# 监控进程的启动与退出可以使用 来创建回调，当新进程产生

PsSetCreateProcessNotifyRoutineEx



时，回调函数会被率先执行，然后执行我们自己的 MyCreateProcessNotifyEx 函数，并在内部进行打印输出。

#include <ntddk.h>

NTKERNELAPI PCHAR PsGetProcessImageFileName(PEPROCESS Process);

NTKERNELAPI NTSTATUS PsLookupProcessByProcessId(HANDLE ProcessId, PEPROCESS Process);

PCHAR GetProcessNameByProcessId(HANDLE ProcessId)

{

NTSTATUS st = STATUS\_UNSUCCESSFUL;

PEPROCESS ProcessObj = NULL; PCHAR string = NULL;

st = PsLookupProcessByProcessId(ProcessId, &ProcessObj); if (NT\_SUCCESS(st))

{

string = PsGetProcessImageFileName(ProcessObj); ObfDereferenceObject(ProcessObj);

}

return string;

}

VOID MyCreateProcessNotifyEx(PEPROCESS Process, HANDLE ProcessId, PPS\_CREATE\_NOTIFY\_INFO CreateInfo)

{

char ProcName[16] = { 0 }; if (CreateInfo != NULL)

{

strcpy(ProcName, PsGetProcessImageFileName(Process));

DbgPrint("父进程ID: %ld --->父进程名: %s --->进程名: %s---->进程路径：%wZ", CreateInfo->ParentProcessId,

GetProcessNameByProcessId(CreateInfo->ParentProcessId), PsGetProcessImageFileName(Process),CreateInfo->ImageFileName);

}

else

{

strcpy(ProcName, PsGetProcessImageFileName(Process));

DbgPrint("进程[ %s ] 离开了，程序被关闭了",ProcName);

}

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

PsSetCreateProcessNotifyRoutineEx((PCREATE\_PROCESS\_NOTIFY\_ROUTINE\_EX)MyCreatePro cessNotifyEx, TRUE);

}

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

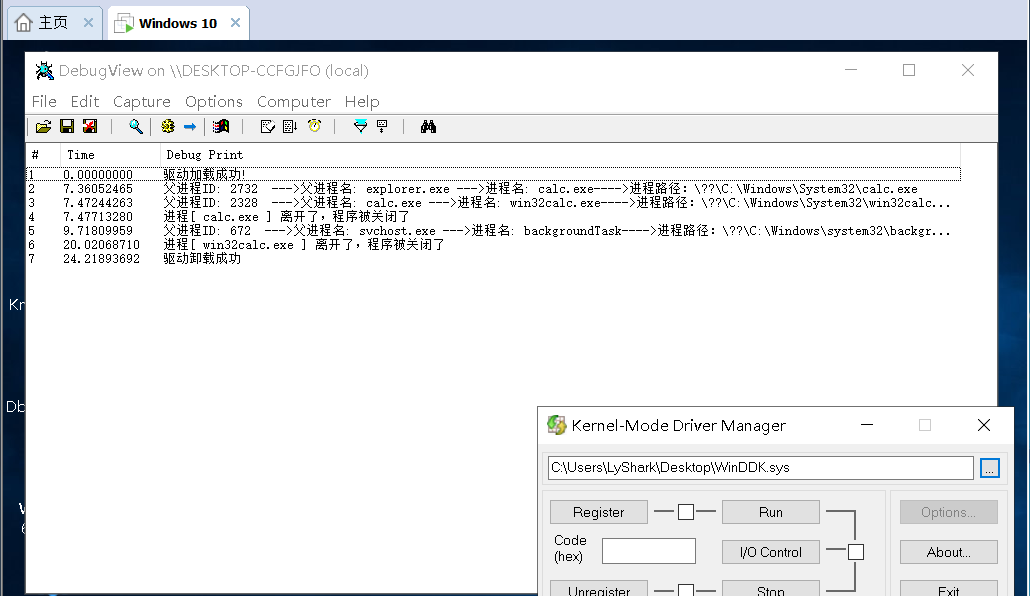
{

NTSTATUS status;

status = PsSetCreateProcessNotifyRoutineEx((PCREATE\_PROCESS\_NOTIFY\_ROUTINE\_EX)MyCreatePro cessNotifyEx, FALSE);

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

}



在上方代码基础上进行一定的改进，思路：通过 PsGetProcessImageFileName 即将PID转换为进程

名，然后通过 \_stricmp 对比，如果发现是 calc.exe 进程则拒绝执行，禁止特定服务的运行，实现代码如下：



#include <ntddk.h>

NTKERNELAPI PCHAR PsGetProcessImageFileName(PEPROCESS Process);

NTKERNELAPI NTSTATUS PsLookupProcessByProcessId(HANDLE ProcessId, PEPROCESS Process);

PCHAR GetProcessNameByProcessId(HANDLE ProcessId)

{

NTSTATUS st = STATUS\_UNSUCCESSFUL;

PEPROCESS ProcessObj = NULL; PCHAR string = NULL;

st = PsLookupProcessByProcessId(ProcessId, &ProcessObj);

if (NT\_SUCCESS(st))

{

string = PsGetProcessImageFileName(ProcessObj); ObfDereferenceObject(ProcessObj);

}

return string;

}

VOID MyCreateProcessNotifyEx(PEPROCESS Process, HANDLE ProcessId, PPS\_CREATE\_NOTIFY\_INFO CreateInfo)

{

char ProcName[16] = { 0 }; if (CreateInfo != NULL)

{

strcpy(ProcName, PsGetProcessImageFileName(Process)); if (!\_stricmp(ProcName, "calc.exe"))

{

CreateInfo->CreationStatus = STATUS\_UNSUCCESSFUL;

}

}

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

PsSetCreateProcessNotifyRoutineEx((PCREATE\_PROCESS\_NOTIFY\_ROUTINE\_EX)MyCreatePro cessNotifyEx, TRUE);

DbgPrint(("驱动卸载成功"));

}

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

{

NTSTATUS status; status =

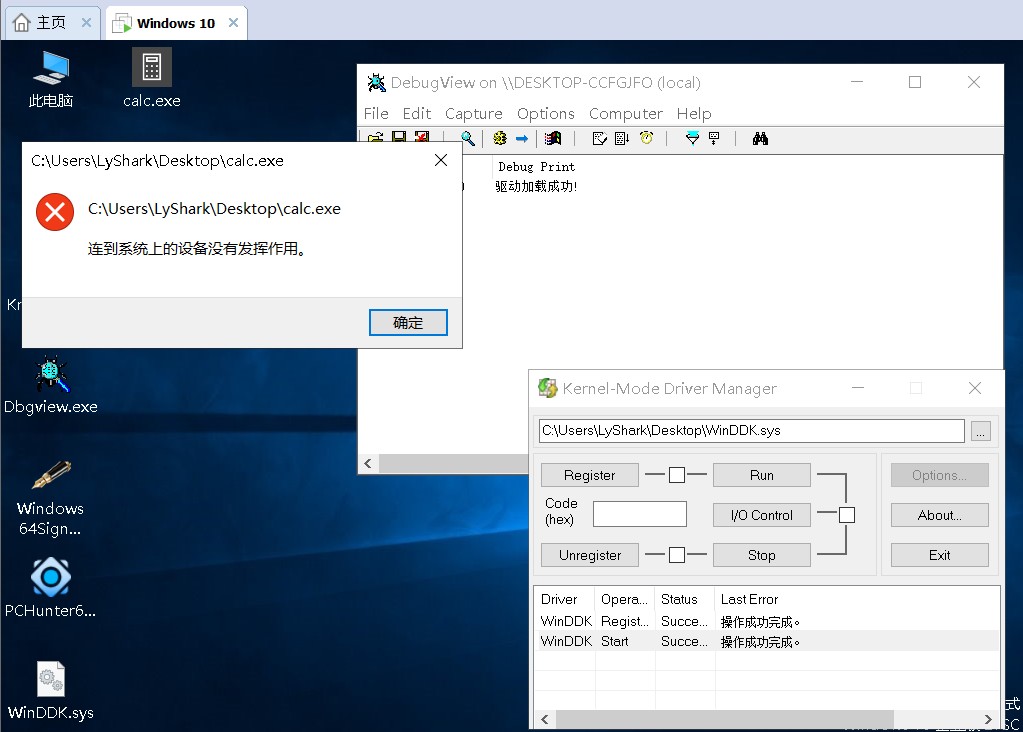
PsSetCreateProcessNotifyRoutineEx((PCREATE\_PROCESS\_NOTIFY\_ROUTINE\_EX)MyCreatePro cessNotifyEx, FALSE);

Driver->DriverUnload = UnDriver;

DbgPrint("驱动加载成功!"); return STATUS\_SUCCESS;

}

将上方代码编译，当我们加载驱动程序以后，再次打开 C:\Windows\System32\calc.exe 计算器进程则提示无法打开，我们的驱动已经成功的拦截了本次的请求。



而检测线程操作与检测进程差不多，检测线程需要调用 PsSetCreateThreadNotifyRoutine 创建回调函数，然后就可以检测线程的创建了，具体代码如下：

#include <ntddk.h>



NTKERNELAPI PCHAR PsGetProcessImageFileName(PEPROCESS Process);

NTKERNELAPI NTSTATUS PsLookupProcessByProcessId(HANDLE ProcessId, PEPROCESS Process);

VOID MyCreateThreadNotify(HANDLE ProcessId, HANDLE ThreadId, BOOLEAN Create)

{

PEPROCESS eprocess = NULL;

PsLookupProcessByProcessId(ProcessId, &eprocess); // 通过此函数拿到程序的EPROCESS结构

if (Create)

DbgPrint("线程TID: %1d --> 所属进程名: %s --> 进程PID: %1d \n", ThreadId, PsGetProcessImageFileName(eprocess), PsGetProcessId(eprocess));

else

DbgPrint("%s 线程已退出...", ThreadId);

}

VOID UnDriver(PDRIVER\_OBJECT driver)

{

PsRemoveCreateThreadNotifyRoutine(MyCreateThreadNotify); DbgPrint(("驱动卸载成功"));

}

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

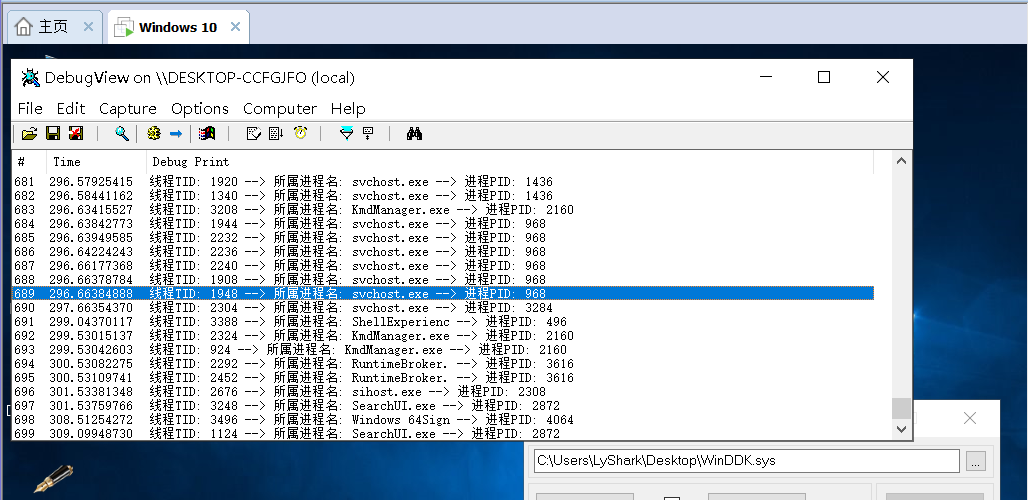
{

NTSTATUS status;

status = PsSetCreateThreadNotifyRoutine(MyCreateThreadNotify); DbgPrint("PsSetCreateThreadNotifyRoutine: %x", status); Driver->DriverUnload = UnDriver;

return STATUS\_SUCCESS;

}



# 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