在前面的博文《驱动开发：Win10内核枚举SSDT表基址》中已经教大家如何寻找SSDT表基地址了，找到后我们可根据序号获取到指定SSDT函数的原始地址，而如果需要输出所有SSDT表信息，则可以定义字符串列表，以此循环调用GetSSDTFunctionAddress()函数得到，当然在此之间也可以调用系统提供的MmGetSystemRoutineAddress()函数顺便把当前地址拿到，并通过循环方式得到完整的SSDT列表。

调用MmGetSystemRoutineAddress()得到当前地址很容易实现，只需要将函数名字符串通过RtlInitUnicodeString()格式化一下即可。

```
// 署名权
// right to sign one's name on a piece of work
// PowerBy: LyShark
// Email: me@lyshark.com

#include <ntifs.h>

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint(("驱动程序卸载成功! \n"));
}

NTSTATUS DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
{
    DbgPrint("hello lyshark.com \n");

    // 获取SSDT起源地址
    UNICODE_STRING uncode;
    RtlInitUnicodeString(&uncode, L"NtOpenFile");
    PULONGLONG source_address = MmGetSystemRoutineAddress(&uncode);
    DbgPrint("[LyShark] NtOpenFile起源地址 = %p \n", source_address);

    DriverObject->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

代码获得NtOpenFile这个函数的内存地址，输出效果如下所示：

根据上一章节的内容扩展，枚举完整SSDT表我们可以这样来实现。

```
// 署名权
// right to sign one's name on a piece of work
// PowerBy: LyShark
// Email: me@lyshark.com

#include <ntifs.h>
#pragma intrinsic(__readmsr)

typedef struct _SYSTEM_SERVICE_TABLE
{
    PVOID     ServiceTableBase;
    PVOID     ServiceCounterTableBase;
    ULONGLONG   NumberOfServices;
    PVOID     ParamTableBase;
} SYSTEM_SERVICE_TABLE, *PSYSTEM_SERVICE_TABLE;

ULONGLONG ssdt_base_aadress;
PSYSTEM_SERVICE_TABLE KeServiceDescriptorTable;

typedef UINT64(__fastcall *SCFN)(UINT64, UINT64);
SCFN scfn;

// 解密算法
VOID DecodeSSDT()
{
    UCHAR strShellCode[36] =
"\x48\x8B\xC1\x4C\x8D\x12\x8B\xF8\xC1\xEF\x07\x83\xE7\x20\x4E\x8B\x14\x17\x4D\x6
3\x1C\x82\x49\x8B\xC3\x49\xC1\xFB\x04\x4D\x03\xD3\x49\x8B\xC2\xC3";
    /*
    48:8BC1                 | mov rax,rcx                           |
rcx=index
    4C:8D12                 | lea r10,qword ptr ds:[rdx]            |
rdx=ssdt
    8BF8                    | mov edi,eax                           |
    C1EF 07                 | shr edi,7                             |
    83E7 20                 | and edi,20                            |
    4E:8B1417               | mov r10,qword ptr ds:[rdi+r10]        |
    4D:631C82               | movsxd r11,dword ptr ds:[r10+rax*4]   |
```

```
    49:8BC3                 | mov rax,r11                           |
    49:C1FB 04              | sar r11,4                             |
    4D:03D3                 | add r10,r11                           |
    49:8BC2                 | mov rax,r10                           |
    C3                      | ret                                   |
    */
    scfn = ExAllocatePool(NonPagedPool, 36);
    memcpy(scfn, strShellCode, 36);
}

// 获取 KeServiceDescriptorTable 首地址
ULONGLONG GetKeServiceDescriptorTable()
{
    // 设置起始位置
    PUCHAR StartSearchAddress = (PUCHAR)__readmsr(0xC0000082) - 0x1806FE;

    // 设置结束位置
    PUCHAR EndSearchAddress = StartSearchAddress + 0x8192;
    // DbgPrint("扫描起始地址: %p --> 扫描结束地址: %p \n", StartSearchAddress,
EndSearchAddress);

    PUCHAR ByteCode = NULL;

    UCHAR OpCodeA = 0, OpCodeB = 0, OpCodeC = 0;
    ULONGLONG addr = 0;
    ULONG templong = 0;

    for (ByteCode = StartSearchAddress; ByteCode < EndSearchAddress; ByteCode++)
    {
        // 使用MmIsAddressValid()函数检查地址是否有页面错误
        if (MmIsAddressValid(ByteCode) && MmIsAddressValid(ByteCode + 1) &&
MmIsAddressValid(ByteCode + 2))
        {
            OpCodeA = *ByteCode;
            OpCodeB = *(ByteCode + 1);
            OpCodeC = *(ByteCode + 2);

            // 对比特征值 寻找 nt!KeServiceDescriptorTable 函数地址
            // LyShark.com
            // 4c 8d 15 e5 9e 3b 00  lea r10,[nt!KeServiceDescriptorTable
(fffff802`64da4880)]
            // 4c 8d 1d de 20 3a 00  lea r11,[nt!KeServiceDescriptorTableShadow
(fffff802`64d8ca80)]
            if (OpCodeA == 0x4c && OpCodeB == 0x8d && OpCodeC == 0x15)
            {
                // 获取高位地址fffff802
                memcpy(&templong, ByteCode + 3, 4);

                // 与低位64da4880地址相加得到完整地址
                addr = (ULONGLONG)templong + (ULONGLONG)ByteCode + 7;
                return addr;
            }
        }
    }
    return  0;
```

```
}

// 得到函数相对偏移地址
ULONG GetOffsetAddress(ULONGLONG FuncAddr)
{
    ULONG dwtmp = 0;
    PULONG ServiceTableBase = NULL;
    if (KeServiceDescriptorTable == NULL)
    {
        KeServiceDescriptorTable =
(PSYSTEM_SERVICE_TABLE)GetKeServiceDescriptorTable();
    }
    ServiceTableBase = (PULONG)KeServiceDescriptorTable->ServiceTableBase;
    dwtmp = (ULONG)(FuncAddr - (ULONGLONG)ServiceTableBase);
    return dwtmp << 4;
}

// 根据序号得到函数地址
ULONGLONG GetSSDTFunctionAddress(ULONGLONG NtApiIndex)
{
    ULONGLONG ret = 0;
    if (ssdt_base_aadress == 0)
    {
        // 得到ssdt基地址
        ssdt_base_aadress = GetKeServiceDescriptorTable();
    }
    if (scfn == NULL)
    {
        DecodeSSDT();
    }
    ret = scfn(NtApiIndex, ssdt_base_aadress);
    return ret;
}

// 查询函数系统地址
ULONG_PTR QueryFunctionSystemAddress(PWCHAR name)
{
    UNICODE_STRING na;
    ULONG_PTR address;
    RtlInitUnicodeString(&na, name);
    address = (ULONG_PTR)MmGetSystemRoutineAddress(&na);
    return address;
}

VOID UnDriver(PDRIVER_OBJECT driver)
{
    DbgPrint(("驱动程序卸载成功! \n"));
}

NTSTATUS DriverEntry(PDRIVER_OBJECT DriverObject, PUNICODE_STRING RegistryPath)
{
    DbgPrint("hello lyshark.com \n");
```

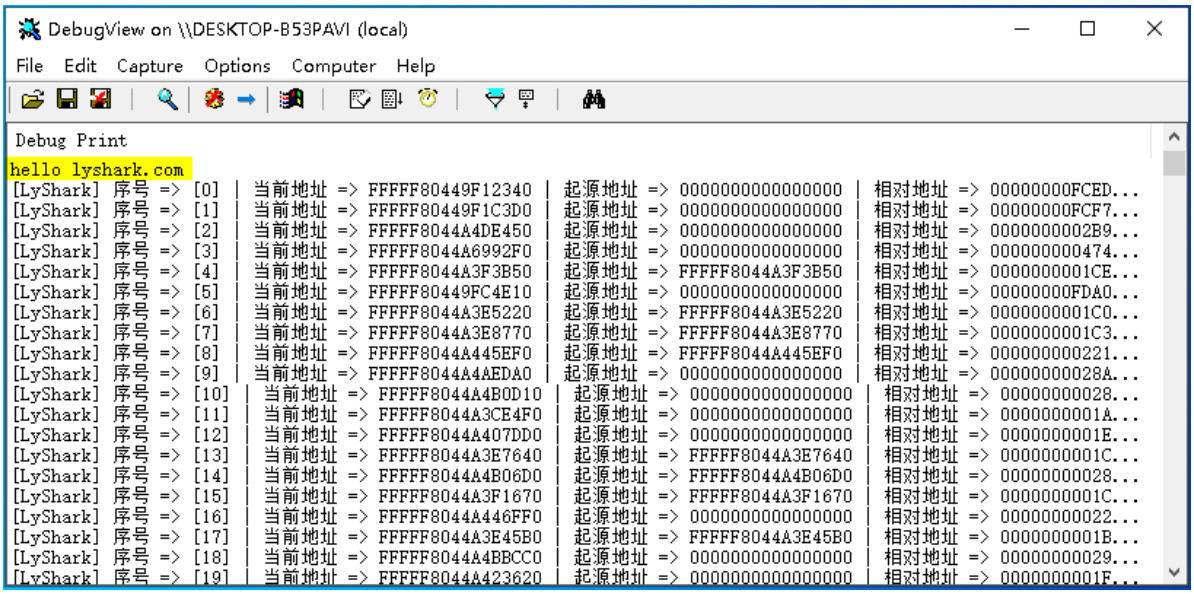
```
    char *SSDT[464] = { "NtAccessCheck", "NtWorkerFactoryWorkerReady",
"NtAcceptConnectPort", "NtMapUserPhysicalPagesScatter", "NtWaitForSingleObject",
"NtCallbackReturn", "NtReadFile", "NtDeviceIoControlFile", "NtWriteFile",
"NtRemoveIoCompletion", "NtReleaseSemaphore", "NtReplyWaitReceivePort",
"NtReplyPort", "NtSetInformationThread", "NtSetEvent", "NtClose",
"NtQueryObject", "NtQueryInformationFile", "NtOpenKey", "NtEnumerateValueKey",
"NtFindAtom", "NtQueryDefaultLocale", "NtQueryKey", "NtQueryValueKey",
"NtAllocateVirtualMemory", "NtQueryInformationProcess",
"NtWaitForMultipleObjects32", "NtWriteFileGather", "NtSetInformationProcess",
"NtCreateKey", "NtFreeVirtualMemory", "NtImpersonateClientOfPort",
"NtReleaseMutant", "NtQueryInformationToken", "NtRequestWaitReplyPort",
"NtQueryVirtualMemory", "NtOpenThreadToken", "NtQueryInformationThread",
"NtOpenProcess", "NtSetInformationFile", "NtMapViewOfSection",
"NtAccessCheckAndAuditAlarm", "NtUnmapViewOfSection",
"NtReplyWaitReceivePortEx", "NtTerminateProcess", "NtSetEventBoostPriority",
"NtReadFileScatter", "NtOpenThreadTokenEx", "NtOpenProcessTokenEx",
"NtQueryPerformanceCounter", "NtEnumerateKey", "NtOpenFile", "NtDelayExecution",
"NtQueryDirectoryFile", "NtQuerySystemInformation", "NtOpenSection",
"NtQueryTimer", "NtFsControlFile", "NtWriteVirtualMemory",
"NtCloseObjectAuditAlarm", "NtDuplicateObject", "NtQueryAttributesFile",
"NtClearEvent", "NtReadVirtualMemory", "NtOpenEvent", "NtAdjustPrivilegesToken",
"NtDuplicateToken", "NtContinue", "NtQueryDefaultUILanguage",
"NtQueueApcThread", "NtYieldExecution", "NtAddAtom", "NtCreateEvent",
"NtQueryVolumeInformationFile", "NtCreateSection", "NtFlushBuffersFile",
"NtApphelpCacheControl", "NtCreateProcessEx", "NtCreateThread",
"NtIsProcessInJob", "NtProtectVirtualMemory", "NtQuerySection",
"NtResumeThread", "NtTerminateThread", "NtReadRequestData", "NtCreateFile",
"NtQueryEvent", "NtWriteRequestData", "NtOpenDirectoryObject",
"NtAccessCheckByTypeAndAuditAlarm", "NtQuerySystemTime",
"NtWaitForMultipleObjects", "NtSetInformationObject", "NtCancelIoFile",
"NtTraceEvent", "NtPowerInformation", "NtSetValueKey", "NtCancelTimer",
"NtSetTimer", "NtAccessCheckByType", "NtAccessCheckByTypeResultList",
"NtAccessCheckByTypeResultListAndAuditAlarm",
"NtAccessCheckByTypeResultListAndAuditAlarmByHandle",
"NtAcquireProcessActivityReference", "NtAddAtomEx", "NtAddBootEntry",
"NtAddDriverEntry", "NtAdjustGroupsToken", "NtAdjustTokenClaimsAndDeviceGroups",
"NtAlertResumeThread", "NtAlertThread", "NtAlertThreadByThreadId",
"NtAllocateLocallyUniqueId", "NtAllocateReserveObject",
"NtAllocateUserPhysicalPages", "NtAllocateUuids", "NtAllocateVirtualMemoryEx",
"NtAlpcAcceptConnectPort", "NtAlpcCancelMessage", "NtAlpcConnectPort",
"NtAlpcConnectPortEx", "NtAlpcCreatePort", "NtAlpcCreatePortSection",
"NtAlpcCreateResourceReserve", "NtAlpcCreateSectionView",
"NtAlpcCreateSecurityContext", "NtAlpcDeletePortSection",
"NtAlpcDeleteResourceReserve", "NtAlpcDeleteSectionView",
"NtAlpcDeleteSecurityContext", "NtAlpcDisconnectPort",
"NtAlpcImpersonateClientContainerOfPort", "NtAlpcImpersonateClientOfPort",
"NtAlpcOpenSenderProcess", "NtAlpcOpenSenderThread", "NtAlpcQueryInformation",
"NtAlpcQueryInformationMessage", "NtAlpcRevokeSecurityContext",
"NtAlpcSendWaitReceivePort", "NtAlpcSetInformation", "NtAreMappedFilesTheSame",
"NtAssignProcessToJobObject", "NtAssociateWaitCompletionPacket",
"NtCallEnclave", "NtCancelIoFileEx", "NtCancelSynchronousIoFile",
"NtCancelTimer2", "NtCancelWaitCompletionPacket", "NtCommitComplete",
"NtCommitEnlistment", "NtCommitRegistryTransaction", "NtCommitTransaction",
"NtCompactKeys", "NtCompareObjects", "NtCompareSigningLevels",
"NtCompareTokens", "ArbPreprocessEntry", "NtCompressKey", "NtConnectPort",
```

```
"NtConvertBetweenAuxiliaryCounterAndPerformanceCounter", "ArbAddReserved",
"NtCreateDebugObject", "NtCreateDirectoryObject", "NtCreateDirectoryObjectEx",
"NtCreateEnclave", "NtCreateEnlistment", "NtCreateEventPair", "NtCreateIRTimer",
"NtCreateIoCompletion", "NtCreateJobObject", "ArbAddReserved",
"NtCreateKeyTransacted", "NtCreateKeyedEvent", "NtCreateLowBoxToken",
"NtCreateMailslotFile", "NtCreateMutant", "NtCreateNamedPipeFile",
"NtCreatePagingFile", "NtCreatePartition", "NtCreatePort",
"NtCreatePrivateNamespace", "NtCreateProcess", "NtCreateProfile",
"NtCreateProfileEx", "NtCreateRegistryTransaction", "NtCreateResourceManager",
"NtCreateSectionEx", "NtCreateSemaphore", "NtCreateSymbolicLinkObject",
"NtCreateThreadEx", "NtCreateTimer", "NtCreateTimer2", "NtCreateToken",
"NtCreateTokenEx", "NtCreateTransaction", "NtCreateTransactionManager",
"NtCreateUserProcess", "NtCreateWaitCompletionPacket", "NtCreateWaitablePort",
"NtCreateWnfStateName", "NtCreateWorkerFactory", "NtDebugActiveProcess",
"NtDebugContinue", "NtDeleteAtom", "NtDeleteBootEntry", "NtDeleteDriverEntry",
"NtDeleteFile", "NtDeleteKey", "NtDeleteObjectAuditAlarm",
"NtDeletePrivateNamespace", "NtDeleteValueKey", "NtDeleteWnfStateData",
"NtDeleteWnfStateName", "NtDisableLastKnownGood", "NtDisplayString",
"NtDrawText", "NtEnableLastKnownGood", "NtEnumerateBootEntries",
"NtEnumerateDriverEntries", "NtEnumerateSystemEnvironmentValuesEx",
"NtEnumerateTransactionObject", "NtExtendSection", "NtFilterBootOption",
"NtFilterToken", "NtFilterTokenEx", "NtFlushBuffersFileEx",
"NtFlushInstallUILanguage", "ArbPreprocessEntry", "NtFlushKey",
"NtFlushProcessWriteBuffers", "NtFlushVirtualMemory", "NtFlushWriteBuffer",
"NtFreeUserPhysicalPages", "NtFreezeRegistry", "NtFreezeTransactions",
"NtGetCachedSigningLevel", "NtGetCompleteWnfStateSubscription",
"NtGetContextThread", "NtGetCurrentProcessorNumber",
"NtGetCurrentProcessorNumberEx", "NtGetDevicePowerState",
"NtGetMUIRegistryInfo", "NtGetNextProcess", "NtGetNextThread",
"NtGetNlsSectionPtr", "NtGetNotificationResourceManager", "NtGetWriteWatch",
"NtImpersonateAnonymousToken", "NtImpersonateThread", "NtInitializeEnclave",
"NtInitializeNlsFiles", "NtInitializeRegistry", "NtInitiatePowerAction",
"NtIsSystemResumeAutomatic", "NtIsUILanguageComitted", "NtListenPort",
"NtLoadDriver", "NtLoadEnclaveData", "NtLoadKey", "NtLoadKey2", "NtLoadKeyEx",
"NtLockFile", "NtLockProductActivationKeys", "NtLockRegistryKey",
"NtLockVirtualMemory", "NtMakePermanentObject", "NtMakeTemporaryObject",
"NtManageHotPatch", "NtManagePartition", "NtMapCMFModule",
"NtMapUserPhysicalPages", "NtMapViewOfSectionEx", "NtModifyBootEntry",
"NtModifyDriverEntry", "NtNotifyChangeDirectoryFile",
"NtNotifyChangeDirectoryFileEx", "NtNotifyChangeKey",
"NtNotifyChangeMultipleKeys", "NtNotifyChangeSession", "NtOpenEnlistment",
"NtOpenEventPair", "NtOpenIoCompletion", "NtOpenJobObject", "NtOpenKeyEx",
"NtOpenKeyTransacted", "NtOpenKeyTransactedEx", "NtOpenKeyedEvent",
"NtOpenMutant", "NtOpenObjectAuditAlarm", "NtOpenPartition",
"NtOpenPrivateNamespace", "NtOpenProcessToken", "NtOpenRegistryTransaction",
"NtOpenResourceManager", "NtOpenSemaphore", "NtOpenSession",
"NtOpenSymbolicLinkObject", "NtOpenThread", "NtOpenTimer", "NtOpenTransaction",
"NtOpenTransactionManager", "NtPlugPlayControl", "NtPrePrepareComplete",
"NtPrePrepareEnlistment", "NtPrepareComplete", "NtPrepareEnlistment",
"NtPrivilegeCheck", "NtPrivilegeObjectAuditAlarm",
"NtPrivilegedServiceAuditAlarm", "NtPropagationComplete", "NtPropagationFailed",
"NtPulseEvent", "NtQueryAuxiliaryCounterFrequency", "NtQueryBootEntryOrder",
"NtQueryBootOptions", "NtQueryDebugFilterState", "NtQueryDirectoryFileEx",
"NtQueryDirectoryObject", "NtQueryDriverEntryOrder", "NtQueryEaFile",
"NtQueryFullAttributesFile", "NtQueryInformationAtom",
```

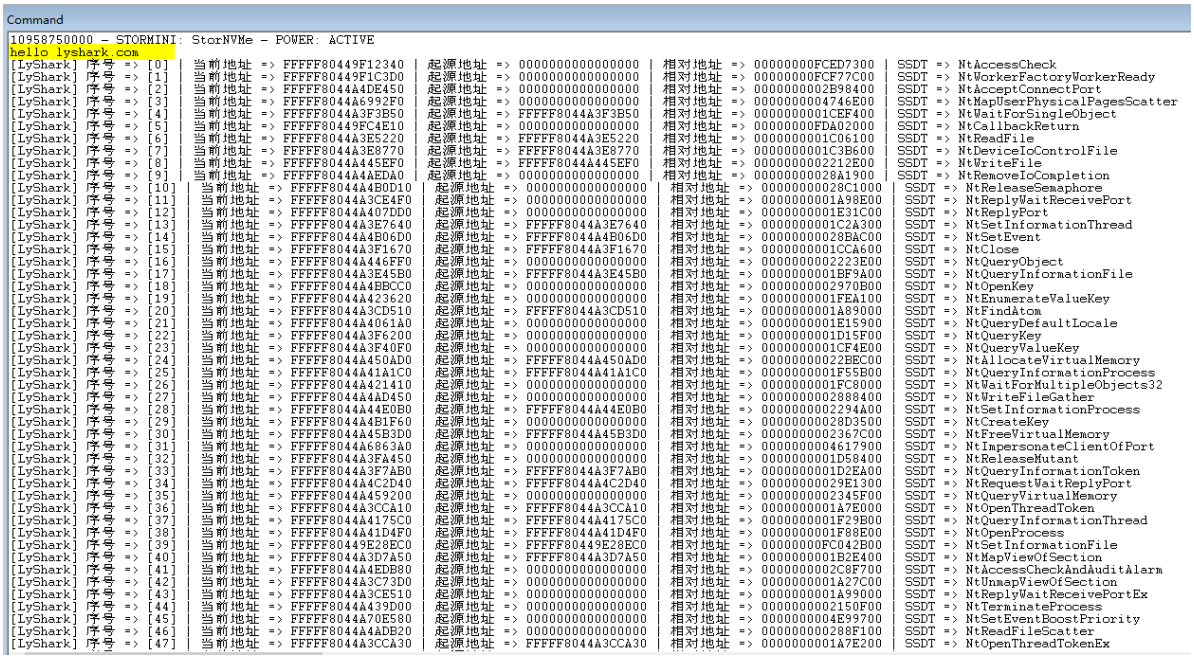
```
"NtQueryInformationByName", "NtQueryInformationEnlistment",
"NtQueryInformationJobObject", "NtQueryInformationPort",
"NtQueryInformationResourceManager", "NtQueryInformationTransaction",
"NtQueryInformationTransactionManager", "NtQueryInformationWorkerFactory",
"NtQueryInstallUILanguage", "NtQueryIntervalProfile", "NtQueryIoCompletion",
"NtQueryLicenseValue", "NtQueryMultipleValueKey", "NtQueryMutant",
"NtQueryOpenSubKeys", "NtQueryOpenSubKeysEx",
"CmpCleanUpHigherLayerKcbCachesPreCallback", "NtQueryQuotaInformationFile",
"NtQuerySecurityAttributesToken", "NtQuerySecurityObject",
"NtQuerySecurityPolicy", "NtQuerySemaphore", "NtQuerySymbolicLinkObject",
"NtQuerySystemEnvironmentValue", "NtQuerySystemEnvironmentValueEx",
"NtQuerySystemInformationEx", "NtQueryTimerResolution", "NtQueryWnfStateData",
"NtQueryWnfStateNameInformation", "NtQueueApcThreadEx", "NtRaiseException",
"NtRaiseHardError", "NtReadOnlyEnlistment", "NtRecoverEnlistment",
"NtRecoverResourceManager", "NtRecoverTransactionManager",
"NtRegisterProtocolAddressInformation", "NtRegisterThreadTerminatePort",
"NtReleaseKeyedEvent", "NtReleaseWorkerFactoryWorker", "NtRemoveIoCompletionEx",
"NtRemoveProcessDebug", "NtRenameKey", "NtRenameTransactionManager",
"NtReplaceKey", "NtReplacePartitionUnit", "NtReplyWaitReplyPort",
"NtRequestPort", "NtResetEvent", "NtResetWriteWatch", "NtRestoreKey",
"NtResumeProcess", "NtRevertContainerImpersonation", "NtRollbackComplete",
"NtRollbackEnlistment", "NtRollbackRegistryTransaction",
"NtRollbackTransaction", "NtRollforwardTransactionManager", "NtSaveKey",
"NtSaveKeyEx", "NtSaveMergedKeys", "NtSecureConnectPort", "NtSerializeBoot",
"NtSetBootEntryOrder", "NtSetBootOptions", "NtSetCachedSigningLevel",
"NtSetCachedSigningLevel2", "NtSetContextThread", "NtSetDebugFilterState",
"NtSetDefaultHardErrorPort", "NtSetDefaultLocale", "NtSetDefaultUILanguage",
"NtSetDriverEntryOrder", "NtSetEaFile", "NtSetHighEventPair",
"NtSetHighWaitLowEventPair", "NtSetIRTimer", "NtSetInformationDebugObject",
"NtSetInformationEnlistment", "NtSetInformationJobObject",
"NtSetInformationKey", "NtSetInformationResourceManager",
"NtSetInformationSymbolicLink", "NtSetInformationToken",
"NtSetInformationTransaction", "NtSetInformationTransactionManager",
"NtSetInformationVirtualMemory", "NtSetInformationWorkerFactory",
"NtSetIntervalProfile", "NtSetIoCompletion", "NtSetIoCompletionEx",
"BvgaSetVirtualFrameBuffer", "NtSetLowEventPair", "NtSetLowWaitHighEventPair",
"NtSetQuotaInformationFile", "NtSetSecurityObject",
"NtSetSystemEnvironmentValue", "NtSetSystemEnvironmentValueEx",
"NtSetSystemInformation", "NtSetSystemPowerState", "NtSetSystemTime",
"NtSetThreadExecutionState", "NtSetTimer2", "NtSetTimerEx",
"NtSetTimerResolution", "NtSetUuidSeed", "NtSetVolumeInformationFile",
"NtSetWnfProcessNotificationEvent", "NtShutdownSystem",
"NtShutdownWorkerFactory", "NtSignalAndWaitForSingleObject",
"NtSinglePhaseReject", "NtStartProfile", "NtStopProfile",
"NtSubscribeWnfStateChange", "NtSuspendProcess", "NtSuspendThread",
"NtSystemDebugControl", "NtTerminateEnclave", "NtTerminateJobObject",
"NtTestAlert", "NtThawRegistry", "NtThawTransactions", "NtTraceControl",
"NtTranslateFilePath", "NtUmsThreadYield", "NtUnloadDriver", "NtUnloadKey",
"NtUnloadKey2", "NtUnloadKeyEx", "NtUnlockFile", "NtUnlockVirtualMemory",
"NtUnmapViewOfSectionEx", "NtUnsubscribeWnfStateChange", "NtUpdateWnfStateData",
"NtVdmControl", "NtWaitForAlertByThreadId", "NtWaitForDebugEvent",
"NtWaitForKeyedEvent", "NtWaitForWorkViaWorkerFactory", "NtWaitHighEventPair",
"NtWaitLowEventPair" };

    for (size_t lyshark = 0; lyshark < 464; lyshark++)
```

```
    {
        // 获取起源地址
        ANSI_STRING ansi = { 0 };
        UNICODE_STRING uncode = { 0 };

        ULONGLONG ssdt_address = GetKeServiceDescriptorTable();
        // DbgPrint("SSDT基地址 = %p \n", ssdt_address);

        // 根据序号得到函数地址
        ULONGLONG address = GetSSDTFunctionAddress(lyshark);
        ULONG offset = GetOffsetAddress(address);

        RtlInitAnsiString(&ansi, SSDT[lyshark]);
        RtlAnsiStringToUnicodeString(&uncode, &ansi, TRUE);
        PULONGLONG source_address = MmGetSystemRoutineAddress(&uncode);
        DbgPrint("[LyShark] 序号 => [%d] | 当前地址 => %p | 起源地址 => %p | 相对地址
=> %p | SSDT => %s \n", lyshark, address, source_address, offset,
SSDT[lyshark]);
    }

    DriverObject->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

我们运行这段程序，即可得到整个系统中所有的SSDT表地址信息；



在WinDBG中可看到完整的输出内容，当然有些函数没有被导出，起源地址是拿不到的。



本书作者：王瑞 (LyShark)

作者邮箱：me@lyshark.com

作者博客：https://lyshark.cnblogs.com

团队首页：www.lyshark.com