Windows 7 遍历 PspCidTable 表检测隐藏进程①

周利荣[1]，马文龙[2]

[1](浙江衢州职业技术学院 图书馆，衢州 324000)
[2](浙江衢州职业技术学院 信息与工程学院，衢州 324000)

摘 要：PspCidTable 表保存着所有进程和线程对象的指针，遍历 PspCidTable 表可以枚举所有进程包括隐藏进程。分析了 windows 7 的 PspCidTable 表的结构，论述了 windows 7 的 PspCidTable 表的内存地址获取方法，遍历 PspCidTable 表的算法，最后给出自动检测的实现步骤及方法。在 windows 7 操作系统上实验表明可高效枚举所有进程，包括通过挂钩枚举进程的函数或进入内核空间直接修改内核数据来达到隐藏自身目的的进程。

关键词：进程；PspCidTable；指针；内核

Windows 7 Ergodice PspCidTable to detect hidden Processes

ZHOU Li-Rong[1], MA Wen-Long[2]

[1](Library, Quzhou College of Technology, Quzhou 324000, China)
[2](Information and Engineering College, Quzhou College of Technology, Quzhou 324000,China)

Abstract: PspCidTable preserves all pointer of processes and threads, Ergodicing PspCidTable can enumerate all processes include hidden processes. The paper analyses the structure of windows 7's PspCidTable, expounds the methed to obtain memory address of windows 7's PspCidTable. The algorithm of Ergodicing PspCidTable, finally brings up the step and methed to automatically detect processes. Experiments on windows 7 operation system showed that the algorithm can enumerate all processes with high efficiency, include processes that hooked functions that enumerated processes or directly entered into kernel space changed kernel data to hide self.

Key words: process; PspCidTable; pointer; Kernel

任何病毒和木马的活动都和进程有关,因此,查看系统中活动的进程成为我们检测病毒最直接的方法。但病毒进程为了逃避检测，常常进入内核空间，修改内核枚举进程的 API 或直接修改内核数据，让它们返回的数据始终遗漏病毒自身进程的信息，一般的进程检测工具自然就检测不到病毒进程了[1]。pspCidTable 是内核未导出的 HANDLE_TALBE 结构，它保存着所有进程和线程对象的指针。只要能遍历这个 PspCid Table 句柄表，就可以遍历到系统的所有进程，包括所有隐藏进程，除非你抹掉 pspCidTable 表。

1 PspCidTable表及结构分析

在 windows 下所有的资源都是用对象的方式进行管理的，诸如：文件、进程、设备等都是对象。当要访问一个对象的时候，如打开一个文件，系统就会创建一个对象句柄，通过这个句柄可以对这个文件进行各种的操作。句柄和对象的联系是通过句柄表来进行的，准确来说一个句柄就是它所对应的对象在句柄表中的索引。通过句柄，可以在句柄表中找到对象的指针，通过指针对对象进行操作。PspCidTable 就是这样的一种表，它不属于任何进程，也不连接在系统的句柄表上，通过它可以访问系统的任何对象。遍历 PspCidTable 句柄表首先要获取 PspCidTable 表的内存地址。由于进程活动是动态的，一个进程包括多个线程，windows 7 操作系统的 PspCidTable 句柄表采用多层表及动态扩展的方法保存进程和线程对象的指针。因此需要分析 PspCidTable 句柄表的层次结构，写出遍历 PspCidTable 表检测隐藏进程的算法，最后编程实现

① 基金项目:衢州职业技术学院科研项目(QZYY1023)
收稿时间:2010-12-24;收到修改稿时间:2011-02-02

© 中国科学院软件研究所 http://www.c-s-a.org.cn

自动检测隐藏进程。

1.1 获取 PspCidTable 表的内存地址的方法

Windows 内核调试工具 windbg 的 dd 命令可以显示内核变量的物理地址。执行 lkd> dd PspCidTable 查看 PspCidTable 的地址及对应内容如下：[2]

83d6b674 88c014b0 00000000 80000018
00000101
83d6b684 8000039c 80000020 00000000
00000000
……

其中下划线部分“83d6b674”正是 PspCidTable 表的内存地址。

PsLookupProcessByProcessId 函数的功能是由进程 ID 得到进程的 eprocess, 由于函数引用了 PspCidTable 表指针，因此可以从该函数中找到 PspCidTable 表的内存地址。Windows 内核调试工具 windbg 的 uf 命令可以把二进制进行反汇编并显示汇编代码，帮助在没有源代码的情况下分析函数。执行 lkd> uf nt! PsLookUpProcessByProcessId 得到 PsLookupProcessByProcessId 函数汇编代码,其中两行为：

83e9dca6 8b3d74b6d683 mov edi,dword ptr [nt!PspCidTable (83d6b674)]
83e9dcac e88727fbff call nt! ExMap HandleToPointer(83e50438)

内存存放顺序跟我们看的不一样，是以 2 字节为单位压栈压进去的，所以逻辑上的顺序和内存上的顺序是正好相反的,所以下划线部分以 2 个字节倒过来就是“83d6b674”，与“dd PspCidTable”命令得到的 PspCidTable 表的内存地址相同。所以可用“83d6b674”前两个字节“3d8b”及后一个字节“e8”作为函数搜索特征串。获取内核函数的地址可用函数 MmGetSystemRoutineAddress()，假设返回值保存在变量 FuncAddr 中，则从 FuncAddr 开始搜索，如果*FuncAddr==0x3d8b 并且*（FuncAddr+6）==0xe8，则*（FuncAddr+2）即为 PspCidTable 表的内存地址。

1.2 PspCidTable 的层次结构

PspCidTable 是一个 HANDLE_TALBE 结构，当新建一个进程时，对应的会在 PspCidTable 存在一个该进程和线程对应的 HANDLE_TABLE_ENTRY 项。在 windows 7 中采用动态扩展的方法，当句柄数少的时候就采用下层表，多的时候才启用中层表或上层表。句柄表分三层，下层表是一个 HANDLE_TABLE_ENTRY 项的索引，整个表共有 256 个元素，每个元素是一个 8 个字节长的 HANDLE_TABLE_ENTRY 项及索引，HANDLE_TABLE_ENTRY 项中保存着指向对象的指针, 下层表可以看成是进程和线程的稠密索引。中层表共有 256 个元素，每个元素是 4 个字节长的指向下层表的入口指针及索引, 中层表可以看成是进程和线程的稀疏索引。上层表共有 256 个元素，每个元素是 4 个字节长的指向中层表的入口指针及索引，上层表可以看成是中层表的稀疏索引。一个句柄表有一个上层表，一个上层表最多可以有 256 个中层表的入口指针,每个中层表最多可以有 256 个下层表的入口指针,每个下层表最多可以有 256 个进程和线程对象的指针。PspCidTable 表可以看成是 HANDLE_TABLE_ENTRY 项的多级索引。

1.3 确定 PspCidTable 的层数

在 windbg 下用 dd 命令查看 PspCidTable 的地址及对应内容如下[2]：

① lkd> dd PspCidTable
83d6b674 88c014b0 00000000 80000018
00000101
83d6b684 8000039c 80000020 00000000
00000000
……

其中下划线线部分“88c014b0”是 HANDLE_ TABLE 的地址。在 windbg 下用 dt 命令显示? HANDLE_TALBE 的部分结构如下[2]：

② lkd>dt _handle_table 88c014b0
nt!_HANDLE_TABLE
+0x000 TableCode : 0x9078b001
+0x004 QuotaProcess : (null)
……

这里的 tableCode 记录这句柄表的地址，如果后两位是 00 则采用的是一层表，01 是两层表，10 是三层表。通过上面的 TableCode 就可以判断这个系统采用了两层表的结构。如果系统采用了两层或者三层的时候，TableCode 就不是句柄表的地址了，把这个值后两位置为 0 之后，则是一个指向多层表的指针[3]。

2 遍历PspCidTable的算法实现。

2.1 确定 PspCidTable 的层数及入口指针的算法

首先调用自定义函数 GetPspCidTable() 得到

© 中国科学院软件研究所 http://www.c-s-a.org.cn

PspCidTable 的地址，保存在指针变量 PspCidTable 中。*PspCidTable 为 HandleTable 的地址, ，保存在指针变量 HandleTable 中。*HandleTable 即为 TableCode 的值。TableCode 的最后两位决定了句柄表的层数, 保存在 flag 变量中，将 TableCode 最后两位置 0，则 TableCode 是一个指向多层表的指针。部分代码如下：

```
PspCidTable=GetPspCidTable();
HandleTable=*(PULONG)PspCidTable;
TableCode=*(PULONG)HandleTable;
flag=TableCode&3;
TableCode&=0xfffffffc;
```

2.2 遍历 PspCidTable 表算法

（1）从下层表开始遍历的函数 BrwsTableL3( )

```
ULONG BrwsTableL3(ULONG TableAddr)
{……    //变量声明部分省
do    { TableAddr+=8;
Object=*(PULONG)TableAddr;
Object&=0xfffffff8;// Object 最后三位置 0,Objec
即为 EPROCESS 的地址
if (Object==0)  {    continue;    }
dwProcessId  =    * ( (PULONG) (Object + dwPidOffset ) ) ; // dwPidOffset 为进程 id 在 EPROCESS 中偏
if(dwProcessId <65536 && dwProcessId !=0
{flags=*(PULONG)(Object+ flag   );
//EPROCESS 中 Flags 偏移量，指明了进程是否结束
if((flags&0xc)!=0xc)
pImageFileName= getProcessName (Object);}
//自定义函数获取进程全路径
} while (--ItemCount>0);
return 0;}
```

（2）从中层表开始遍历的函数 BrwsTableL2( )

```
ULONG BrwsTableL2(ULONG TableAddr)
{   do    { BrwsTableL3(*(PULONG)TableAddr);
  TableAddr+=4; //中层表每项占四个字节
} while ((*(PULONG)TableAddr)!=0);
return 0;}
```

（3）从上层级表开始遍历的函数 BrwsTableL1( )

```
ULONG   BrwsTableL1(ULONG TableAddr)
{ do    { BrwsTableL2(*(PULONG)TableAddr);
  TableAddr+=4; //上层表每项占四个字节
} while ((*(PULONG)TableAddr)!=0);
return 0;}
```

有了上述三个函数及变量 flag 及 TableCode 的值，就可以遍历 PspCidTable 了。如果 flag 等于 0，则 PspCidTable 只有下层表，TableCode 是下层表入口指针，调用 BrwsTableL3(TableCode) 遍历 PspCidTable 表。如果 flag 等于 1，则 PspCidTable 有两层表，TableCode 是中层表入口指针，调用 BrwsTableL2 (TableCode) 遍历 PspCidTable 表。如果 flag 等于 3，则 PspCidTable 有三层表，TableCode 是上层表入口指针，调用 BrwsTableL1(TableCode) 遍历 PspCidTable 表。

3 自动检测隐藏进程的实现

3.1 检测隐藏进程的一般方法

（1）用 VS.NET 编写应用程序 Query.exe 在用户态枚举进程。方法：首先用 EnumProcesses()获取进程 ID 集合，用 OpenProcesses()函数打开进程返回的句柄需进程 ID,用 EnumProcessModules()来枚举进程模块,用 GetModuleFileNameEx（）获得可执行文件的模块路径。应用程序建立文本文件，将进程名保存在文本文件中。

（2）由于 PspCidTable 表属于内核空间，用户程序是无法访问的。可以编写驱动程序进入内核遍历 PspCidTable 表检测进程，当驱动程序启动时，操作系统会首先调用 DriverEntry()函数，DriverEntry()函数是驱动程序的入口点。DriverEntry() 函数又调用 PsCreateSystemThread(&hThread,(ACCESS_MASK)0, NULL，(HANDLE)0，NULL，WorkThread，NULL) 函数在内核中创建线程,第四个参数为空，则创建系统线程,第六个参数 WorkThread 是自定义函数，其功能是遍历 PspCidTable 表检测进程，也是新线程的运行地址。WorkThread 函数需将检测到的进程名保存在文本文件中，方法是调用 ZwCreateFile()内核函数创建文本文件，调用内核函数 ZwWriteFile()将进程名保存在文本文件中[4]。

（3）第(2)步生成的文件是驱动程序文件，扩展名为.sys,不能自动执行，需用 VS.NET 编写能够启动.sys 文件的驱动加载应用程序 PspCidTable.exe。方法是①调用 OpenSCManager(NULL,NULL,SC_MANAGER_ALL_ACCESS)函数打开服务控制管理器。

© 中国科学院软件研究所 http://www.c-s-a.org.cn

②调用 CreateService()函数创建一个服务，服务类型为内核驱动。③调用 OpenService()取得服务句柄。④调用 StartService()启动服务。⑤调用 ControlService()停止服务。⑥调用 DeleteService()删除服务。⑦调用 CloseServiceHandle()关闭服务句柄[4]。

（4）用 VB.NET 编写应用程序,在窗体中放两个 ListBox 控件，调用 Query.exe 程序枚举进程，结果保存在文本文件中，打开对应的文本文件，将进程名添加到第一个 ListBox 控件中。调用程序 PspCidTable.exe 进入内核遍历 PspCidTable 表检测进程，结果保存在文本文件中，打开对应的文本文件，将进程名添加到第二个 ListBox 控件中。程序比较两个控件的内容，若第二个 ListBox 控件中的内容是第一个 ListBox 控件所没有的，可视为隐藏进程。

3.2 提高检测隐藏进程成功率的改进措施

要真正检测到隐藏进程，需要解决许多细节问题，否则很可能将正常进程当成隐藏进程，或检测不到隐藏进程。主要有：

(1) 内核中进程名保存在结构体 EPROCESS 的 ImageFileName 变量中，但 EPROCESS.ImageFileName 变量只能显示 15 个字节，超过 15 字节的进程名将截断显示，且不显示进程路径。因此不能从 EPROCESS.ImageFileName 变量获取进程名。在结构体 EPROCESS 中的成员变量 SeAuditProcessCreationInfo 中有 ImageFileName 字段，类型为_OBJECT_NAME_ INFORMATION，可以从该字段获取内核模式的全路径和进程名[4]。

（2）遍历 PspCidTable 表检测进程包含了结束进程，必须剔除。判断进程是否结束方法是根据 EPROCESS.Flags 的值,若为 0xc 则表明该进程是已结束进程。

4 检测技术的效率与可靠性分析

目前常用隐藏进程检测方法有：(1)遍历 EPROCESS 结构体双向链表,此方法效率高但病毒进程可轻易修改双向链表中的指针，使进程从链表中脱离，在枚举进程时就能绕过要隐藏的进程，因此可靠性不高。(2) 挂钩 SwapContext，在每次线程切换时定位线程，根据线程找到进程。Windows 系统中线程切换是十分频繁，每次线程切换时都要进行进程对比，因此，效率与可靠性均不高。(3) 挂钩系统调用，原理是当进程转向系统调用接口时将其拦截，在处理程序中获取指向当前进程 EPROCESS 的指针，从而与现有的进程列表进行对照以发现隐藏进程，但进程可能长时间处于等待状态并且不进行系统调用，因此，效率与可靠性也不高。(4) 基于内存搜索的隐藏进程检测技术。此方法是在 2G 系统内存中根据 EPROCESS 结构体特征及 OBJECT 对象头类型特征来定位进程，因此搜索效果低，如果进程修改结构体或对象头类型特征值则搜索不到进程，因此可靠性不高。

遍历 PspCidTable 表检测隐藏进程的效率高，进程要逃避检测，必须从 PspCidTable 中删除自身对象，句柄项被用 NULL 替代。但当系统想关闭进程当时候，它将找到 PspCidTable 并且得到一个 NULL 对象指针，这将导致蓝屏。病毒进程在进程被中止之前必须调用 PsSetCreateProcessNotifyRoutine 安装一个进程以 避免系统崩溃。但此实现方法难度较大，因此此检测方法可靠性较高。

5 结语

本文对 windows 7 的 PspCidTable 表的结构进行分析,并给出在 windows 7 操作系统上实现遍历 PspCidTable 表检测隐藏进程的算法，最后编程实现自动检测，并对该检测技术与其它检测技术的效率与可靠性进行比较。经过在计算机上反复实验表明，该检测技术的效率与可靠性均较高，要逃避该检测技术实现进程隐藏需掌握 windows 内核编程有关知识且具有一定难度。

参考文献

1 王驎峰,董亮卫.Windows_2000_XP_下隐藏进程的检测机制.计算机工程,2006,32(20):95−96,99.

2 聂雪军.Windows 高级调试.北京:机械工业出版社,2009.120−124.

3 薛英飞,单蓉胜,李小勇.Windows(2000/2003)下进程隐藏检测.信息安全与通信保密,2007,(12):70−71.

4 王建华,张焕生,侯丽坤.Windows 核心编程.北京:机械工业出版社,2006.45−56.

© 中国科学院软件研究所 http://www.c-s-a.org.cn