doi :10.3969/j.issn.1671-1122.2011.07.009

Windows 7 SP1 下 DKOM 防攻击技术研究

尹亮，文伟平

（北京大学，北京 102600）

摘 要：Windows 在内存中存储了一些记账信息，用于管理进程、线程、设备驱动程序等对象，并报告给用户，向用户反映系统的运行状况。由于这些信息位于内存中，因此可以直接对其进行修改。文章以微软公司最新的操作系统 Windows 7 SP1 为平台，揭示了 Windows 7 SP1 为进程、线程、驱动等对象建立的一系列结构体和链表，并提出了几种方法，通过修改这些结构体和链表，达到保护、隐藏进程和驱动的目的。

关键词：DKOM；Rootkit；进程隐藏；进程保护；驱动隐藏

中图分类号：TP393.08 文献标识码：A 文章编号：1671-1122（2011）07-0023-03

Windows 7 SP1 for DKOM Under the Attack Technology Research

YIN Liang, WEN Wei-Ping

(*Peking University, Beijing* 102600, *China*)

Abstract: Windows records some information in memory, for the purpose of managing objects like process, thread, driver, and reporting the status to user. Because the information is in memory, we can modify it. This paper will show some structs and lists that Windows 7 SP1 created, and introduce some methods to protect and hide processes and drivers.

Key words: DKOM; rootkit; hide process; protect process; hide driver

0 引言

DKOM，全称是：Direct Kernel Object Manipulation，即直接内核对象操作技术。对于运行在系统中的进程、线程、设备驱动程序等对象，Windows 7 SP1 在内存中存储了一些记账信息，由对象管理器管理，通常采用结构体形式，以链表或者树的形式组织起来。本文揭示了 Windows 7 SP1 为进程、线程、驱动等对象建立的一些结构体和链表，并提出了一些思路和方法，通过修改这些结构体和链表，达到保护、隐藏进程和驱动的目的，以便更多的同行做进一步的研究和参考。

1 Rootkit 实现常用技术介绍

Rootkit 是可以在计算机上实现隐藏作用的一系列代码和程序，其主要功能就是实现隐藏，其中包括进程的隐藏，注册表的隐藏，文件的隐藏等，保护注入的病毒、后门程序、木马等不被系统发现，保证尽可能长期地生存于系统之中，达到攻击者的目的。

在 Windows 的日常应用中，通常会用到如下的一些程序：Taskmgr.exe：任务管理器，显示进程、结束进程；Explorer.exe：资源管理器，显示文件、删除文件；Regedit.exe：注册表编辑器，显示注册表，编辑、删除注册表。

这些程序为了完成相应的功能，会进行系统调用，利用操作系统提供的一些 API 来获得系统的信息。在进行系统调用时，通过 SYSENTER 来陷入内核，并在 EAX 寄存器中存储系统调用号，通过系统调用号查询 SSDT 得到服务函数的地址，然后调用服务函数完成相应功能。Rootkit 通常会对 SSDT 中服务函数的地址进行替换，执行自定义的函数，过滤掉不希望显示的信息，对自身进程拒绝结束、文件拒绝删除，从而达到隐藏和保护的目的。这种技术很容易被检测到，只需要从磁盘上读取原始的 ntoskrnl.exe 或 ntkrnlpa.exe，与现在内存中的内容做对比，即可检测到服务函数的地址是否被替换。

收稿时间：2011-06-10

作者简介：尹亮（1986-），男，湖南，硕士研究生，主要研究方向：系统与网络安全；文伟平（1976-），男，湖南，副教授，博士，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。

DKOM 技术可以直接修改内存中现有对象或结构，并且 Windows 不会在磁盘或其他地方保存有对象或结构的备份，因此无法比较现有对象或结构是否被修改了，相对于替换 SSDT 服务函数地址、过滤 SSDT 服务函数返回结果，DKOM 技术更难以检测。

2 DKOM 技术安全性分析

2.1 进程链表

在 Windows 7 SP1 中，进程和线程在内存中都有对应的对象或数据结构，其中描述进程的是 EPROCESS 结构体，描述线程的是 ETHREAD 结构体，每一个进程都对应一个 EPROCESS 结构体。在 EPROCESS 结构体中有个成员叫 ActiveProcessLinks，它是一个 LIST_ENTRY 结构体，这个结构有两个指针成员 FLINK 和 BLINK，分别指向当前进程的前一个和后一个进程 EPROCESS 结构体中的 ActiveProcessLinks 成员。系统中所有进程的 EPROCESS 结构体通过 ActiveProcessLinks 成员构成了一个双向链表，链表表头叫 PsActiveProcessHead，也是一个 LIST_ENTRY 结构体，它位于 ntoskrnl.exe（或 ntkrnlpa.exe）中，如图 1 所示：

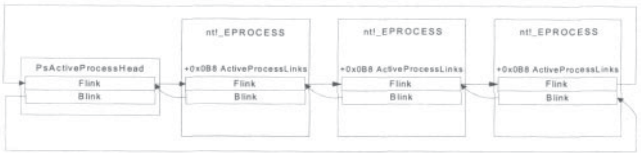

图1 相关结构示意图

图 1 中 +0x0B8 指明了 ActiveProcessLinks 成员在 EPROCESS 结构体中的偏移。

在 Windows 7 SP1 中，任务管理器将会调用 NtQuerySystemInformation 函数来获取系统中所有的进程信息。NtQuerySystemInformation 函数会遍历上面提到的 PsActiveProcessHead 链表，收集链表中的每一个 EPROCESS 结构体相关信息返回给用户，用户可以在任务管理器中结束某一进程。

2.2 线程链表

当创建一个进程时，Windows 会给进程建立一个 EPROCESS 结构，EPROCESS 结构第一个成员是 KPROCESS 结构；当创建一个线程时，给线程建立一个 ETHREAD 结构，ETHREAD 结构第一个成员是 KTHREAD 结构。KTHREAD 和 ETHREAD 中分别有一个名为 ThreadListEntry 的成员，它也是一个 LIST_ENTRY 结构体，用于构成一个双向链表，链表的表头为 KPROCESS 和 EPROCESS 中的 ThreadListHead。\

一个进程所有的线程都位于这两条线程链表中，当进程新建一个线程时，线程的 KTHREAD 和 ETHREAD 会加入到线程链表中，进程的一个线程退出时，线程的 KTHREAD 和 ETHREAD 会从线程链表中移除。

在上文中提到，在 Windows 中，以线程为单位执行调度，只有当一个进程的所有线程都结束了，进程才会被系统自动结束。Windows 的任务管理器在结束一个进程时，将会调用 NtTerminateProcess 函数，这个函数会遍历进程的线程链表，结束链表中的每一个线程，当最后一个线程被结束时，Windows 将会释放为进程分配的资源，删除进程的 EPROCESS 结构体，真正结束进程。360 安全卫士、XueTr 等安全软件结束进程的操作也是遍历线程链表，然后用自己的方法结束其中每一个线程。

2.3 驱动链表

当系统加载一个驱动时，会为这个驱动建立一个 _KLDR_DATA_TABLE_ENTRY 结构体，以下是 WRK 中 _KLDR_DATA_TABLE_ENTRY 结构体的定义：

```
typedef struct _KLDR_DATA_TABLE_ENTRY {
    LIST_ENTRY InLoadOrderLinks;
    PVOID ExceptionTable;
    ULONG ExceptionTableSize;
    PVOID GpValue;
    PNON_PAGED_DEBUG_INFO NonPagedDebugInfo;
    PVOID DllBase;
    PVOID EntryPoint;
    ULONG SizeOfImage;
    UNICODE_STRING FullDllName;
    UNICODE_STRING BaseDllName;
    ULONG Flags;
    USHORT LoadCount;
    USHORT __Unused5;
    PVOID SectionPointer;
    ULONG CheckSum;
    PVOID LoadedImports;
    PVOID PatchInformation;
} KLDR_DATA_TABLE_ENTRY, *PKLDR_DATA_TABLE_ENTRY;
```

其中 PVOID DllBase; 成员指明了驱动的加载基址；
UNICODE_STRING FullDllName; 指明了驱动 .sys 文件的全路径；

所有驱动的 KLDR_DATA_TABLE_ENTRY 结构体通过 InLoadOrderLinks 成员链接起来，链表头部为 PsLoadedModuleList，系统通过遍历 PsLoadedModuleList 链表来得到所有加载的驱动。

3 DKOM 技术应用

3.1 测试环境说明

本文所采用的操作系统为微软公司最新的操作系统 Windows 7 SP1。部分测试中用到了一些安全软件，名称及版本如下：

360 安全卫士 8.0.0.2001，下载地址：http://www.360.cn/；

XueTr 0.39，近年推出的一款广受好评的 ARK 工具，支持 Windows 7，下载地址：http://www.xuetr.com/。

3.2 隐藏进程

如果需要隐藏某一个进程，使得它不会在任务管理器中显示，只需要把该进程的 EPROCESS 结构体从这个双向链表中摘除即可。摘除的方法如下：把需要隐藏进程的前一个进程的 BLINK 修改为当前进程的 BLINK，再把需要隐藏进程的后一个进程的 FLINK 修改为当前进程的 FLINK。如图 2 所示：

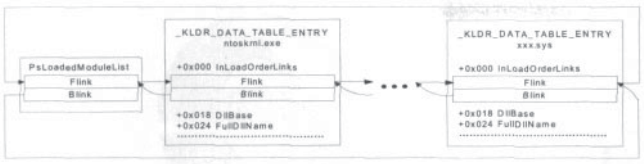

图2 相关进程示意图

在 Windows 中，以线程为单位执行调度，在线程调度时，不会访问 PsActiveProcessHead 链表，所以把进程的 EPROCESS 结构体从链表中摘除并不会影响进程的正常运行。在实际测试中，以隐藏任务管理器（taskmgr.exe）自身为目标。

图3 任务管理器的截图

如图 3 所示，“ 映象名称 ” 以字母顺序排序，可以看到其中没有显示 taskmgr.exe 进程，任务管理器正常功能不受影响，仍然可以用来显示应用程序、结束其他进程、显示服务等等。

3.3 保护进程不被结束

如果需要保护一个进程不会被任务管理器、360 安全卫士、XueTr 结束，只要将进程 EPROCESS 和 KPROCESS 结构体中的线程链表置空，让它们无法获取到受保护进程的线程，从而无法结束受保护进程的线程，达到保护进程的目的。Windows 在执行线程调度时，不会访问 EPROCESS 结构体中的线程链表，所以把线程链表置空并不会影响进程的正常运行。

3.4 隐藏驱动

如果采用 DKOM 技术将某一驱动对应的 KLDR_DATA_TABLE_ENTRY 结构体中的 DllBase 成员设为 0，XueTr 0.39 在枚举驱动模块时，不会认为这是一个有效的驱动，达到隐藏驱动的目的。

测试中使用驱动加载工具加载了一个测试驱动 test1.sy，并将该驱动对应的 KLDR_DATA_TABLE_ENTRY 结构体中的 DllBase 成员设为 0，采用 XueTr 0.39 进行检测。

图4 测试截图示意图

从图 4 中可以看到加载顺序 161 没有显示，这样就发现了隐藏的驱动 test1.sys。

3.5 保护驱动不被卸载

当系统加载一个驱动时，Windows 在建立会 KLDR_DATA_TABLE_ENTRY 结构体的同时，会建立一个 DRIVER_OBJECT 结构体，用于描述这个驱动对象，其中的 DriverSection 成员指向驱动对应的 KLDR_DATA_TABLE_ENTRY 结构体。

在系统卸载一个驱动时，会判断 DRIVER_OBJECT 结构体中 DriverSection 成员是否为空，如果不为空的话，将会调用 MmUnloadSystemImage 函数将驱动映象文件从内存中卸载掉。因此可以采用 DKOM 技术，将受保护驱动对应的 DRIVER_OBJECT 结构体中 DriverSection 成员设为空，这样就可以使得需要保护驱动的映象文件不会从内存中被卸载。

4 DKOM 技术相关分析

为了隐藏自身不被发现，保护自身不被结束和卸载，DKOM 技术较多地被 Rootkit、木马所采用。本文只展示了 DKOM 技术在进程、驱动的隐藏和保护上的应用，但该技术不限于对这两个对象的隐藏和保护，也可以用于文件、网络连接等对象的隐藏和保护上。

另外，Windows 可能为同一种对象建立了不同的结构体，并且同一个结构体有可能以其他方式组织起来。例如，对于进程的 EPROCESS 结构体，通过 ActiveProcessLinks 成员组织成一条链表，除此之外，Windows 在内核中还维护了一个句柄表 PspCidTable，里面也记录了进程的 EPROCESS 结构体地址，通过 PspCidTable 同样可以遍历得到系统里面所有的进程信息。因此 DKOM 技术的难点是，如何尽可能多地知道 Windows 为同一种对象建立了多少种结构体来描述，这些结构体有多少种组织方法，通过哪些函数可以获取到这些结构信息等等。

DKOM 技术直接对保存在内存中的信息进行修改，不需要钩住函数调用，也不需要过滤函数返回结果，同时 Windows 不会在磁盘或其他位置保存相同的信息（即使有，同样也可以修改），因此很难检测。结合 DKOM 技术和其他隐藏方法，可以实现更为强大的隐藏、保护。

5 结束语

本文揭示了 Windows 7 SP1 为进程、线程、驱动等对象建立的一些结构体和链表，并提出了一些思路和方法，通过修改这些结构体和链表，达到了保护、隐藏进程和驱动的目的，这些思路和方法可以推广到其他版本的 Windows 中。另外，本文还指出了 DKOM 技术的难点，这些难点值得继续研究、分析。（责编 杨晨）

参考文献：

[1] 霍格兰德，巴特勒 . ROOTKITS——Windows 内核的安全防护 [M]. 韩智文 . 北京 ：清华大学出版社，2007.4.

[2] Bill Blunden.The Rootkit Arsenal[M]. Jones & Bartlett. 2009.

[3] Mark E，David A. 深入解析 Windows 操作系统 [M]. 北京 ：电子工业出版社，2007.4.

[4] 潘爱民 .Windows 内核原理与实现 [M]. 北京 ：电子工业出版社，2010.5.

[5] 毛德操 .Windows 内核情景分析 [M]. 北京 ：电子工业出版社，2009.5.