在笔者上一篇文章 《驱动开发：内核监视LoadImage映像回调》 中 LyShark 简单介绍了如何通过

PsSetLoadImageNotifyRoutine 函数注册回调来 监视驱动 模块的加载，注意我这里用的是 监视 而不是 监控 之所以是监视而不是监控那是因为 PsSetLoadImageNotifyRoutine 无法实现参数控制，而如果我们想要控制特定驱动的加载则需要自己做一些事情来实现，如下 LyShark 将解密如何实现屏蔽特定驱动的加载。

要想实现 驱动屏蔽 其原理很简单，通过 ImageInfo->ImageBase 得到镜像基地址，然后调用

GetDriverEntryByImageBase 函数来得到程序的入口地址，找NT头的 OptionalHeader 节点，该节点里面就是被加载驱动入口，通过汇编在驱动头部写入 ret 返回指令，即可实现屏蔽加载特定驱动文件。

原理其实很容易理解，如果我们需要实现则只需要在 《驱动开发：内核监视LoadImage映像回调》 这篇文章的代码上稍加改进即可，当检测到 lyshark.sys 驱动加载时，直接跳转到入口处快速写入一个 Ret 让驱动返回即可，至于如何写出指令的问题如果不懂建议回头看看 《驱动开发：内核CR3切换读写内存》 文章中是如何读写内存的，这段代码实现如下所示。



// 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 <ntddk.h> #include <intrin.h> #include <ntimage.h>

PVOID GetDriverEntryByImageBase(PVOID ImageBase)

{

PIMAGE\_DOS\_HEADER pDOSHeader; PIMAGE\_NT\_HEADERS64 pNTHeader;

PVOID pEntryPoint;

pDOSHeader = (PIMAGE\_DOS\_HEADER)ImageBase;

pNTHeader = (PIMAGE\_NT\_HEADERS64)((ULONG64)ImageBase + pDOSHeader-

>e\_lfanew);

pEntryPoint = (PVOID)((ULONG64)ImageBase + pNTHeader-

>OptionalHeader.AddressOfEntryPoint); return pEntryPoint;

}

VOID UnicodeToChar(PUNICODE\_STRING dst, char src)

{

ANSI\_STRING string; RtlUnicodeStringToAnsiString(&string, dst, TRUE); strcpy(src, string.Buffer); RtlFreeAnsiString(&string);

}

// 使用开关写保护需要在[C/C++]->[优化]->启用内部函数

// 关闭写保护

KIRQL WPOFFx64()

{

KIRQL irql = KeRaiseIrqlToDpcLevel(); UINT64 cr0 = readcr0();

cr0 &= 0xfffffffffffeffff;

\_disable();

writecr0(cr0); return irql;



}

// 开启写保护

void WPONx64(KIRQL irql)

{

UINT64 cr0 = readcr0(); cr0 |= 0x10000;

\_enable();

writecr0(cr0); KeLowerIrql(irql);

}

BOOLEAN DenyLoadDriver(PVOID DriverEntry)

{

UCHAR fuck[] = "\xB8\x22\x00\x00\xC0\xC3"; KIRQL kirql;

/ 在模块开头写入以下汇编指令

Mov eax,c0000022h ret

/

if (DriverEntry == NULL) return FALSE; kirql = WPOFFx64();

memcpy(DriverEntry, fuck, sizeof(fuck) / sizeof(fuck[0])); WPONx64(kirql);

return TRUE;

}

VOID MyLySharkComLoadImageNotifyRoutine(PUNICODE\_STRING FullImageName, HANDLE ModuleStyle, PIMAGE\_INFO ImageInfo)

{

PVOID pDrvEntry;

char szFullImageName[256] = { 0 };

// MmIsAddress 验证地址可用性

if (FullImageName != NULL && MmIsAddressValid(FullImageName))

{

// ModuleStyle为零表示加载sys if (ModuleStyle == 0)

{

pDrvEntry = GetDriverEntryByImageBase(ImageInfo->ImageBase); UnicodeToChar(FullImageName, szFullImageName);

if (strstr(\_strlwr(szFullImageName), "lyshark.sys"))

{

DbgPrint("[LyShark] 拦截SYS内核模块：%s", szFullImageName); DenyLoadDriver(pDrvEntry);

}

}

}

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

PsRemoveLoadImageNotifyRoutine((PLOAD\_IMAGE\_NOTIFY\_ROUTINE)MyLySharkComLoadImage NotifyRoutine);

DbgPrint("驱动卸载完成...");

}

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

{

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

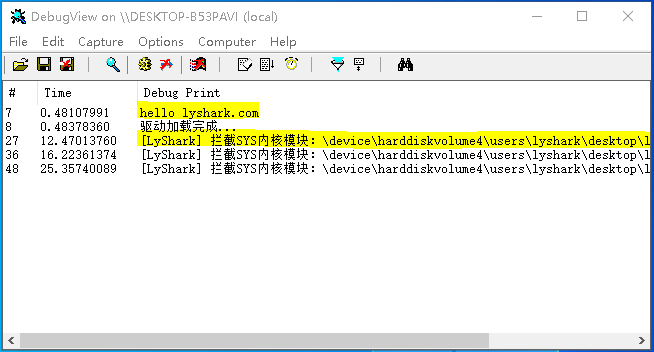
PsSetLoadImageNotifyRoutine((PLOAD\_IMAGE\_NOTIFY\_ROUTINE)MyLySharkComLoadImageNot ifyRoutine);

DbgPrint("驱动加载完成...");

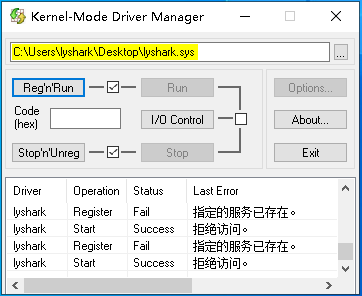
Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

首先运行我们的驱动，然后我们接着加载 lyshark.sys 则你会发现驱动被拦截了。



# 我们看下驱动加载器，提示的信息是拒绝访问，因为这个驱动其实是加载了的，只是入口处被填充了返 回而已。



除了使用 Ret 强制返回的方法意外，屏蔽驱动加载还可以使用另一种方式实现禁用模块加载，例如当驱动被加载首先回调函数内可以接收到，当接收到以后直接调用 MmUnmapViewOfSection 函数强制卸载掉即可，如果使用这种方法实现则这段代码需要改进成如下样子。



// 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> #include <ntimage.h> #include <intrin.h>

NTSTATUS MmUnmapViewOfSection(PEPROCESS Process, PVOID BaseAddress); NTSTATUS SetNotifyRoutine();

NTSTATUS RemoveNotifyRoutine();

VOID LoadImageNotifyRoutine(PUNICODE\_STRING FullImageName, HANDLE ProcessId, PIMAGE\_INFO ImageInfo);

NTSTATUS U2C(PUNICODE\_STRING pustrSrc, PCHAR pszDest, ULONG ulDestLength);

VOID ThreadProc(\_In\_ PVOID StartContext);

// 拒绝加载驱动

NTSTATUS DenyLoadDriver(PVOID pImageBase);

// 拒绝加载DLL模块

NTSTATUS DenyLoadDll(HANDLE ProcessId, PVOID pImageBase);

typedef struct \_MY\_DATA

{

HANDLE ProcessId; PVOID pImageBase;

}MY\_DATA, PMY\_DATA;

// 设置消息回调

NTSTATUS SetNotifyRoutine()

{

NTSTATUS status = STATUS\_SUCCESS;

status = PsSetLoadImageNotifyRoutine(LoadImageNotifyRoutine);

return status;

}

// 关闭消息回调

NTSTATUS RemoveNotifyRoutine()

{

NTSTATUS status = STATUS\_SUCCESS;

status = PsRemoveLoadImageNotifyRoutine(LoadImageNotifyRoutine); return status;

}

VOID LoadImageNotifyRoutine(PUNICODE\_STRING FullImageName, HANDLE ProcessId, PIMAGE\_INFO ImageInfo)

{

DbgPrint("PID: %d --> 完整路径: %wZ --> 大小: %d --> 基地址: 0x%p \n", ProcessId, FullImageName, ImageInfo->ImageSize, ImageInfo->ImageBase);

HANDLE hThread = NULL; CHAR szTemp[1024] = { 0 };

U2C(FullImageName, szTemp, 1024);

if (NULL != strstr(szTemp, "lyshark.sys"))

{

// EXE或者DLL

if (0 != ProcessId)

{

// 创建多线程 延时1秒钟后再卸载模块

PMY\_DATA pMyData = ExAllocatePool(NonPagedPool, sizeof(MY\_DATA)); pMyData->ProcessId = ProcessId;

pMyData->pImageBase = ImageInfo->ImageBase;

PsCreateSystemThread(&hThread, 0, NULL, NtCurrentProcess(), NULL, ThreadProc, pMyData);

DbgPrint("[LyShark] 禁止加载DLL文件 \n");

}

// 驱动

else

{

DenyLoadDriver(ImageInfo->ImageBase);

DbgPrint("[LyShark] 禁止加载SYS驱动文件 \n");

}

}

}

// 拒绝加载驱动

NTSTATUS DenyLoadDriver(PVOID pImageBase)

{

NTSTATUS status = STATUS\_SUCCESS; PMDL pMdl = NULL;

PVOID pVoid = NULL;

ULONG ulShellcodeLength = 16;

UCHAR pShellcode[16] = { 0xB8, 0x22, 0x00, 0x00, 0xC0, 0xC3, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90, 0x90 };

PIMAGE\_DOS\_HEADER pDosHeader = pImageBase;

PIMAGE\_NT\_HEADERS pNtHeaders = (PIMAGE\_NT\_HEADERS)((PUCHAR)pDosHeader +

pDosHeader->e\_lfanew);

PVOID pDriverEntry = (PVOID)((PUCHAR)pDosHeader + pNtHeaders-



>OptionalHeader.AddressOfEntryPoint);

pMdl = MmCreateMdl(NULL, pDriverEntry, ulShellcodeLength); MmBuildMdlForNonPagedPool(pMdl);

pVoid = MmMapLockedPages(pMdl, KernelMode);

RtlCopyMemory(pVoid, pShellcode, ulShellcodeLength); MmUnmapLockedPages(pVoid, pMdl);

IoFreeMdl(pMdl);

return status;

}

// 调用 MmUnmapViewOfSection 函数来卸载已经加载的 DLL 模块

NTSTATUS DenyLoadDll(HANDLE ProcessId, PVOID pImageBase)

{

NTSTATUS status = STATUS\_SUCCESS; PEPROCESS pEProcess = NULL;

status = PsLookupProcessByProcessId(ProcessId, &pEProcess); if (!NT\_SUCCESS(status))

{

return status;

}

// 卸载模块

status = MmUnmapViewOfSection(pEProcess, pImageBase); if (!NT\_SUCCESS(status))

{

return status;

}

return status;

}

VOID ThreadProc(\_In\_ PVOID StartContext)

{

PMY\_DATA pMyData = (PMY\_DATA)StartContext; LARGE\_INTEGER liTime = { 0 };

// 延时 1 秒 负值表示相对时间

liTime.QuadPart = -10 1000 1000; KeDelayExecutionThread(KernelMode, FALSE, &liTime);

// 卸载

DenyLoadDll(pMyData->ProcessId, pMyData->pImageBase);

ExFreePool(pMyData);

}

NTSTATUS U2C(PUNICODE\_STRING pustrSrc, PCHAR pszDest, ULONG ulDestLength)

{

NTSTATUS status = STATUS\_SUCCESS;

ANSI\_STRING strTemp;

RtlZeroMemory(pszDest, ulDestLength);

RtlUnicodeStringToAnsiString(&strTemp, pustrSrc, TRUE); if (ulDestLength > strTemp.Length)

{

RtlCopyMemory(pszDest, strTemp.Buffer, strTemp.Length);

}

RtlFreeAnsiString(&strTemp);

return status;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

PsRemoveLoadImageNotifyRoutine((PLOAD\_IMAGE\_NOTIFY\_ROUTINE)RemoveNotifyRoutine); DbgPrint("驱动卸载完成...");

}

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

{

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

PsSetLoadImageNotifyRoutine((PLOAD\_IMAGE\_NOTIFY\_ROUTINE)SetNotifyRoutine); DbgPrint("驱动加载完成...");

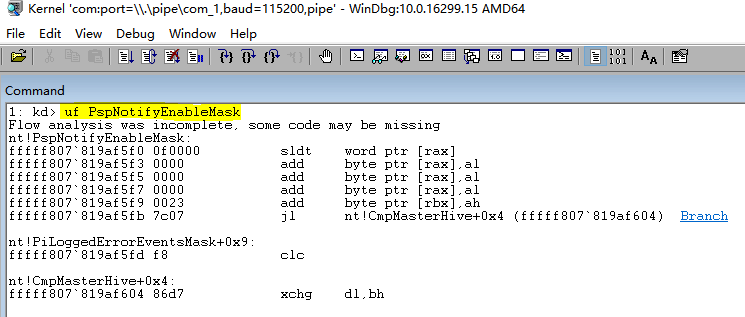
Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

# 加载这段驱动程序，当有DLL文件被加载后，则会强制弹出，从而实现屏蔽模块加载的作用。

当然用 LoadImage 回调做监控并不靠谱，因为它很容易被绕过，其实系统里存在一个开关，叫做

PspNotifyEnableMask 如果它的值被设置为 0 ，那么所有的相关操作都不会经过回调，所有回调都会失效。



# 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！