x64内核：攻防相关

1. 沙盒（监控行为 hook ssdt inline hook 系统回调)
2. 主动防御(hook ssdt inline hook 系统回调)
3. 防火墙过滤驱动（TDI、NDIS、WFP)
4. 文件过滤驱动(sfilter、minifilter)

隐藏进程

- 方法：EPROCESS脱链
- 检测：全局有张句柄表（包含进程、线程等信息）

```
void HideProcess(ULONG pid)
{
    PEPROCESS process = NULL;
    NTSTATUS status;
    PLIST_ENTRY listEntry;

    status = PsLookupProcessByProcessId((HANDLE)pid, &process);
    if (NT_SUCCESS(status)) {
        //断链表
        listEntry = (PLIST_ENTRY)((char*)process + 0x188);
        listEntry->Flink->Blink = listEntry->Blink;
        listEntry->Blink->Flink = listEntry->Flink;

        if (process)
            ObDereferenceObject(process);
    }
}
```

拒绝（或强制）结束进程

- 拒绝结束进程：注册系统回调，访问权限清掉
- 强制结束进程：调用更底层的API（因为结束进程时先遍历回调数组并调用，之后调用更底层的API)

对于未导出的底层API，通过特征码寻找

```
typedef NTSTATUS
(*PsTerminateProcess)(
    PEPROCESS Process,
    NTSTATUS ExitStatus
);

void MyTerminateProcess(ULONG pid)
{
```

```
    PEPROCESS process = NULL;
    NTSTATUS status;
    // 此函数地址需要通过特征码来寻找
    PsTerminateProcess pfnPsTerminateProcess =
(PsTerminateProcess)0xfffff80004284120;

    status = PsLookupProcessByProcessId((HANDLE)pid, &process);
    if (NT_SUCCESS(status)) {
        status = pfnPsTerminateProcess(process, 0);

        if (process)
            ObDereferenceObject(process);
    }
}
```

操作其他进程的句柄

进程中的句柄存储在进程的句柄表（EPROCESS + 0x200处）中

例如强制删除正在运行的文件：

1. `ZwQuerySystemInformation(SystemHandleInformation);`遍历系统信息，拿到句柄信息
2. 在句柄信息中存在引用对象，使用对象前利用`ObReferenceObject`增加对象引用计数
3. `ObQueryNameString`查询对象的名称，对比是否为目标
4. `KeStackAttachProcess`附加到目标进程中
5. `ZwClose`关闭句柄
6. `KeUnstackDetachProcess`解除附加
7. Delete文件