**基于SSDT HOOK的window行为监控**

1. **实验环境**

虚拟机软件：VMWare

操作系统：Microsoft Windows XP Professional SP3

开发环境：VC++ 6.0

WDK版本：WDK 7.1

1. **原理简介**

SSDT 的全称是 System Services Descriptor Table，系统服务描述符表。这个表就是把 Ring3 的 Win32 API 和 Ring0 的内核 API 联系了起来。

SSDT 并不仅仅只包含一个庞大的地址索引表，它还包含着一些其它有用的信息，诸如地址索引的基地址、服务函数个数等。通过修改此表的函数地址可以对常用 Windows 函数及 API 进行 Hook，从而实现对一些关心的系统动作进行过滤、监控的目的。一些 HIPS、防毒软件、系统监控、注册表监控软件往往会采用此接口来实现自己的监控模块。

我就对SSDT HOOK进行了一些初步的探索，但受限于对windows内核了解的欠缺只实现了一些简单的功能。

1. **项目简介**

在本次课程实践中，我实现了一个最基础的SSDT HOOK的框架，并且挂钩了

NtCreateFile

NtCreateKey

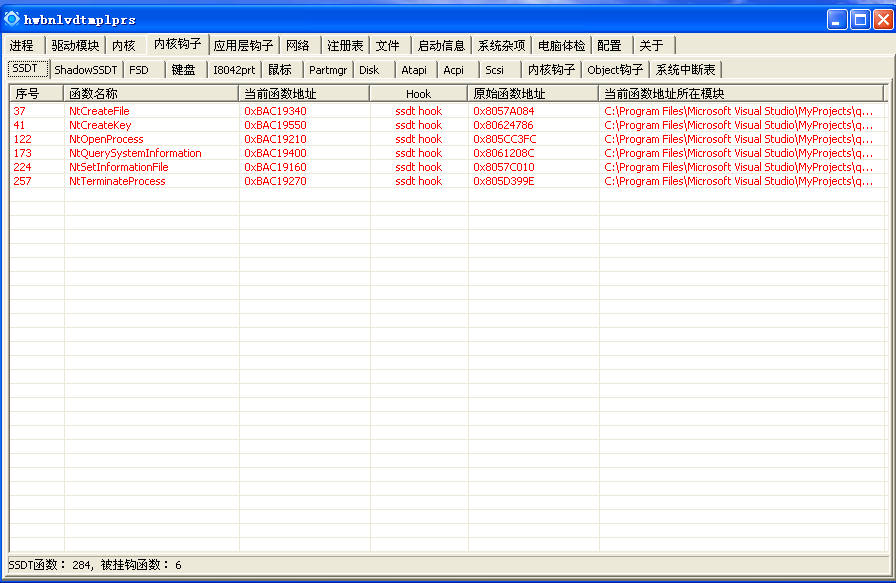
NtOpenProcess

NtQuerySystemInformation

NtSetInformationFile

NtTerminateProcess

这6个函数，实现了对这6个函数的新的挂钩函数，并在这些函数中做了一些测试性的探索，实现了一些简单的监测保护的功能。



在pchunter中显示的被挂钩的ssdt函数

1. **使用说明**

Ssdt文件夹中是源代码，ssdt.sys是构建的驱动程序，首先使用instdrv装载ssdt.sys驱动，然后打开debugview即可看见输出打印的信息。

测试环境：windows xp sp3

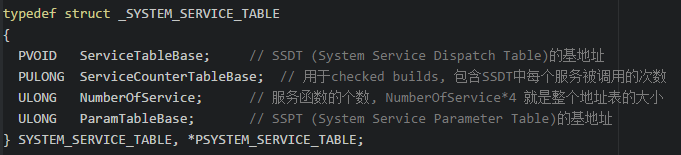
1. **环境配置**

我使用的开发环境是vc6++ 和 wdk7.1，并且用debugview来查看调试信息，使用instdrv来加载驱动。在配置开发环境时，由于对于驱动开发的不熟悉，在网上找了很久资料，最后经过多次尝试终于成功了。参考网址：

<http://blog.chinaunix.net/uid-9847882-id-3076919.html>

关键是要设置好VC编译驱动的环境，这里我照着上面网址中的步骤做，终于成功了。但是并不是很懂具体的实现原理。

1. **SSDT HOOK框架的实现**

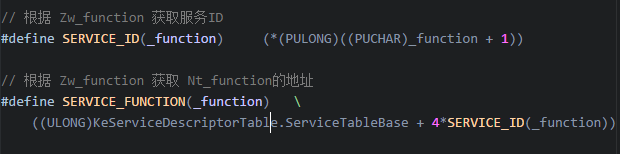
SSDT HOOK的原理部分在课件 02-windows木马后门监测中已经有了比较详细的解释，在这里只做简单的介绍。首先KeServiceDescriptorTable是windows内核导出的SSDT表。结构如下

然后适用这句话就能导出这个表



Step1：要实现ssdt hook 首先需要禁用windows内存保护机制。上课时介绍了三种方法。在这里，为了简便起见，我使用的第二种，就是修改控制寄存器cr0，将wp位设为0.

Step2：根据Zw\*函数找到Nt\*函数地址。提供了一下两个宏。



Step3：实现通用的HOOK和UNHOOK函数，以及初始化servicetable的函数，提供HOOK和UNHOOK以及驱动初始化服务。

Step4：实现自己的MyNt\*挂钩函数，使用HOOK和UNHOOK注册。

具体细节可以看源代码。

1. **具体挂钩函数的实现**

在实现具体的挂钩函数的时候，我遇到了这么几个**问题**：

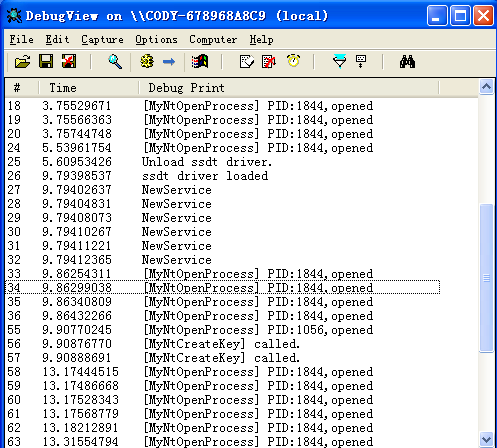
如何找到自己需要的需要挂钩的内核函数？

有些内核函数在ntddk.h中并没有声明，如何使用？

通过查找资料，对于那些有但是没有在ntddk.h中声明的函数，只需要自己手动声明即可。而关于内核函数的意义的一些解释以及数据结构的解释我找到了一本电子书

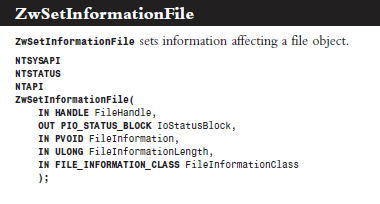
Windows NT 2000 Native API Reference.pdf，在这中有对所有zw\*函数的详细解释，给了我很多的参考。

1. 对于：**NtCreateFile，NtCreateKey，NtOpenProcess**这三个函数，我只是简单了用自己的挂钩函数进行的简单的监测在debugview中打印出相关的信息，比如哪个进程调用了NtOpenProcess这个函数等。



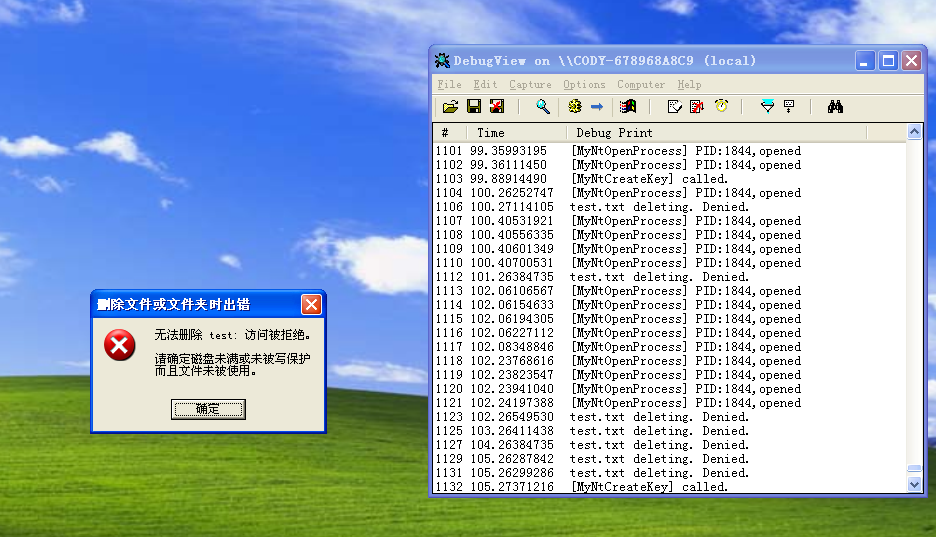
装载ssdt.sys驱动之后，debugview中的输出。监测NtOpenProcess,NtCreateKey的调用并在debugview中输出信息。

1. **NtSetInformationFile**。通过这个函数，我实现了禁止用户删除名字为（test.txt）的文件/文件夹。

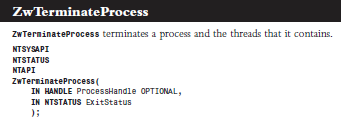


数据结构

我们通过FileHandler获取对应的FileObject对象，然后判断文件名是否等于test.txt，如果是那就拒绝删除即可。

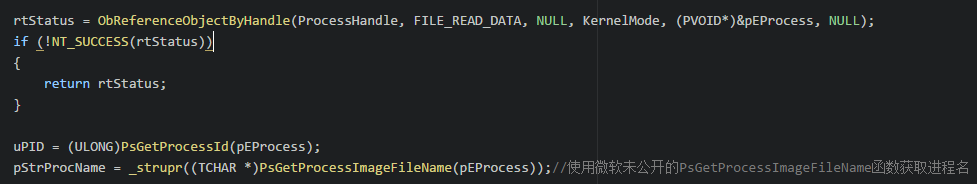


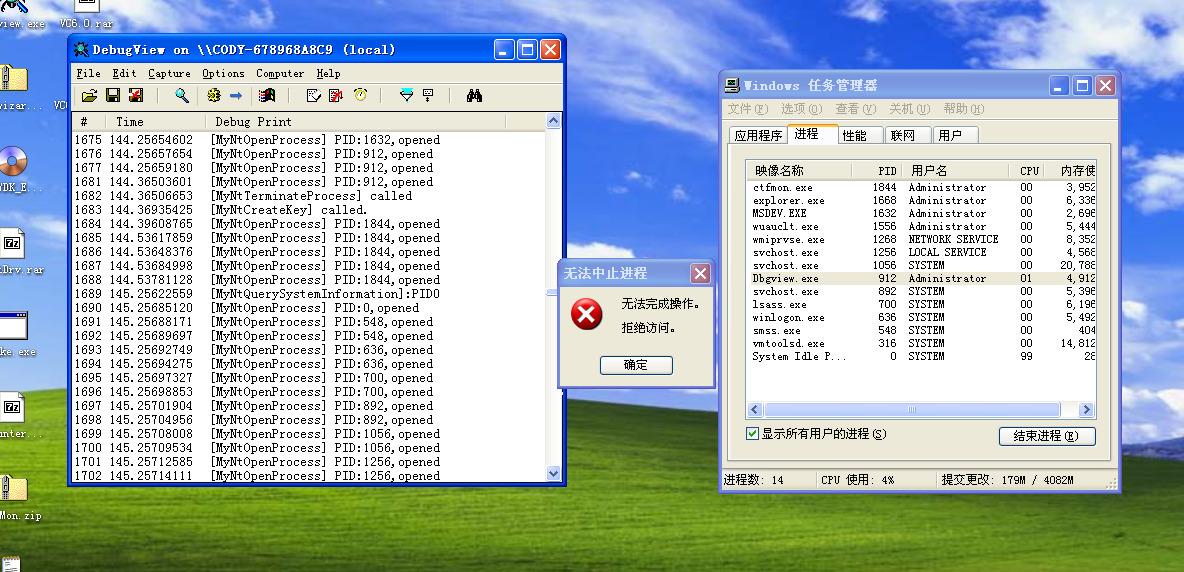
1. **NtTerminateProcess**。通过使用这个函数，我实现了禁止（PID<1000）的进程被强制结束。



数据结构

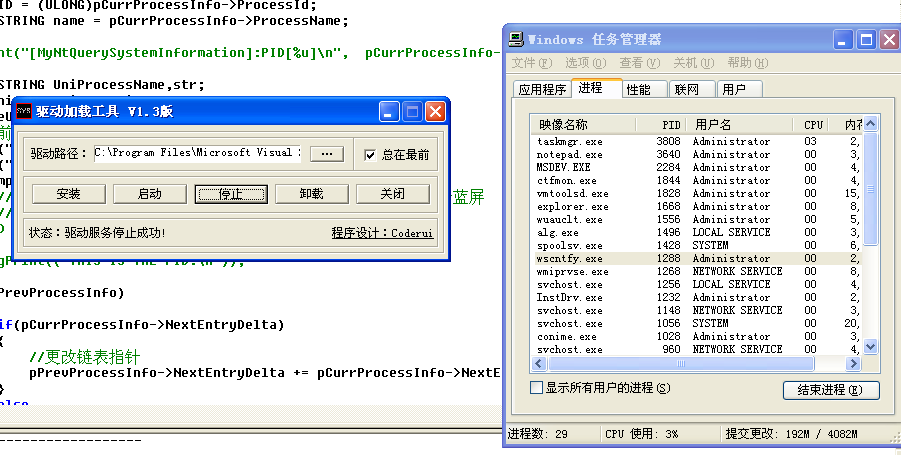
我们通过提供的ProcessHandle获取对应的进程的EProcess对象，从而能够得到对象的pid和name，然后就可以实现我们想要的功能。

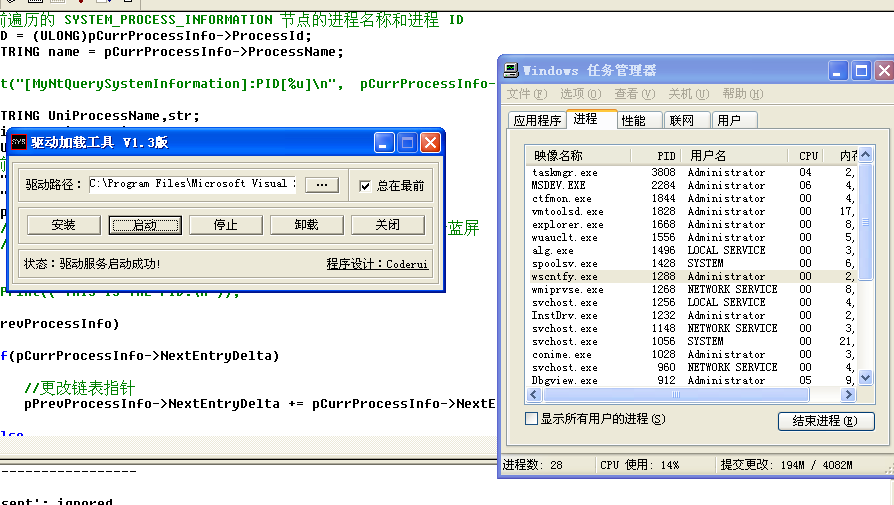




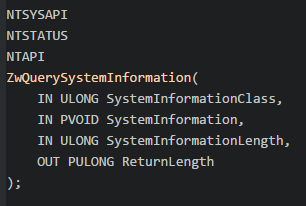
禁止强制结束PID<1000的进程

1. **NtQuerySystemInformation**。通过使用这个函数，以课件上的代码为参考，我实现了对于特定进程的隐藏。具体实现过程：通过遍历进程信息的链表，当找到了notepad进程就将notepad进程前一个链表项和notepad进程后一个链表项相连即可。

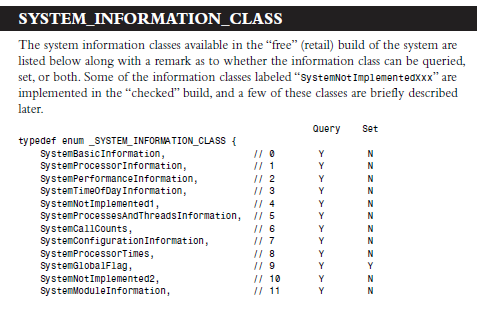




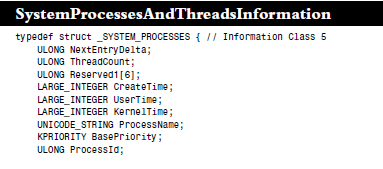
notepad进程被隐藏

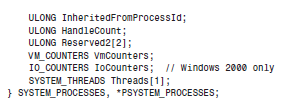


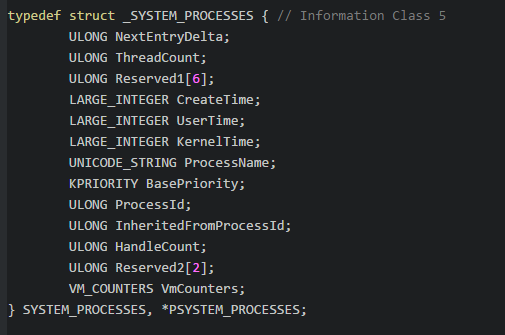
函数原型



我们选择使用SystemInformation == 5时候的信息

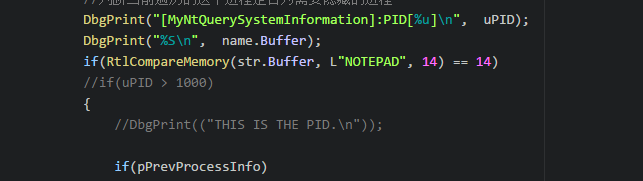






SystemInformation == 5时使用的数据结构

1. **遇到的问题和困难**
2. 在调试程序的过程中，常常会遇到蓝屏的发生，然而通常我都无法明白导致蓝屏的原因到底是什么。
3. 由于之前没有经验，配实验环境花费了较多时间
4. 由于对于windows内核编程不了解，所以对于很多问题都会花费较多时间去寻找答案
5. 本次实验完成的内容相当于一种测试的性质，所以在实现具体的挂钩函数时，考虑有所欠缺，只用作展示，缺乏可扩展性。
6. 一开始，我使用dbgview一直无法输出unicode string，在输出信息中只能显示常量字符串，而且在调试MyNtQuerySystemInformation的下面这一部分if判断语句时，时不时蓝屏



然后后来突然就好了，由于并没有使用git做版本管理，所以也不知道原来的问题在哪里。

1. **总结感想**

在本次实验中，我完成的内容很大一部分有一种测试的性质，所以在实现具体的挂钩函数时，考虑有所欠缺，只能用作展示，缺乏可扩展性和实用性。但是ssdt hook的框架是通用的。在未来，如果想要实现更人性化，更具体的挂钩函数的话，我还要对windows内核编程有更多的了解，要扎实基础，才能更好的写出ssdt hook的驱动程序。

总之，通过这次实践，我对ssdt hook有了更好的了解，也了解了一些windows内核编程的一些内容，也让我明白了要打好基础，不断学习才会有更多的进步与提升。

1. **参考资料**
2. 《寒江独钓》
3. 《Windows NT 2000 Native API Reference.pdf》
4. 课件：windows 木马后门检测.pptx
5. 参考博客 <http://www.cnblogs.com/BoyXiao/archive/2011/09/03/2164574.html>
6. 参考博客 <http://blog.csdn.net/yxyhack/article/details/5391698>