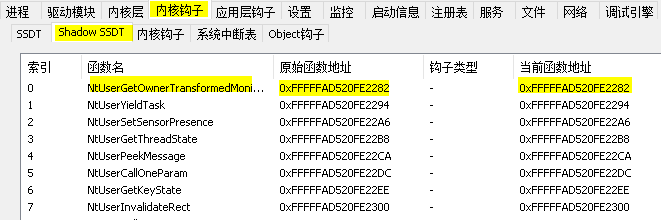
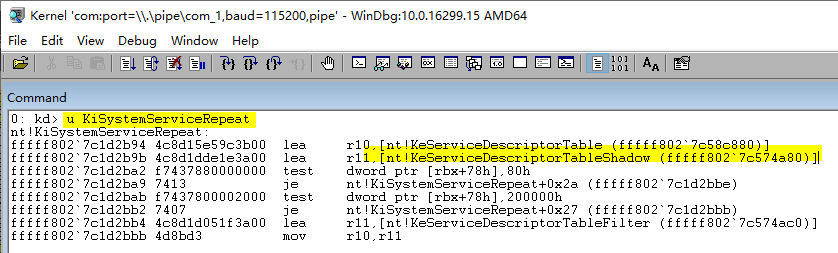
在笔者上一篇文章 《驱动开发：Win10枚举完整SSDT地址表》 实现了针对 SSDT 表的枚举功能，本章继续实现对 SSSDT 表的枚举， ShadowSSDT 中文名 影子系统服务描述表 ，SSSDT其主要的作用是管理系统中的图形化界面，其 Win32 子系统的内核实现是 Win32k.sys 驱动，属于GUI线程的一部分，其自身没有导出表，枚举 SSSDT 表其与 SSDT 原理基本一致。

# 如下是闭源ARK工具的枚举效果:



首先需要找到 SSSDT 表的位置，通过 《驱动开发：Win10内核枚举SSDT表基址》 文章中的分析可知，

SSSDT就在SSDT的下面，只需要枚举 4c8d1dde1e3a00 特征即可，如果你找不到上一篇具体分析流程了，那么多半你是看到了转载文章。



先实现第一个功能，得到 SSSDT 表的基地址以及 SSDT 函数个数，完整代码如下所示。



// 署名权

// right to sign one's name on a piece of work

// PowerBy: LyShark

// Email: [me@lyshark.com](mailto: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;

PSYSTEM\_SERVICE\_TABLE KeServiceDescriptorTableShadow = 0; ULONG64 ul64W32pServiceTable = 0;

// 获取 KeServiceDescriptorTableShadow 首地址



ULONGLONG GetKeServiceDescriptorTableShadow()

{

// 设置起始位置

PUCHAR StartSearchAddress = (PUCHAR) readmsr(0xC0000082) - 0x1808FE;

// 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 kd> u KiSystemServiceRepeat nt!KiSystemServiceRepeat:

fffff802`7c1d2b94 4c8d15e59c3b00 lea r10,

[nt!KeServiceDescriptorTable (fffff802`7c58c880)]

fffff802`7c1d2b9b 4c8d1dde1e3a00 lea r11, [nt!KeServiceDescriptorTableShadow (fffff802`7c574a80)]

fffff802`7c1d2ba2 f7437880000000 test dword ptr

[rbx+78h],80h

fffff802`7c1d2ba9 7413 je

nt!KiSystemServiceRepeat+0x2a (fffff802`7c1d2bbe)

fffff802`7c1d2bab f7437800002000 test dword ptr [rbx+78h],200000h

fffff802`7c1d2bb2 7407 je

nt!KiSystemServiceRepeat+0x27 (fffff802`7c1d2bbb)

fffff802`7c1d2bb4 4c8d1d051f3a00 lea r11, [nt!KeServiceDescriptorTableFilter (fffff802`7c574ac0)]

fffff802`7c1d2bbb 4d8bd3 mov r10,r11

/

if (OpCodeA == 0x4c && OpCodeB == 0x8d && OpCodeC == 0x1d)

{

// 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SSSDT个数

ULONGLONG GetSSSDTCount()

{

PSYSTEM\_SERVICE\_TABLE pWin32k;

ULONGLONG W32pServiceTable;

pWin32k = (PSYSTEM\_SERVICE\_TABLE)((ULONG64)KeServiceDescriptorTableShadow + sizeof(SYSTEM\_SERVICE\_TABLE));

W32pServiceTable = (ULONGLONG)(pWin32k->ServiceTableBase);

// DbgPrint("Count => %d \n", pWin32k->NumberOfServices);

return pWin32k->NumberOfServices;

}

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");

KeServiceDescriptorTableShadow = (PSYSTEM\_SERVICE\_TABLE)GetKeServiceDescriptorTableShadow();

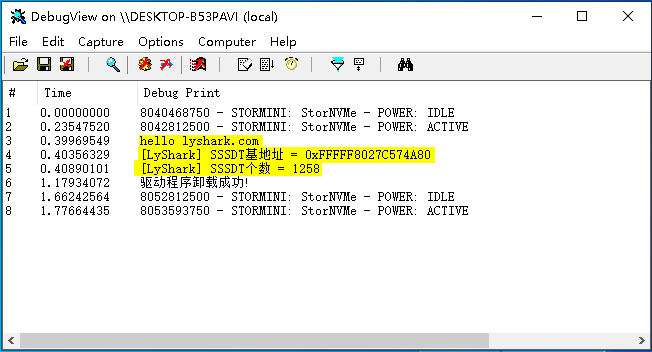
DbgPrint("[LyShark] SSSDT基地址 = 0x%p \n", KeServiceDescriptorTableShadow); ULONGLONG count = GetSSSDTCount();

DbgPrint("[LyShark] SSSDT个数 = %d \n", count);

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

}

# 这段代码运行后即可得到 SSSDT 表基地址，以及该表中函数个数。



在此基础之上增加枚举计算过程即可，完整源代码如下所示。

SSSDT 函数起始index是 0x1000 ，但 W32pServiceTable 是从基址开始记录的，这个误差则需要

(index-0x1000) 来得到，至于 +4 则是下一个元素与上一个元素的偏移。

# 计算公式：

W32pServiceTable + 4 \* (index-0x1000)



// 署名权

// right to sign one's name on a piece of work

// PowerBy: LyShark

// Email: [me@lyshark.com](mailto: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;

PSYSTEM\_SERVICE\_TABLE KeServiceDescriptorTableShadow = 0; ULONG64 ul64W32pServiceTable = 0;

// 获取 KeServiceDescriptorTableShadow 首地址

ULONGLONG GetKeServiceDescriptorTableShadow()

{

// 设置起始位置

PUCHAR StartSearchAddress = (PUCHAR) readmsr(0xC0000082) - 0x1808FE;

// 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 kd> u KiSystemServiceRepeat nt!KiSystemServiceRepeat:

fffff802`7c1d2b94 4c8d15e59c3b00 lea r10,

[nt!KeServiceDescriptorTable (fffff802`7c58c880)] fffff802`7c1d2b9b 4c8d1dde1e3a00 lea r11,

[nt!KeServiceDescriptorTableShadow (fffff802`7c574a80)]

fffff802`7c1d2ba2 f7437880000000 test dword ptr [rbx+78h],80h fffff802`7c1d2ba9 7413 je

nt!KiSystemServiceRepeat+0x2a (fffff802`7c1d2bbe) fffff802`7c1d2bab f7437800002000 test dword ptr

[rbx+78h],200000h

fffff802`7c1d2bb2 7407 je nt!KiSystemServiceRepeat+0x27 (fffff802`7c1d2bbb)

fffff802`7c1d2bb4 4c8d1d051f3a00 lea r11, [nt!KeServiceDescriptorTableFilter (fffff802`7c574ac0)]

fffff802`7c1d2bbb 4d8bd3 mov r10,r11

/

if (OpCodeA == 0x4c && OpCodeB == 0x8d && OpCodeC == 0x1d)

{

// 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SSSDT个数

ULONGLONG GetSSSDTCount()

{

PSYSTEM\_SERVICE\_TABLE pWin32k;

ULONGLONG W32pServiceTable;



pWin32k = (PSYSTEM\_SERVICE\_TABLE)((ULONG64)KeServiceDescriptorTableShadow + sizeof(SYSTEM\_SERVICE\_TABLE));

W32pServiceTable = (ULONGLONG)(pWin32k->ServiceTableBase);

// DbgPrint("Count => %d \n", pWin32k->NumberOfServices);

return pWin32k->NumberOfServices;

}

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");

KeServiceDescriptorTableShadow = (PSYSTEM\_SERVICE\_TABLE)GetKeServiceDescriptorTableShadow();

DbgPrint("[LyShark] SSSDT基地址 = 0x%p \n", KeServiceDescriptorTableShadow); ULONGLONG count = GetSSSDTCount();

DbgPrint("[LyShark] SSSDT个数 = %d \n", count);

// 循环枚举SSSDT

for (size\_t Index = 0; Index < count; Index++)

{

PSYSTEM\_SERVICE\_TABLE pWin32k;

ULONGLONG W32pServiceTable;

pWin32k = (PSYSTEM\_SERVICE\_TABLE)

((ULONG64)KeServiceDescriptorTableShadow + sizeof(SYSTEM\_SERVICE\_TABLE)); W32pServiceTable = (ULONGLONG)(pWin32k->ServiceTableBase);

// 获取SSSDT地址

//ln win32k!W32pServiceTable+((poi(win32k!W32pServiceTable+4 (1- 1000))&0x00000000`ffffffff)>>4)-10000000

//u win32k!W32pServiceTable+((poi(win32k!W32pServiceTable+4 (Index-

0x1000))&0x00000000`ffffffff)>>4)-0x10000000

//u poi(win32k!W32pServiceTable+4 (1-0x1000))

//u poi(win32k!W32pServiceTable+4 (1-0x1000))&0x00000000`ffffffff

//u (poi(win32k!W32pServiceTable+4 (1-0x1000))&0x00000000`ffffffff)>>4

//u win32k!W32pServiceTable+((poi(win32k!W32pServiceTable+4 (1- 0x1000))&0x00000000`ffffffff)>>4)-0x10000000

ULONGLONG qword\_temp = 0; LONG dw = 0;

// SSSDT 下标从1000开始，而W32pServiceTable是从0开始



// + 4 则是每次向下4字节就是下一个地址

qword\_temp = W32pServiceTable + 4 (Index - 0x1000);

dw = (PLONG)qword\_temp;

// dw = qword\_temp & 0x00000000ffffffff; dw = dw >> 4;

qword\_temp = W32pServiceTable + (LONG64)dw;

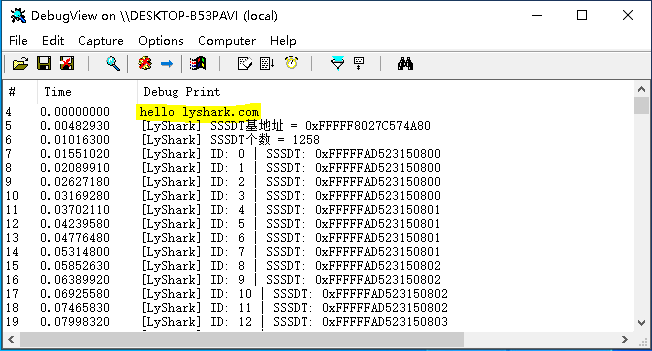
DbgPrint("[LyShark] ID: %d | SSSDT: 0x%p \n", Index, qword\_temp);

}

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

}

# 枚举效果如下所示（存在问题）:



注这一步必须要在GUI线程中执行，否则会异常，建议将枚举过程写成DLL文件，注入到 explorer.exe

# 进程内执行。

本书作者： 王瑞 (LyShark)

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

作者博客： h ttps://lyshark.cnblogs.com

团队首页： w ww.lyshark.com