说完基本环境配置，这篇就稍微轻松点了，可以听听我吹牛侃大山，谈谈 WIN64 内核编程的基本规则。在 WIN32 环境下，大家都能乱来，随便加载各种驱动，进行各种挂钩和 DKOM，各种穷凶极恶的手段粉墨登场。把好端端的 WINDOWS 弄得连七八糟，严重影响了系统安全。虽说为编程爱好者提供了表演的舞台，但是苦了那些把电脑当工具的人。于是那段时间，“LINUX 和 MAC OS 比 WINDOWS 安全可靠”的谣言四起（似乎也不是谣言），大有把 WINDOWS 和不安全划上等号之势。

为此，微软很生气，后果很严重。从 WINDOWS 2003 X64 开始，微软开始对 WIN64 系统增加限制，增强系统安全。总体来说有两条，一是 KPP（Kernel Patch Protection，内核补丁保护），二是 DSE（Driver Signature Enforcement，驱动签名强制）。WINDOWS 2003 X64 只有 KPP，从 VISTA 开始有了 DSE。KPP 利用 PatchGuard 技术检查内核有没有被“打补丁”（不仅检查内核函数有没有被 HOOK，也包括一些关键的内核结构体有没有被修改，比如进程链表 PsActiveProcessLinks 有没有被摘链），如果发现被“打补丁”，则直接引发 0x109 蓝屏（CRITICAL_STRUCTURE_CORRUPTION，直译为关键结构损毁）。DSE 则是拒绝加载不包含正确签名的驱动（包括伪造签名和测试签名）。多说一句，总有人把 KPP 把 PatchGuard 划等号，其实二者是不等的。KPP 是机制，PatchGuard 是实现。就好比 CIA 是机构，CIA 的特工才是一系列“黑色行动”的执行者。

说完正儿八经的，说点通地气的话。实际上 KPP 和 DSE 并非铁板一块，二是各有漏洞可钻的。KPP 保护不了内核所有的部分，只保护了几个驱动：NTOSKRNL.EXE、HAL.DLL、CI.DLL、NDIS.SYS 等（当然，NTOS 部分包括了 IDT、GDT、MSR 等）。对一些较为上层的驱动，比如 FAT32 和 NTFS 作 IRP HOOK，PG 是不管的。而 DSE 则在某些条件下不启动，比如在 PE 环境下；或者说有些时候管得不严格，比如在测试模式下，允许含有测试签名的驱动加载。总结一句：**进行 WIN64 内核编程的时候，别想用 API HOOK 解决问题；当发布含有 WIN64 驱动的时候，记得给“证书签发机构”交保护费（购买正规数字签名）**。不过，这两项限制让很多黑客乃至安全公司大为不满，各种过 KPP 和 DSE 的方法层出不穷。目前，VISTA、WIN7、WIN8、WIN8.1 的 KPP 和 DSE 已全部被攻破。

编程的时候，大家基本都是需要使用 API 的。在 RING3 下使用 WINAPI，在 RING0 下则使用内核 API。**特别注意的是，内核编程是无法使用 WINAPI 的（当然，也有特殊的方法调用，不过非标准方法，这里略过不提）**。什么叫做内核 API 呢，就是虚拟地址位于内核空间的 API。**不管是不是微软的内核模块，也不管导出没导出，只要知道地址和每个参数的含义，就能调用**。不过，我们写程序大多时候都是使用微软模块（NTOSKRNL、HAL 等）导出的 API。例如：ZwOpenProcess，IoCreateFile 等。

内核编程用内核 API，而自然也有内核结构体。其实“内核结构体”这个说法不太妥当，因为结构体是不分场合甚至不分系统的。但这么说大家也能理解是什么意思，就是内核编程中常用的结构体。这种结构体又分为两种，一种是“万年不变”的，一种是每个系统都不同的。“万年不变”的结构体通常能在 MSDN 上查到，比如 CLIENT_ID、IO_STATUS_BLOCK；每个系统都不同的结构体通常在 MSDN 上查不到，但是存在于符号文件里，比如 EPROCESS、ETHREAD。**我们编程的时候，尽量只使用“万年不变”的结构体，不使用每个系统都不同的结构体。当非要使用不可的时候，必须根据系统版本定义制定成员的偏移量**。如果发现未知的系统版本，则提示并退出。如果不这样做，等着 BSOD 吧。

内核编程的基本规矩不是一篇能讲完的，下面几篇会细化讲解，这篇只是个引子。