在笔者上一篇文章 《驱动开发：内核枚举进程与线程ObCall回调》 简单介绍了如何枚举系统中已经存在的

进程与线程 回调，本章 LyShark 将通过对象回调实现对进程线程的 句柄 监控，在内核中提供了

ObRegisterCallbacks 回调，使用这个内核 回调 函数，可注册一个 对象 回调，不过目前该函数 只能 监控进程与线程句柄操作，通过监控进程或线程句柄，可实现保护指定进程线程不被终止的目的。

由于目前 对象回调 只能监控进程与线程，而这个监控是通过 ObjectType 这么一个成员控制的，如果成员是 PsProcessType 则代表监控进程，反之 PsThreadType 则是监控线程，无论监控进程还是线程都调用 ObRegisterCallbacks 这个函数来完成注册。

函数 ObRegisterCallbacks 其微软对他的定义是这样的，用户传入 OB\_OPERATION\_REGISTRATION 结构，以及 OB\_CALLBACK\_REGISTRATION 回调结构，其中 PreOperation 则是传入的回调函数，也是最重要的，其次是 ObjectType 指定成进程回调。



NTSTATUS ObRegisterCallbacks(

[in] POB\_CALLBACK\_REGISTRATION CallbackRegistration, [out] PVOID RegistrationHandle

);

首先来实现一个检测的案例，注册一个进程回调对象 MyLySharkComObjectCallBack ，通过

ObRegisterCallbacks 注册的回调只需要传入一个填充好的 OB\_CALLBACK\_REGISTRATION 回调结构体，以及一个全局句柄即可，这个全局句柄的作用仅仅只是在程序结束时，调用

ObUnRegisterCallbacks 卸载监控而已，实现代码如下所示。



// 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 <ntddk.h> #include <ntstrsafe.h>

PVOID Globle\_Object\_Handle;

// 绕过签名检测

void BypassCheckSign(PDRIVER\_OBJECT pDriverObj)

{

typedef struct \_LDR\_DATA

{

struct \_LIST\_ENTRY InLoadOrderLinks; struct \_LIST\_ENTRY InMemoryOrderLinks;

struct \_LIST\_ENTRY InInitializationOrderLinks; VOID DllBase;

VOID EntryPoint;

ULONG32 SizeOfImage;

UINT8 \_PADDING0\_[0x4];

struct \_UNICODE\_STRING FullDllName; struct \_UNICODE\_STRING BaseDllName; ULONG32 Flags;

}LDR\_DATA, PLDR\_DATA;

PLDR\_DATA ldr;

ldr = (PLDR\_DATA)(pDriverObj->DriverSection);

ldr->Flags |= 0x20;

}

// 自定义回调

OB\_PREOP\_CALLBACK\_STATUS MyLySharkComObjectCallBack(PVOID RegistrationContext, POB\_PRE\_OPERATION\_INFORMATION OperationInformation)

{

DbgPrint("[lyshark] 执行回调函数... \n"); return STATUS\_SUCCESS;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

ObUnRegisterCallbacks(Globle\_Object\_Handle);

DbgPrint("回调卸载完成... \n");

}

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

{

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

BypassCheckSign(Driver);

OB\_OPERATION\_REGISTRATION Base; // 回调函数结构体

OB\_CALLBACK\_REGISTRATION CallbackReg; // 回调函数

CallbackReg.RegistrationContext = NULL; // 注册上下文(你回调函数返回参数)

CallbackReg.Version = OB\_FLT\_REGISTRATION\_VERSION; // 注册回调版本

CallbackReg.OperationRegistration = &Base; // 回调结构体

CallbackReg.OperationRegistrationCount = 1; // 操作计数(下钩数量)

RtlUnicodeStringInit(&CallbackReg.Altitude, L"600000"); // 长度

Base.ObjectType = PsProcessType; // 进程操作类型.此处为进程操作

Base.Operations = OB\_OPERATION\_HANDLE\_CREATE; // 操作句柄创建Base.PreOperation = MyLySharkComObjectCallBack; // 你自己的回调函数Base.PostOperation = NULL;

// 注册回调

if (ObRegisterCallbacks(&CallbackReg, &Globle\_Object\_Handle))

{

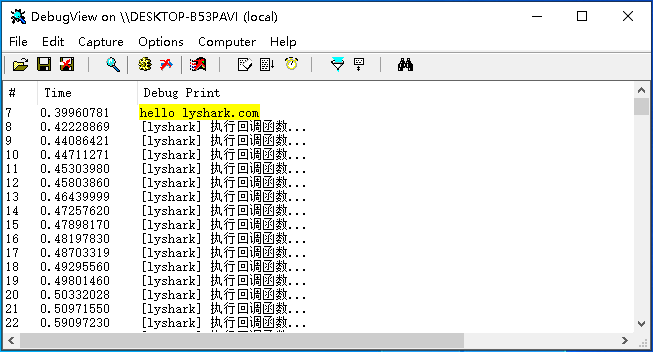
DbgPrint("[lyshark message] 回调注册成功...");

}

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

}

当驱动程序被加载以后，一旦有进程运行则会执行我们自己的 MyLySharkComObjectCallBack 回调，而在回调函数内则可以执行任意功能，运行如下所示。



如上所示只是演示基本的回调申请流程，回调函数通常需要包含两个值，其一 RegistrationContext 用于标注上下文，其二 POB\_PRE\_OPERATION\_INFORMATION 则用于标注进程或者线程创建的信息结构体。

OB\_PREOP\_CALLBACK\_STATUS MyLySharkComObjectCallBack(PVOID RegistrationContext, POB\_PRE\_OPERATION\_INFORMATION OperationInformation)

# 那么如何实现 拦截进程启动 这个功能呢，我们可以在回调函数中写入以下代码进行拦截。

CreateHandleInformation.DesiredAccess 将打开句柄的权限清零

CreateHandleInformation.OriginalDesiredAccess 判断是否终止

if (pOperationInformation->Operation == OB\_OPERATION\_HANDLE\_CREATE)

{

DbgPrint("lyshark.exe 进程打开 \n");

pOperationInformation->Parameters->CreateHandleInformation.DesiredAccess=0; if ((pOperationInformation->Parameters-

>CreateHandleInformation.OriginalDesiredAccess & PROCESS\_TERMINATE) == PROCESS\_TERMINATE)

{

pOperationInformation->Parameters->CreateHandleInformation.DesiredAccess &= ~PROCESS\_TERMINATE;

}

}

拦截进程创建核心代码如下所示。

// 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 <ntddk.h> #include <ntstrsafe.h>

#define PROCESS\_TERMINATE 0x1

// 导出两个API



NTKERNELAPI PEPROCESS IoThreadToProcess(PETHREAD Thread); NTKERNELAPI char PsGetProcessImageFileName(PEPROCESS Process);

// 全局句柄

PVOID Globle\_Object\_Handle = NULL;

// 绕过签名检测

void BypassCheckSign(PDRIVER\_OBJECT pDriverObj)

{

typedef struct \_LDR\_DATA

{

struct \_LIST\_ENTRY InLoadOrderLinks; struct \_LIST\_ENTRY InMemoryOrderLinks;

struct \_LIST\_ENTRY InInitializationOrderLinks; VOID DllBase;

VOID EntryPoint;

ULONG32 SizeOfImage;

UINT8 \_PADDING0\_[0x4];

struct \_UNICODE\_STRING FullDllName; struct \_UNICODE\_STRING BaseDllName; ULONG32 Flags;

}LDR\_DATA, PLDR\_DATA;

PLDR\_DATA ldr;

ldr = (PLDR\_DATA)(pDriverObj->DriverSection); ldr->Flags |= 0x20;

}

// 判断是否是需要保护的进程

BOOLEAN CheckProcess(PEPROCESS eprocess)

{

char Name = PsGetProcessImageFileName(eprocess); if (!\_stricmp("lyshark.exe", Name))

return TRUE;

else

return FALSE;

}

// 进程回调

OB\_PREOP\_CALLBACK\_STATUS MyLySharkProcessObjectCallBack(PVOID RegistrationContext, POB\_PRE\_OPERATION\_INFORMATION pOperationInformation)

{

HANDLE pid;

// 只取出进程回调

if (pOperationInformation->ObjectType != PsProcessType)

{

return OB\_PREOP\_SUCCESS;

}

// 得到所有进程的ID

pid = PsGetProcessId((PEPROCESS)pOperationInformation->Object);

// DbgPrint("进程PID= %ld \n", pid);

UNREFERENCED\_PARAMETER(RegistrationContext);

// 验证是否是需要的进程

if (CheckProcess((PEPROCESS)pOperationInformation->Object))

{

// 创建句柄

if (pOperationInformation->Operation == OB\_OPERATION\_HANDLE\_CREATE)

{

DbgPrint("lyshark.exe 进程打开事件 \n"); pOperationInformation->Parameters-

>CreateHandleInformation.DesiredAccess=0;

if ((pOperationInformation->Parameters-

>CreateHandleInformation.OriginalDesiredAccess & PROCESS\_TERMINATE) == PROCESS\_TERMINATE)

{

DbgPrint("[LyShark Message] 拦截进程打开 \n"); pOperationInformation->Parameters-

>CreateHandleInformation.DesiredAccess &= ~PROCESS\_TERMINATE;

}

}

// 复制句柄

if (pOperationInformation->Operation == OB\_OPERATION\_HANDLE\_DUPLICATE)

{

DbgPrint("lyshark.exe 进程被关闭 \n"); pOperationInformation->Parameters-

>DuplicateHandleInformation.DesiredAccess=0;

if ((pOperationInformation->Parameters-

>DuplicateHandleInformation.OriginalDesiredAccess & PROCESS\_TERMINATE) == PROCESS\_TERMINATE)

{

pOperationInformation->Parameters-

>DuplicateHandleInformation.DesiredAccess &= ~PROCESS\_TERMINATE;

}

}

}

return OB\_PREOP\_SUCCESS;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

ObUnRegisterCallbacks(Globle\_Object\_Handle);

DbgPrint("回调卸载完成... \n");

}

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

{

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

BypassCheckSign(Driver);

OB\_OPERATION\_REGISTRATION ob\_process\_callback; OB\_CALLBACK\_REGISTRATION op\_process\_operation;

memset(&ob\_process\_callback, 0, sizeof(ob\_process\_callback)); ob\_process\_callback.ObjectType = PsProcessType;

ob\_process\_callback.Operations = OB\_OPERATION\_HANDLE\_CREATE | OB\_OPERATION\_HANDLE\_DUPLICATE;

ob\_process\_callback.PreOperation = MyLySharkProcessObjectCallBack;

ob\_process\_callback.PostOperation = NULL;

RtlUnicodeStringInit(&op\_process\_operation.Altitude, L"600000"); op\_process\_operation.RegistrationContext = NULL; op\_process\_operation.Version = OB\_FLT\_REGISTRATION\_VERSION; op\_process\_operation.OperationRegistration = &ob\_process\_callback; op\_process\_operation.OperationRegistrationCount = 1;

// 注册进程回调

if (ObRegisterCallbacks(&op\_process\_operation, &Globle\_Object\_Handle))

{

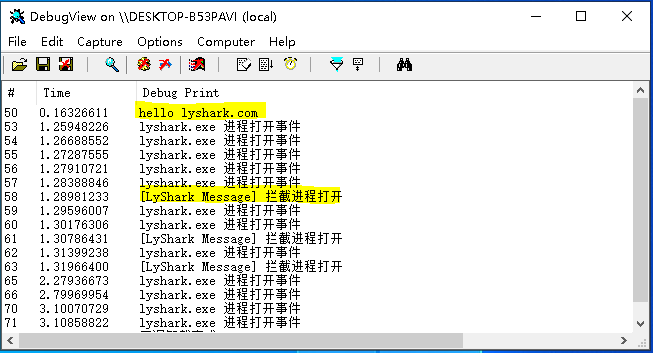
DbgPrint("进程回调注册成功...");

}

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

}

加载这个驱动，当有进程被创建时，则首先判断是否是 lyshark.exe 如果是则直接禁止打开，也就是终止掉。



# 同理进程可以被拦截，那么如果增加更多的过滤条件，则 线程 同样可以被拦截，拦截线程代码如下所示。

// 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 <ntddk.h> #include <ntstrsafe.h>

#define THREAD\_TERMINATE2 0x1

// 导出两个API

NTKERNELAPI PEPROCESS IoThreadToProcess(PETHREAD Thread); NTKERNELAPI char PsGetProcessImageFileName(PEPROCESS Process);



// 全局句柄

PVOID Globle\_Object\_Handle = NULL;

// 绕过签名检测

void BypassCheckSign(PDRIVER\_OBJECT pDriverObj)

{

typedef struct \_LDR\_DATA

{

struct \_LIST\_ENTRY InLoadOrderLinks; struct \_LIST\_ENTRY InMemoryOrderLinks;

struct \_LIST\_ENTRY InInitializationOrderLinks; VOID DllBase;

VOID EntryPoint;

ULONG32 SizeOfImage;

UINT8 \_PADDING0\_[0x4];

struct \_UNICODE\_STRING FullDllName; struct \_UNICODE\_STRING BaseDllName; ULONG32 Flags;

}LDR\_DATA, PLDR\_DATA;

PLDR\_DATA ldr;

ldr = (PLDR\_DATA)(pDriverObj->DriverSection); ldr->Flags |= 0x20;

}

// 判断是否是需要保护的进程

BOOLEAN CheckProcess(PEPROCESS eprocess)

{

char Name = PsGetProcessImageFileName(eprocess); if (!\_stricmp("lyshark.exe", Name))

return TRUE; else

return FALSE;

}

// 线程回调

OB\_PREOP\_CALLBACK\_STATUS MyThreadObjectCallBack(PVOID RegistrationContext, POB\_PRE\_OPERATION\_INFORMATION pOperationInformation)

{

PEPROCESS ep; PETHREAD et; HANDLE pid;

// 线程过滤

if (pOperationInformation->ObjectType != PsThreadType)

{

return OB\_PREOP\_SUCCESS;

}

et = (PETHREAD)pOperationInformation->Object; ep = IoThreadToProcess(et);

pid = PsGetProcessId(ep);

// DbgPrint("线程PID= %ld | TID= %ld \n", pid, PsGetThreadId(et)); UNREFERENCED\_PARAMETER(RegistrationContext);

if (CheckProcess(ep))

{

if (pOperationInformation->Operation == OB\_OPERATION\_HANDLE\_CREATE)

{

pOperationInformation->Parameters-

>CreateHandleInformation.DesiredAccess=0;

if ((pOperationInformation->Parameters-

>CreateHandleInformation.OriginalDesiredAccess & THREAD\_TERMINATE2) == THREAD\_TERMINATE2)

{

DbgPrint("[LyShark] 拦截lyshark.exe进程内 %d 线程创建 \n", PsGetThreadId(et));

pOperationInformation->Parameters-

>CreateHandleInformation.DesiredAccess &= ~THREAD\_TERMINATE2;

}

}

if (pOperationInformation->Operation == OB\_OPERATION\_HANDLE\_DUPLICATE)

{

pOperationInformation->Parameters-

>DuplicateHandleInformation.DesiredAccess=0;

if ((pOperationInformation->Parameters-

>DuplicateHandleInformation.OriginalDesiredAccess & THREAD\_TERMINATE2) == THREAD\_TERMINATE2)

{

pOperationInformation->Parameters-

>DuplicateHandleInformation.DesiredAccess &= ~THREAD\_TERMINATE2;

}

}

}

return OB\_PREOP\_SUCCESS;

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

ObUnRegisterCallbacks(Globle\_Object\_Handle);

DbgPrint("回调卸载完成... \n");

}

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

{

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

BypassCheckSign(Driver);

OB\_OPERATION\_REGISTRATION ob\_thread\_callback; OB\_CALLBACK\_REGISTRATION op\_thread\_operation;

memset(&ob\_thread\_callback, 0, sizeof(ob\_thread\_callback)); ob\_thread\_callback.ObjectType = PsThreadType; ob\_thread\_callback.Operations = OB\_OPERATION\_HANDLE\_CREATE |

OB\_OPERATION\_HANDLE\_DUPLICATE;

ob\_thread\_callback.PreOperation = MyThreadObjectCallBack; ob\_thread\_callback.PostOperation = NULL;

RtlUnicodeStringInit(&op\_thread\_operation.Altitude, L"600001"); op\_thread\_operation.RegistrationContext = NULL; op\_thread\_operation.Version = OB\_FLT\_REGISTRATION\_VERSION; op\_thread\_operation.OperationRegistration = &ob\_thread\_callback; op\_thread\_operation.OperationRegistrationCount = 1;

// 注册进程回调

if (ObRegisterCallbacks(&op\_thread\_operation, &Globle\_Object\_Handle))

{

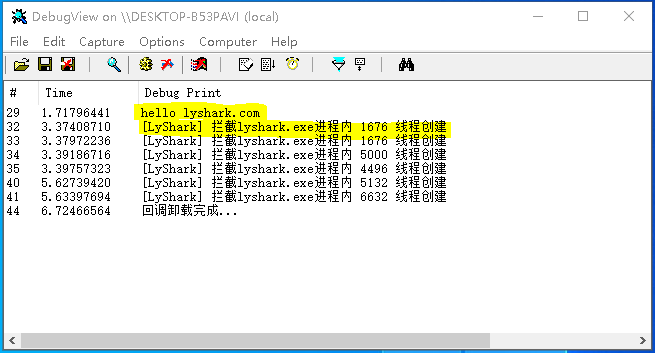
DbgPrint("进程回调注册成功...");

}

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

}

# 这段驱动加载后，如果有新线程被创建，则会被拦截并打印输出，效果图如下。



**参考文献**

# h[ttps://www.cnblogs.com/ciyze0101/p/5468175.htm](http://www.cnblogs.com/ciyze0101/p/5468175.html)l

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！