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。

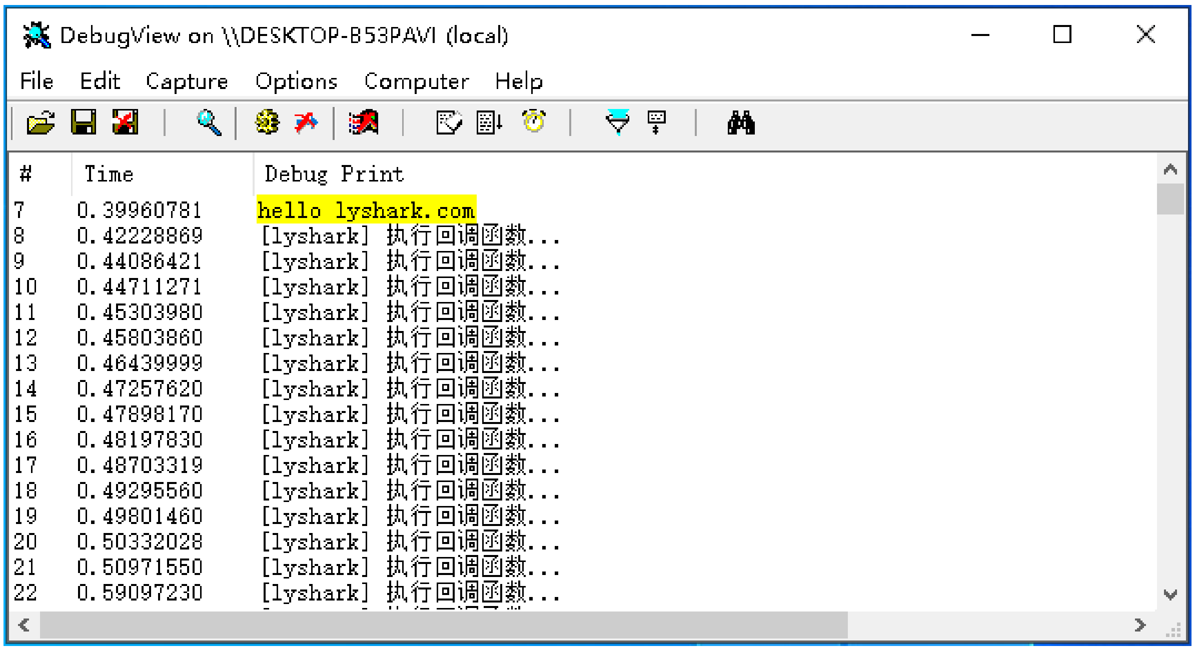
函数`ObRegisterCallbacks`其微软对他的定义是这样的，用户传入`OB_OPERATION_REGISTRATION`结构，以及`OB_CALLBACK_REGISTRATION`回调结构，其中`PreOperation`则是传入的回调函数，也是最重要的，其次是`ObjectType`指定成进程回调。

```
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
#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。


DebugView on \\DESKTOP-B53PAVI (local)
File Edit Capture Options Computer Help

```
#   Time        Debug Print
7   0.39960781  hello lyshark.com
8   0.42228869  [lyshark] 执行回调函数...
9   0.44086421  [lyshark] 执行回调函数...
10  0.44711271  [lyshark] 执行回调函数...
11  0.45303980  [lyshark] 执行回调函数...
12  0.45803860  [lyshark] 执行回调函数...
13  0.46439999  [lyshark] 执行回调函数...
14  0.47257620  [lyshark] 执行回调函数...
15  0.47898170  [lyshark] 执行回调函数...
16  0.48197830  [lyshark] 执行回调函数...
17  0.48703319  [lyshark] 执行回调函数...
18  0.49295560  [lyshark] 执行回调函数...
19  0.49801460  [lyshark] 执行回调函数...
20  0.50332028  [lyshark] 执行回调函数...
21  0.50971550  [lyshark] 执行回调函数...
22  0.59097230  [lyshark] 执行回调函数...
```

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

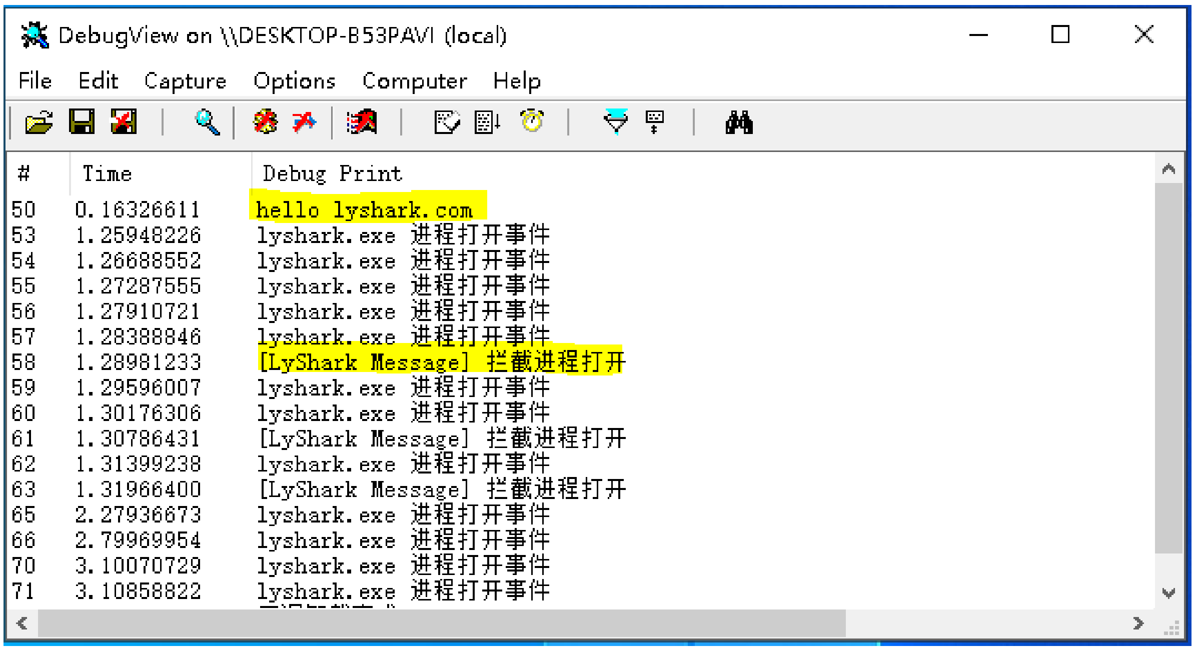
```
if (pOperationInformation->Operation == OB_OPERATION_HANDLE_CREATE)
{
    DbgPrint("lyshark.exe 进程打开 \n");
    pOperationInformation->Parameters->CreateHandleInformation.DesiredAccess=0;
    if ((pOperationInformation->Parameters-
>CreateHandleInformation.OriginalDesiredAccess & PROCESS_TERMINATE) ==
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
#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。


DebugView on \\DESKTOP-B53PAVI (local)

```
#    Time          Debug Print
50   0.16326611    hello lyshark.com
53   1.25948226    lyshark.exe 进程打开事件
54   1.26688552    lyshark.exe 进程打开事件
55   1.27287555    lyshark.exe 进程打开事件
56   1.27910721    lyshark.exe 进程打开事件
57   1.28388846    lyshark.exe 进程打开事件
58   1.28981233    [LyShark Message] 拦截进程打开
59   1.29596007    lyshark.exe 进程打开事件
60   1.30176306    lyshark.exe 进程打开事件
61   1.30786431    [LyShark Message] 拦截进程打开
62   1.31399238    lyshark.exe 进程打开事件
63   1.31966400    [LyShark Message] 拦截进程打开
65   2.27936673    lyshark.exe 进程打开事件
66   2.79969954    lyshark.exe 进程打开事件
70   3.10070729    lyshark.exe 进程打开事件
71   3.10858822    lyshark.exe 进程打开事件
```

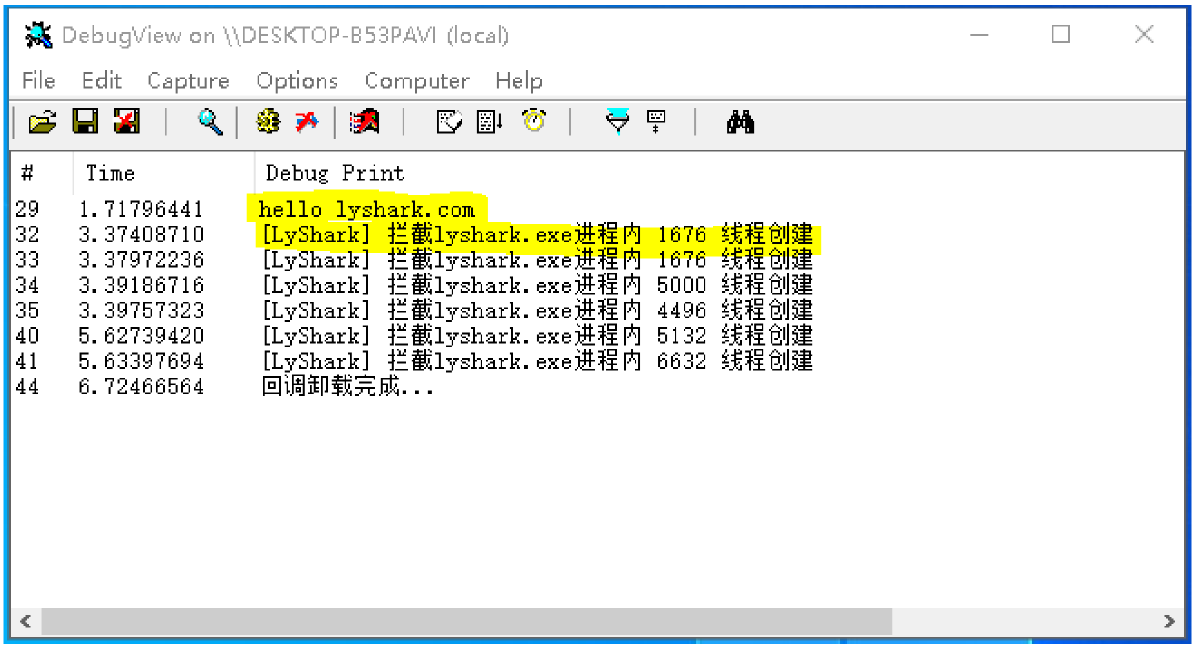
同理进程可以被拦截，那么如果增加更多的过滤条件，则`线程`同样可以被拦截，拦截线程代码如下所示。

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


DebugView on \\DESKTOP-B53PAVI (local)
File Edit Capture Options Computer Help

```
#   Time         Debug Print
29  1.71796441   hello lyshark.com
32  3.37408710   [LyShark] 拦截lyshark.exe进程内 1676 线程创建
33  3.37972236   [LyShark] 拦截lyshark.exe进程内 1676 线程创建
34  3.39186716   [LyShark] 拦截lyshark.exe进程内 5000 线程创建
35  3.39757323   [LyShark] 拦截lyshark.exe进程内 4496 线程创建
40  5.62739420   [LyShark] 拦截lyshark.exe进程内 5132 线程创建
41  5.63397694   [LyShark] 拦截lyshark.exe进程内 6632 线程创建
44  6.72466564   回调卸载完成...
```

参考文献

https://www.cnblogs.com/ciyze0101/p/5468175.html

作者： 王瑞 (LyShark)

作者邮箱： me@lyshark.com