文章编号:1001 - 9081(2009)01 - 0175 - 03

基于内存搜索的隐藏进程检测技术

胡和君[1,2],范明钰[1,2]
(1. 电子科技大学 计算机科学与工程学院,成都 610054; 2. 电子科技大学 信息安全研究中心,成都 610054)
(huhejun1984@yahoo.com.cn)

摘 要:对现有的 Windows 下各种隐藏进程检测技术及其反检测技术进行了研究,提出了基于内存搜索的隐藏进程检测技术,并针对该技术的性能提出了改进。该种检测技术利用进程的固有特征对系统地址空间的遍历建立完整的进程列表来检测隐藏进程。通过实验表明,该技术具有较好的可靠性、检测效率和完整性。

关键词:内存搜索;进程隐藏;Rootkit
中图分类号: TP316.7 **文献标志码**:A

Hidden process detection technique based on memory search

HU He-jun[1, 2], FAN Ming-yu[1, 2]
(1. *School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*;
2. *Research Center of Information Security, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*)

Abstract: To research the existing hidden process detection techniques and its anti-detection techniques in Windows, a new detect method based on the memory search was brought forth and its performance was improved. This technique made use of the inherent characteristics of process to traverse the system address space for establishing integrated process list, and then detected hidden process. Experiments show that this detection method is of higher reliability, efficiency and integrity.

Key words: memory search; hidden process; Rootkit

0 引言

随着网络攻击的目的逐渐从炫耀技术转向获取经济利益和情报,以窃取主机控制权和敏感信息为目标的恶意程序迅速增加。对于此类恶意程序,隐藏自身、保障生存是首要需求,实现这一功能的恶意代码称作 Rootkit。Rootkit 的隐藏特性严重影响系统的安全性和可靠性,而 Rootkit 的隐藏进程的特性又是系统管理员和安全检测软件所面临的最为迫切的安全威胁。

1 Rootkit 隐藏进程技术

Rootkit 进程的隐藏技术可以在用户态和内核态两个层次来实现[1]。用户态的进程隐藏实现机制包括动态链接库注入,应用程序接口挂钩,远程线程注入等方式。内核态的进程隐藏的一般方法包括内核挂钩,直接内核对象操作等,如:挂钩系统服务描述表(System Service Descriptor Table, SSDT)获取重要函数地址、修改内核中存储信息的结构或对象等方式。两种层次的 Rootkit 都可以通过挂钩系统重要函数模块、修改系统中重要数据结构等途径,对系统进程的相关信息进行劫持与修改,以达到欺骗上层安全检测软件而隐藏进程的目的。

2 现有进程隐藏检测技术

现有的隐藏进程检测技术一般有以下几种:

2.1 用内核函数查找

在系统内核中有进程枚举查询类函数如:ZwQuerySystemInformation 等,通过对这类内核函数的使用,来获得进程列表以发现隐藏进程。但是该方法只能对一些用户态层次的 Rootkit 进行检测,它检测不到任何内核态层次 Rootkit 的隐藏进程。

2.2 遍历 EPROCESS 结构体双向链表

由于每一个进程都有自己的地址空间、句柄、线程,等等。这些都与相应的内核数据结构密切相关。Windows 系统中,每一个进程都是由 EPROCESS 结构体来描述的,所有进程的结构体连接成一个环形的双向链表。通过对 EPROCESS 结构体双向链表的遍历检测,获取所有进程的列表以检测隐藏进程[2]。该项方法的不足在于,一旦 Rootkit 修改双向链表中的指针,使隐藏进程从链表中脱离,在枚举进程时就能绕过要隐藏的进程,从而避开检测。

2.3 扫描调度线程表

在 Windows 操作系统中维持了系统调用的线程列表,而线程作为进程运行的真正实体,通过对系统调用的线程列表的遍历,收集所有线程的信息,就可以得到所有进程的列表,从而发现隐藏进程。在 Windows 2000 中有三个双向线程链表:KiWaitInListHead,KiWaitOutListHead 和 KiDispatcherReadyListHead。前两个链表包含的是等待某事件的线程,而第三个链表包含的是准备好执行的线程。遍历链表并减去 ETHREAD 线程结构体中的线程链表偏移,就能得到指向线程的 ETHREAD 的指针。这个结构体中包含了指向该线程所属进程的指针,通过这些指针就能确定拥有该线程的进程,从而构造出所有进程的列表[3]。不过该方法只在 Windows 2000 系统上才能运行。因为在 Windows XP 中的线程调度只有两个

收稿日期:2008 - 07 - 25;**修回日期**:2008 - 09 - 25。
基金项目:国家自然科学基金资助项目(60373109);北京电子科技学院开放基金资助项目(KFHT200704)。
作者简介:胡和君(1984 -),男,四川简阳人,硕士研究生,主要研究方向:计算机信息安全; 范明钰(1962 -),女,四川成都人,教授,博士生导师,主要研究方向:信息安全、网络安全。

线程链表：KiWaitListHead 和 KiDispatcherReadyListHead。可以通过搜索 KeDelayExecutionThread 函数来查找 KeWaitListHead，但是 KiDispatcherReadyListHead 却很难查找，主要的问题是 KiDispatcherReadyListHead 的地址并没有被任何一个导出的函数使用，并且该线程链表在不同的 Windows 系统版本上都不一样。

2.4 挂钩 SwapContext

在 Windows 系统中每个线程切换时，系统会调用 ntoskrml.exe 中的 SwapContext 函数将当前运行线程的上下文与重新执行线程的上下文进行交换。调用 SwapContext 后，EDI 寄存器的值就是下一个要交换进入线程的指针。通过挂钩 SwapContext 函数，可以在每次线程切换时就能定位线程，从而根据线程找到进程。但是该项技术的一个弊端就是性能不高。在 Windows 系统中线程切换是十分频繁的。一旦挂钩 SwapContext，那么在每次线程切换时都要进行进程查询、对比进程列表，在线程切换十分频繁情况下，如此耗时的工作必然影响系统的正常运作。

2.5 挂钩系统调用

在 Windows 操作系统中任何进程的运行都需要通过用户态下 API 与系统协作，而所有的上层 API 函数，在系统的内核态都有对应的系统服务函数，大多数这类 API 服务请求都会通过系统调用转入内核。基于这个事实，隐藏进程要在目标机上工作，也一定会调用上层 API 函数对内核发出服务请求。一般来说，在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[4]。不过该技术还是存在不足，某些进程可能长时间处于等待状态并且不进行系统调用，这样的隐藏进程就无法检测到；另外目前 Rootkit 也可以采用修改执行系统调用的中断或是用全局描述符表（Global Descriptor Table，GDT）中的调用门实现系统调用避开这种方法的检测。

2.6 枚举 PspCidTable

在 Windows 下所有的资源都是以对象的方式进行管理。PspCidTable 是 Windows 系统中一种特殊的句柄表，它不链接在系统句柄表上，也不属于任何一个进程。通过它可以访问系统中所有的对象。系统内所有进程对象的对象类型是一样的，先取得任一进程的类型，然后访问所有可能的句柄值，如果句柄的类型是进程的话，就记录下来，这样的话，系统内所有的进程对象就都被检测出来[5]。但是如果 Rootkit 更改了 PspCidTable 中的信息，那么检测程序就无法将进程对象区分出来。

3 基于内存搜索的隐藏进程检测技术

一个进程要工作，必定有自己的地址空间、句柄、描述符、线程，等等。即使 Rootkit 进程修改了进程结构链表的指针，使得“遍历 EPROCESS 结构体双向链表”方法失效，通过修改执行系统调用的中断或是用 GDT 中的调用门实现系统调用避开“挂钩系统调用”方法的检测，通过修改 PspCidTable 表数据结构来混淆“枚举 PspCidTable”方法的检测。但是这些 Rootkit 技术并不影响进程的任何功能，无论怎样，EPROCESS 结构总是存在的，对一个进程的正常功能来说它是必要的。与此同时，一个进程要运行，必然会加载到内存中。基于这个事实，隐藏进程要在目标机上进行工作，在内存中一定会存在对应的 EPROCESS 结构。基于内存搜索的进程检测技术利用这一原理，可以有效地对隐藏进程进行全面的查找和检测，进而判定系统中是否存在 Rootkit。

3.1 分页处理

由于 Windows 系统的分页机制，我们首先就需要对分页进行检查，因为虚拟地址是被映射过的，一旦搜索到的页是交换到硬盘上的，那么就会引起系统崩溃。检查原理如下：如果我们访问的虚拟地址所在页在物理内存里，虚拟地址所在页相应的 PDE，PTE 就都是有效的，将其交给 CPU 转换成物理地址后访问就可以了。如果页不在物理内存中，那对应的 PDE，PTE 都是无效的。在 WinDbg 中通过 dt _HARDWARE_PTE 命令可以看到有效页表项 PTE 的结构如图 1 所示。

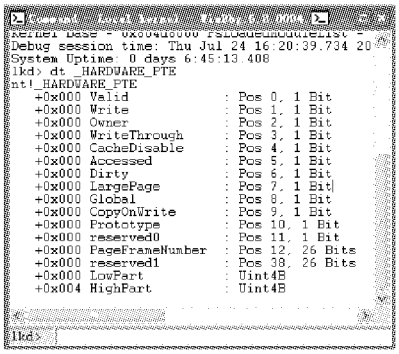

图 1 PTE 结构

我们只关心里面的第 0 位和第 7 位。当第 0 位为 1 时，说明 PTE 有效，对应的页在物理内存中，我们可以安全地进行搜索。当第 7 位的 LargePag 为 1 表明当前虚拟地址大于等于 0x80000000 并且小于 0xa0000000，这种情况，将地址减去 0x80000000 就得到了实际物理地址，也能进行安全地搜索。

分页处理的关键代码如下：

```
ULONG VALIDpage(ULONG addr)
{
  ULONG pte, pde;
  pde = MiGetPdeAddress(addr);
  if((*(PULONG)pde&0x1) != 0){
    if((*(PULONG)pde&0x80) != 0){
      return VALID;
    }
    pte = MiGetPteAddress(addr);
    if((*(PULONG)pte&0x1) != 0){
      return VALID;
    }else{
       return PTE_INVALID;
    }
  }
  return PDE_INVALID;
}
```

3.2 进程搜索

在 Windows 系统中 0x80000000 ~ 0xFFFFFFFF 是系统地址空间[6]。一般 kd debug 的时候很容易看到 EPROCESS 都集中在 0x80000000 ~ 0x90000000 区域。我们只需要对该区域进行遍历搜索就能够对进程进行全面的查找和检测。在 Windows 系统虚拟内存的 4 Gb 地址空间的 1 024 个页表按顺序被映射到了 0xC0000000 ~ 0xC03FFFFF 的 4 Mb 地址空间。第一个 4 Mb 地址空间的页表对应 0xC0000000 开始的 4 kb，以此类推，页目录被映射到了 0xC0300000 开始处的 4 kb 地址空间。也就是说，PTE 是按 0x1000（4 kb）步进，而 PDE 是按 0x400000（4 Mb）递增。

当获取内存页以后，就需要判断该页所存储的信息是否

为 EPROCESS 地址指针。由于每个进程的 PEB 地址指针前半部分都相同[7],所以可以通过比较 PEB 地址指针的前半部分来进行进程的判断。这个用于比较的 PEBAddress 地址在内核实现的时候可以采取如下方法:通过 PsGetCurrentProcess( ) 函数获取当前进程的 PEPROCESS 指针,利用偏移地址得到 PEB 地址指针,用该地址与 0xFFFF0000 做 & 运算,获取 PEBAddress 地址。

在确定了 PEB 地址以后,就可以结合 EPROCESS 的结构,来获取进程信息。进程搜索的主体代码如下:

```
VOID SearchProcess ( )
{
  …
  for( i = 0x80000000; i < 0x90000000; i + = 4) {
    ret = VALIDpage( i);
    if ( ret == VALID) {
      Address = * ( PULONG) i;
      if ( ( Address&0xFFFF0000) == PEBAddress) {
        if ( IsaRealProcess( i) ) {
          ShowProcess( i - PEB_OFFSET);
          i + = EPROCESS_SIZE; }
      }
    } else if( ret == PTE_INVALID) {
       i - = 4; i + = 0x1000;
    } else{ i - = 4; i + = 0x400000; }
  …
  }}
```

3.3　基于内存搜索的隐藏进程检测技术的改进

在实际环境里,EPROCESS 的地址区域不可能到达 0x90000000,因此可以通过确定当前系统中最大的 EPROCESS 地址值来减少循环查找空间,提高效率。在实验中我们发现 System 进程的 EPROCESS 总是排列在所有其他进程的最后,所以我们可以通过获取 System 进程的 EPROCESS 的地址来作为循环查找的最大值条件,来减少遍历查找时间,提高内存搜索速度。

由于采用的是直接搜索内存的方式,与系统进程调度没任何关系,能够枚举到已经结束的进程,因为系统进程调度的时候,仅仅是销毁对应的一系列指针链表结构,而本身的 EPROCESS 结构整体和其他信息在内存中仍然存在。所以需要对搜索到的进程是否结束来进行判断,一种方法就是通过进程的 ExitTime 来判断该进程是否结束。利用 EPROCESS 结构的偏移地址找到 ExitTime -> QuadPart 项,判断该值是否为零,即可知道该进程是否已经结束,因为已经结束的进程的该值为非零。

改进后的基于内存搜索的隐藏进程检测流程如图 2 所示。

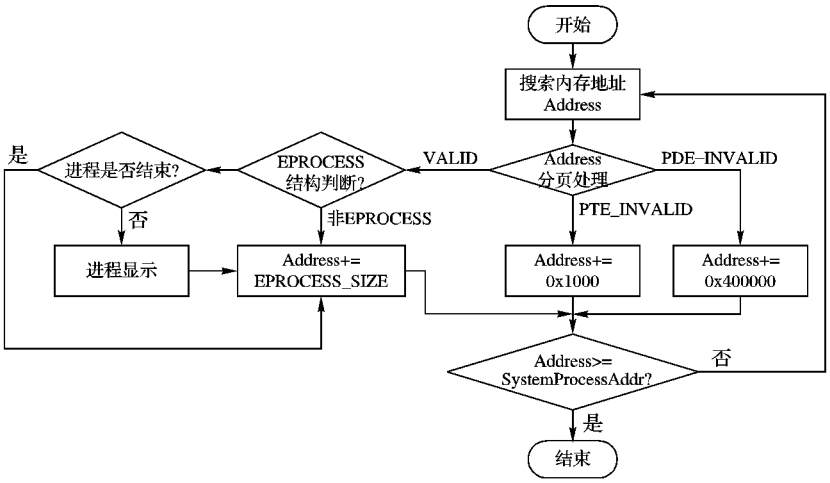

图 2　基于内存搜索的隐藏进程检测流程

4　性能分析与实验结果

4.1　性能分析

从检测效率上看,基于内存搜索的隐藏进程检测技术是通过搜索整个 EPROCESS 区域的内存地址空间而得到进行,比起直接调用内核函数进行查找、直接遍历 EPROCESS 双向链表以及枚举 PspCidTable 来说,其完成时间要长一些。但是比起扫描调度线程来说,其执行速度则要快很多,因为后者需要收集所有线程的信息,再进行分析才能得出结果。挂钩 SwapContext 在每次线程切换时都要进行进程查询、对比,会严重影响系统的正常运作且耗时大。挂钩系统调用要对每次的系统调用进行分析,需要较长的时间才能得出结果。因此基于内存搜索的隐藏进程检测技术在效率上处于优势地位。

从可靠性上看,内核函数查找只能检测用户态层次的进程隐藏;直接遍历 EPROCESS 双向链表会被修改 EPROCESS 链表指针技术所欺骗;枚举 PspCidTable 依靠于 PspCidTable 中已经能够被恶意程序任意修改的句柄类型信息,使得该检测技术失效;扫描调度线程的技术由于其只能工作在 Windows 2000 系统中,而不能适应新系统的要求;挂钩系统调用的技术目前已经能够被修改执行系统调用的中断技术或是用 GDT 中的调用门实现系统调用的技术所避开;挂钩 SwapContext 虽然目前还没有反检测手段,但是其对系统性能的影响成为了该技术的弊端;对于基于内存搜索的隐藏进程检测技术来说,除非恶意的 Rootkit 能够把 EPROCESS 结构修改掉,然而修改 EPROCESS 的结构的同时要保证进程的运行在实现上是不可能的,因此该技术在目前能够检测到各种方法隐藏的进程,具有很好的可靠性

从完整性上看,基于内存搜索的隐藏进程检测技术,由于其实现原理,使得该技术能枚举出已经结束的进程信息。因此该技术比起现有的其检测技术具有明显的完整性上的优势。

4.2　实验结果

为了验证基于内存搜索的隐藏进程检测技术的性能,测试使用了8种基于不同进程隐藏技术的恶意代码,包括

(下转第 188 页)

positories [C]// Proceedings of the 1st International Conference on Web Information Systems Engineering. Hong Kong: IEEE Computer Society, 2000: 138 - 145.

[2] SANDHU R, COYNE E J. Role based access control models[J]. IEEE Computer, 1996, 29(2): 38 - 47.

[3] 顾春华,肖宝亮. RBAC 模型层次关系中的角色权限[J]. 华东理工大学学报: 自然科学版, 2007, 33(1): 96 - 99.

[4] CHADWICK D W, XU W, OTENKO S, *et al*. Multi-session separation of duties for rbac[C]// IEEE 23rd International Conference on Data Engineering Workshop. Istanbul: IEEE Computer Society, 2007. 744 - 753.

[5] 霍晓丽,卢正鼎. 在角色访问控制系统中实现职责分离的方法[J]. 计算机工程与应用, 2006, 42(1): 74 - 76.

[6] GAIL-JOON A. Specification and classification of role-based authorization policies, enabling technologies[C]// Proceedings 12th IEEE International Workshops on Infrastructure for Collaborative Enterprises. Linz: IEEE Computer Society, 2003: 202 - 207.

[7] SHAFIQ B, MASOOD A, JOSHI J. A role-based access control policy verification framework for real-time systems[C]// 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems. Sedona: IEEE Computer Society, 2005: 13 - 20.

[8] 魏定国,吴时霖. 基于 Petri 网的 RBAC 策略验证的研究[J]. 小型微型计算机系统, 2004, 25(5): 827 - 832.

[9] 杜萍,刘弘. 协同设计系统中基于 XML 的访问控制实现[J]. 计算机应用研究, 2007, 24(1): 174 - 176.

[10] 段隆振,文锋,黄水源. 一种描述 RBAC 角色层次关系和互斥关系的模型及实现[J]. 南昌大学学报: 理科版, 2006, 30(6): 601 - 604.

[11] 徐松,赵曦滨,顾明. 网格环境下的分布式 RBAC 模型框架[J]. 计算机工程, 2006, 32(6): 163 - 166.

(上接第 177 页)

hxdef[11]、灰鸽子[12]、Agony[13]、Vanquish[14]、Ntrootkit[15]、Fu[16]和两个未空开的 rootkit 程序 RKX 和 RKZ。将实验所用恶意代码分别安装在测试系统 Windows XP SP3 中,然后分别使用现有的检测技术 A(内核函数查找),B(遍历 EPROCESS),C(挂钩 SwapContext),D(挂钩系统调用),E(枚举 PspCidTable)和本文提出的基于内存搜索的检测技术 F 进行扫描检测。经过多次的实验,将检测结果与检测耗时汇总统计如表 1 所示。

表 1 实验结果

恶意代码	A	B	C	D	E	F
Hxdef	Y	Y	Y	Y	Y	Y
灰鸽子	Y	Y	Y	Y	Y	Y
Agony		Y	Y	Y	Y	Y
Vanquish			Y	Y	Y	Y
Ntrootkit			Y		Y	Y
Fu			Y			Y
RKX			Y			Y
RKZ			Y			Y
耗时	<2 s	<2 s	150 ~ 200 s	50 ~ 90 s	3 ~ 8 s	5 ~ 15 s

从实验结果可以看出,A、B、D、E 检测技术面对相应的反检测技术失去了原有的效果,虽然 C 检测技术也能够检测出所有的隐藏进程,但是由于其耗时太大影响了他的性能。基于内存搜索的隐藏进程检测技术能够检测出所有类型的隐藏进程,并且具有较好的检测效率。

5 结语

基于内存搜索的隐藏进程检测技术是针对现有的 Rootkit 恶意程序中进程隐藏行之有效的检测方法。除非恶意的 Rootkit 能够把 EPROCESS 结构修改掉,但是修改掉 EPROCESS 的结构在实现上难度非常大,因此基于内存搜索的隐藏进程检测技术能够检测到各种方法隐藏的进程。

基于内存搜索的隐藏进程检测只能检测出 Rootkit 的隐藏进程行为,然后通过结束进程来防止 Rootkit 在当前系统环境下的运行,但是并不能够检测出 Rootkit 在系统中隐藏的其他行为,比如隐藏文件、隐藏注册表信息等。所以要完全地对抗 Rootkit 还需要将该技术与其他检测技术进行结合。多种检测技术的共同使用,也是安全检测更为有效的方法。在安全检测技术的研究上,仍然有许多的工作要做。

参考文献:

[1] 杨彦,黄皓. Windows Rootkit 隐藏技术研究[J]. 计算机工程, 2008, 34(12): 152 - 156.

[2] BUTELER J R I L. Detecting compromises of core subsystems and kernel function in WindowsNT/ 2000/ XP[D]. Baltimore County: University of Maryland, 2002.

[3] 梁晓,李毅超. 基于线程调度的进程隐藏检测技术研究[J]. 计算机科学, 2006, 33(10): 114 - 118.

[4] 何志,范明钰. 基于 HSC 的进程隐藏检测技术[J]. 计算机应用, 2008, 28(7): 1772 - 1775.

[5] 段俊锋. 一种基于 PspCidTable 的进程检测方法[J]. 黑客防线, 2007(10): 107 - 109.

[6] MARK E R, DAVID A S. Microsoft Windows internals: Microsoft Windows Server 2003, Windows XP, and Windows 2000[M]. 4th ed. Seattle: Microsoft Corporation Press, 2007.

[7] GREG H, JAMES B. Rootkit: Subverting the Windows kernel[M]. Boston: Addison Wesley, 2005.

[8] Detect hidden process[EB/OL]. [2006 - 04 - 20]. http://wasm.ru/article.php?article=hiddndt.

[9] NAGAR R. Windows NT file system internals [M]. New York: O'Reilly, 2007.

[10] 王建华,张焕生,侯丽坤,等. Windows 核心编程[M]. 北京: 机械工业出版社, 2006.

[11] HackerDefender[EB/OL]. [2007 - 11 - 25]. http://hxdef.org/.

[12] 葛军,黄土平. 灰鸽子远程控制系列[EB/OL]. [2005 - 06 - 11]. http://www.huigezi.net/index.asp.

[13] Intox. Agony ring0 rootKit [EB/OL]. [2006 - 01 - 01]. http://www.undergroundkonnekt.net.

[14] Vanquish[EB/OL]. [2005 - 09 - 18]. http://www.rootkit.com/project.php?id=9.

[15] HOGLUND G. Ntrootkit [EB/OL]. [2005 - 12 - 19]. http://www.rootkit.com/project.php?id=11.

[16] Fuzen_op, FU Rootkit[EB/OL]. [2007 - 11 - 23]. http://www.rootkit.com/project.php?id=12.