x60, SystemVerifierFaultsInformation = 0x61, SystemSystemPartitionInformation = 0x62, SystemSystemDiskInformation = 0x63, SystemProcessorPerformanceDistribution = 0x64, SystemNumaProximityNodeInformation = 0x65, SystemDynamicTimeZoneInformation = 0x66, SystemCodeIntegrityInformation = 0x67, SystemProcessorMicrocodeUpdateInformation = 0x68, SystemProcessorBrandString = 0x69, SystemVirtualAddressInformation = 0x6a, SystemLogicalProcessorAndGroupInformation = 0x6b, SystemProcessorCycleTimeInformation = 0x6c, SystemStoreInformation = 0x6d, SystemRegistryAppendString = 0x6e, SystemAitSamplingValue = 0x6f, SystemVhdBootInformation = 0x70, SystemCpuQuotaInformation = 0x71, SystemNativeBasicInformation = 0x72, SystemErrorPortTimeouts = 0x73, SystemLowPriorityIoInformation = 0x74, SystemBootEntropyInformation = 0x75, SystemVerifierCountersInformation = 0x76, SystemPagedPoolInformationEx = 0x77, SystemSystemPtesInformationEx = 0x78, SystemNodeDistanceInformation = 0x79, SystemAcpiAuditInformation = 0x7a, SystemBasicPerformanceInformation = 0x7b, SystemQueryPerformanceCounterInformation = 0x7c, SystemSessionBigPoolInformation = 0x7d,

SystemBootGraphicsInformation = 0x7e, SystemScrubPhysicalMemoryInformation = 0x7f, SystemBadPageInformation = 0x80, SystemProcessorProfileControlArea = 0x81, SystemCombinePhysicalMemoryInformation = 0x82, SystemEntropyInterruptTimingInformation = 0x83, SystemConsoleInformation = 0x84, SystemPlatformBinaryInformation = 0x85, SystemThrottleNotificationInformation = 0x86, SystemHypervisorProcessorCountInformation = 0x87, SystemDeviceDataInformation = 0x88, SystemDeviceDataEnumerationInformation = 0x89, SystemMemoryTopologyInformation = 0x8a, SystemMemoryChannelInformation = 0x8b, SystemBootLogoInformation = 0x8c, SystemProcessorPerformanceInformationEx = 0x8d, SystemSpare0 = 0x8e, SystemSecureBootPolicyInformation = 0x8f, SystemPageFileInformationEx = 0x90, SystemSecureBootInformation = 0x91, SystemEntropyInterruptTimingRawInformation = 0x92, SystemPortableWorkspaceEfiLauncherInformation = 0x93, SystemFullProcessInformation = 0x94, SystemKernelDebuggerInformationEx = 0x95, SystemBootMetadataInformation = 0x96, SystemSoftRebootInformation = 0x97, SystemElamCertificateInformation = 0x98, SystemOfflineDumpConfigInformation = 0x99, SystemProcessorFeaturesInformation = 0x9a, SystemRegistryReconciliationInformation = 0x9b, MaxSystemInfoClass = 0x9c,

} SYSTEM\_INFORMATION\_CLASS;

最后的 JudgeLoadDriver() 是核心函数，我们看下该函数具体是如何实现的，原理很简单。

# 1.通过MmGetSystemRoutineAddress得到动态的地址。

2.动态调用m\_NtQuerySystemInformation得到参数。

3.判断自身是否被加载，如果是输出路径。



#include <ntifs.h> #include <windef.h> #include <stdlib.h>

typedef NTSTATUS( NTQUERYSYSTEMINFORMATION)(

IN ULONG SystemInformationClass, OUT PVOID SystemInformation,

IN ULONG\_PTR SystemInformationLength,

OUT PULONG\_PTR ReturnLength OPTIONAL);

typedef struct \_SYSTEM\_MODULE\_INFORMATION { HANDLE Section;

PVOID MappedBase; PVOID Base;

ULONG Size; ULONG Flags;

USHORT LoadOrderIndex; USHORT InitOrderIndex; USHORT LoadCount; USHORT PathLength; CHAR ImageName[256];



} SYSTEM\_MODULE\_INFORMATION, PSYSTEM\_MODULE\_INFORMATION;

typedef enum \_SYSTEM\_INFORMATION\_CLASS

{

SystemBasicInformation = 0x0, SystemProcessorInformation = 0x1, SystemPerformanceInformation = 0x2, SystemTimeOfDayInformation = 0x3, SystemPathInformation = 0x4, SystemProcessInformation = 0x5, SystemCallCountInformation = 0x6, SystemDeviceInformation = 0x7, SystemProcessorPerformanceInformation = 0x8, SystemFlagsInformation = 0x9, SystemCallTimeInformation = 0xa, SystemModuleInformation = 0xb, SystemLocksInformation = 0xc,

} SYSTEM\_INFORMATION\_CLASS;

// 判断当前Driver是否加载成功

// By: LyShark

ULONG JudgeLoadDriver()

{

NTQUERYSYSTEMINFORMATION m\_NtQuerySystemInformation = NULL; UNICODE\_STRING NtQuerySystemInformation\_Name; PSYSTEM\_MODULE\_INFORMATION ModuleEntry;

ULONG\_PTR RetLength, BaseAddr, EndAddr;

ULONG ModuleNumbers, Index; NTSTATUS Status;

PVOID Buffer;

RtlInitUnicodeString(&NtQuerySystemInformation\_Name, L"NtQuerySystemInformation");

m\_NtQuerySystemInformation =

(NTQUERYSYSTEMINFORMATION)MmGetSystemRoutineAddress(&NtQuerySystemInformation\_Na me);

if (m\_NtQuerySystemInformation == NULL)

{

DbgPrint("获取NtQuerySystemInformation函数失败！\n"); return 1;

}

RetLength = 0;

Status = m\_NtQuerySystemInformation(SystemModuleInformation, NULL, 0, &RetLength);

if (Status < 0 && Status != STATUS\_INFO\_LENGTH\_MISMATCH)

{

DbgPrint("NtQuerySystemInformation调用失败！错误码是：%x\n", Status); return 1;

}

Buffer = ExAllocatePoolWithTag(NonPagedPool, RetLength, 'lysh'); if (Buffer == NULL)



{

DbgPrint("分配内存失败！\n"); return 1;

}

Status = m\_NtQuerySystemInformation(SystemModuleInformation, Buffer, RetLength, &RetLength);

if (Status < 0)

{

DbgPrint("NtQuerySystemInformation调用失败 %x\n", Status); return 1;

}

ModuleNumbers = (ULONG )Buffer;

ModuleEntry = (PSYSTEM\_MODULE\_INFORMATION)((ULONG\_PTR)Buffer + 8);

for (Index = 0; Index < ModuleNumbers; ++Index)

{

BaseAddr = (ULONG\_PTR)ModuleEntry->Base; EndAddr = BaseAddr + ModuleEntry->Size;

if (BaseAddr <= (ULONG\_PTR)JudgeLoadDriver && (ULONG\_PTR)JudgeLoadDriver

<= EndAddr)

{

DbgPrint("模块名称是：%s\n", ModuleEntry->ImageName); return 2;

}

++ModuleEntry;

}

return 0;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

DbgPrint("驱动卸载成功 \n");

}

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

DbgPrint("hello lyshark.com \n");

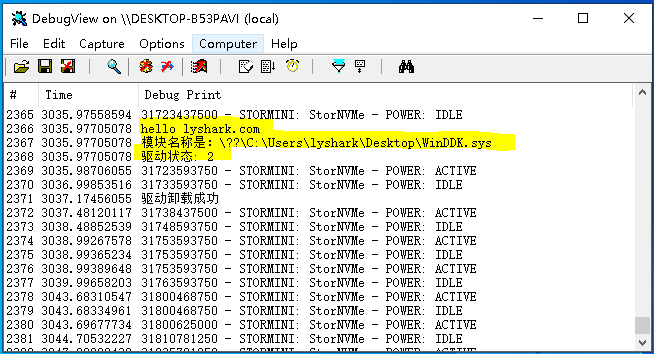
ULONG ul = JudgeLoadDriver();

DbgPrint("驱动状态: %d \n", ul);

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

# 代码运行效果如下所示：



作者： 王瑞 (LyShark)

作者邮箱： m [e@lyshark.com](mailto:e@lyshark.com)

版权声明：本博客文章与代码均为学习时整理的笔记，文章 [均为原创] 作品，转载文章请遵守

《中华人民共和国著作权法》相关法律规定或遵守《署名CC BY-ND 4.0国际》规范，合理合规携带原创出处转载，如果不携带文章出处，并恶意转载多篇原创文章被本人发现，本人保留起诉权！