Windows 平台下 Rootkit 进程检测

张登银,陈召国
(南京邮电大学 计算机学院,江苏 南京 210003)

摘 要:Rootkit 是能够持久或可靠地存在于计算机系统上的一组程序或代码。为了达到无法检测的目的,Rootkit 必须使用进程隐藏技术。Rootkit 进程隐藏技术是一种以秘密方式在系统后台运行并窃取用户信息的技术。通过分析 Windows 平台下 Rootkit 进程隐藏技术的原理,研究了应用层和内核层两种模式下的 Rootkit 进程隐藏技术。针对 Rootkit 进程隐藏技术的特点,开发了一个基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测平台。系统测试表明,本平台能够检测出当前绝大部分主流 Rootkit 技术实现的隐藏进程,在实际应用中达到了较好的效果。

关键词:Rootkit;进程隐藏与检测;句柄表;内核
中图分类号:TP316.7 **文献标识码**:A **文章编号**:1673-629X(2011)07-0141-04

Rootkit Process Detection under Windows Platform

ZHANG Deng-yin, CHEN Zhao-guo
(College of Computer, Nanjing University of Posts & Telecommunications, Nanjing 210003, China)

Abstract: Rootkit is a set of procedures or code that is able to exist in a lasting or reliable computer system. In order not to detect the hidden process, Rootkit must use the technology of process hiding. The Rootkit process hiding is a kind of technology to run under system platform and steal user information as a secret way. By analyzing the principle of Rootkit process hiding under Windows platform, makes a comprehensive study about the technology of Rootkit process hiding. According to the characteristics of the Rootkit process hiding, develops a trinity and cross-mapping platform based on handle table for detecting the Rootkit hidden process. The experiment indicates that the platform can detect most of the current Rootkit hidden process and takes good effect in practice.

Key words: Rootkit; process hiding and detection; handle table; kernel object

0 引 言

Rootkit 是一种非法获取访问能力并隐匿踪迹的后门技术。攻击者一旦突破计算机系统后,往往在已入侵的计算机系统中安装 Rootkit,使得攻击者能够保持访问计算机上具有最高权限的用户"Root"。为了避开系统检测,Rootkit 必须尽可能地隐藏自身进程,通常采用 Rootkit 进程隐藏技术[1,2]。根据对操作系统入侵的层次,Rootkit 进程隐藏技术可分为应用层 Rootkit 和内核层 Rootkit 两种。应用层 Rootkit 进程隐藏技术,一般通过修改或者替换系统现有的二进制程序和运行时的内存补丁技术来达到隐藏自身的目的;内核层 Rootkit 进程隐藏技术,则通过修改系统底层内核来隐藏进程信息。Rootkit 进程隐藏技术可以使恶意代码在目标主机中长期潜伏,窃取数据、监视用户和持续控制主机而不被发觉,从而造成极大危害。因此 Rootkit 进程隐藏技术已经成为信息安全领域最大的挑战之一,如何检测 Rootkit 隐藏进程对于系统安全具有重大意义。

收稿日期:2010-12-24;**修回日期**:2011-02-15
基金项目:国家自然科学基金项目(61071093);国家 863 计划项目(2009AA701202);Swedish Research Links Programme (348-2008-6212);留学回国人员项目(NJ209002)
作者简介:张登银(1964-),男,江苏靖江人,博士,研究员,博士生导师,CCF 会员,研究方向为信息安全、信号与信息处理、IP 网络技术;陈召国,硕士研究生,研究方向为基于 IP 的下一代通信与系统安全。

1 Rootkit 进程隐藏技术分析

1.1 应用层进程隐藏

(1) DLL 嵌入技术。

目前,在 Windows 平台下,人们通常将被隐藏的程序以线程方式嫁接于其它远程进程之中来实现进程隐藏,一般使用 CreateRemoteThread 函数来实现[3]。首先,通过进程 PID 获得想要注入代码的进程句柄,然后把恶意代码封装在动态链接库(DLL)中,将 DLL 路径名复制到远程进程的内存空间,再创建远程线程,并通过远程线程技术调用所设计的 DLL 文件。

(2) API 钩子技术。

设计 API 钩子替换函数自入口点开始的一些字

节,这种方法相当简单[4]。首先通过 GetProcAddress 来获取 DLL 导出的 API 函数的地址,然后把原来的 API 函数地址替换为设计的 JMP 跳转指令,再使用远程线程技术把设计的 DLL 注入到目标进程,由此来实现 API 函数的挂钩。

(3) IAT 钩子技术。

PE 文件结构中含有. idata 和. edata 两个重要的节,分别包含了导入的相关信息和导入函数的地址,其中. idata 中存放有输入地址表(IAT)[5]。导入地址表(IAT)的地址可通 ImageDirectoryEnrtryToData 返回,并在其中查找要修改的导入函数所在的 DLL,然后遍历"User32. dll"中导入的符号,查找并修改要挂钩函数的地址,转到设定的目标函数的地址,再使用远程线程技术把设计的 DLL 注入到目标进程,由此实现 IAT 钩子技术。

1.2 内核层进程隐藏

(1) SSDT 技术。

挂钩系统服务描述表(SSDT)技术[6,7],通过钩住 ZwQuerySystemInformation 函数隐藏进程名。Windows 操作系统通过 ZwQuerySystemInformation 函数可以查询不同类型的信息。例如,Taskmgr. exe 通过该函数可以获取系统上的进程列表。返回的信息类型取决于请求的 SystemInformationClass。通过把 SystemInformationClass 设置为 5,可以得到一组进程链表,然后钩子就可以调用原始的函数查找并过滤要隐藏的进程名。在隐藏进程的过程中要用到缓冲区的结构,缓冲区中的信息包括_SYSTEM_PROCESS 结构及其相应的_SYSTEM_THREADS 结构。_SYSTEM_PROCESS 结构中重要的一项是包含进程名的 UNICODE_STRING,还有两个 LARGE_INTEGERS 包含了进程所使用的用户和内核时间。在隐藏进程时,Rootkit 要把隐藏进程使用的时间放到系统进程(Idle)中,从而使总的时间加起来达到 CPU 时间的 100%。

(2) DKOM 技术。

在 Windows 下,系统会给每一个进程分配一个 executive process(EPROCESS),该结构包含本进程的所有线程信息。直接内核对象操作(DKOM)技术[8,9],就是通过遍历在每个进程的 EPROCESS 结构中引用的一个双向链表获得 Windows 操作系统的活动进程列表。EPROCESS 结构包含一个具有指针成员 FLINK 和 BLINK 的 LIST_ENTRY 结构,这两个指针分别指向当前进程描述符的前方和后方进程。

隐藏进程首先需要在内存中找到 EPROCESS 结构。EPROCESS 结构几乎在操作系统的每个发布版本中都会变化,但通过 PsGetCurrentProcess 函数始终能够找到当前运行进程的指针,从而找到它的 EPROCESS。然后遍历 EPROCESS 结构的双向链表,一旦发现了要隐藏进程的 EPROCESS,必须修改它的前方和后方 EPROCESS 块的 FLINK 和 BLINK 指针值,将要隐藏的进程从双向链表中断开。

应用层 Rootkit 通过修改工作在用户态的可执行文件和动态链接库达到隐藏进程的目的。可以通过分析 PE 文件结构和调用内核函数的方法发现隐藏的钩子,相对于内核层 Rootkit 检测起来相对简单,在实际应用中应用不多。内核层 Rootkit 一般是以驱动程序的形式载入内核,通过对内核数据结构的修改实现隐藏,它需要研究人员了解系统的整体架构以及内核细节。因此在内核层隐藏进程的难度较大,如何发掘未知的可以隐藏进程的区域是难点之一。

2 基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测方案

2.1 设计思想

三位一体交叉映射检测是通过获取内核层三个点的进程映像并和用户层的进程映像进行交叉映射比较,判断应用层和内核层的进程映像是否一样。如是,则正常;否则,则认为遭到 Rootkit 的攻击。

目前在内核层获取进程映像的方法有挂钩 SwapContext 技术、挂钩 SSDT 技术和 DKOM 技术等。在 Windows 系统中,处理器的调度对象是线程,当系统 CPU 时钟中断触发时,会产生线程调度请求。可以说,CPU 的线程切换离不开 SwapContext 函数。Rootkit 所执行线程的也会通过 SwapContext 函数来切换使之被 CPU 处理,因此有人提出了设计自己替换线程的调度函数从而实现检测隐藏进程的目的。但是这种检测方法会占用很大的资源,毕竟 CPU 的线程切换非常频繁[10]。而目前对于挂钩 SSDT 技术和 DKOM 技术的研究比较多,已经不适宜作为检测的方法。研究发现在 Windows 系统中,基本上每一个进程都是以双链指针连接在一起组成一个句柄表,也就是说每一个进程都有指向它的前一个和后一个进程的句柄指针。一旦得到句柄指针,就可以遍历得到一组进程映象。笔者通过研究发现了获取进程句柄表的三种途径,考虑到检测的准确性,采用了统一三种句柄表三位一体交叉映射的设计思想。

2.2 方案设计

本平台采用遍历句柄表技术得到内核层进程映像,然后在用户层调用系统进程快照交叉映射得到隐藏进程,据此文中开发了一个基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测平台。在内核层中,遍历句柄表技术采用遍历当前进程的句柄表、遍历 csrss. exe 进程的句柄指针和遍历 PspCidTable 的句柄

表三种方法枚举进程。

在 Windows 平台下,除了 Idle 进程外,每一个进程都有一个句柄表[11](_HANDLE_TABLE),通过 WinDbg 调试一个进程的 EPROCESS 结构发现,EPROCESS 的地址加上 0x0c4 就可以得到当前进程的句柄表;进一步调试内核可以发现句柄表是一个_HANDLE_TABLE 结构体,在 0x01c 的位置上有一个 HandleTableList 的双链指针,如图 1 所示。通过枚举发现,该双链可以枚举出除 Idle 以外的所有进程信息。因此可以通过遍历当前进程 EPROCESS 中的句柄表双链指针得到一组进程的映像。

```
ntdll!_HANDLE_TABLE
   +0x000 TableCode          : Uint4B
   +0x004 QuotaProcess       : Ptr32 _EPROCESS
   +0x008 UniqueProcessId    : Ptr32 Void
   +0x00c HandleTableLock    : [4] _EX_PUSH_LOCK
   +0x01c HandleTableList    : _LIST_ENTRY
   +0x024 HandleContentionEvent : _EX_PUSH_LOCK
   +0x028 DebugInfo          : Ptr32 _HANDLE_TRACE_DEBUG_INFO
   +0x02c ExtraInfoPages     : Int4B
   +0x030 FirstFree          : Uint4B
   +0x034 LastFree           : Uint4B
   +0x038 NextHandleNeedingPool : Uint4B
   +0x03c HandleCount        : Int4B
   +0x040 Flags              : Uint4B
   +0x040 StrictFIFO         : Pos 0, 1 Bit
```

图 1　_HANDLE_TABLE 结构

csrss. exe 进程是 Windows 平台下子进程的服务进程,除 Idle、system、smss 以及 csrss 以外的所有进程信息都保存在其中[12]。每个新创建的进程都要到它那里注册方能运行。当进程结束时,也会给它发送一个销毁消息,从而确保它保存的是实时运行的进程信息。通过遍历模块列表找到 CSRSRV. DLL 的基址,调试内核可以发现里面有一个未导出的指针 CsrRootProcess,该指针指向一个名为 CSR_PROCESS 的结构体,该结构体包含了指针指向的进程信息。结构体中的 ListLink 将系统内的活动进程串接起来形成链表。通过遍历整个链表,可以得到一组当前活动进程的映像。

PspCidTable 是一个没有被内核 ntoskrnl. exe 导出的句柄表结构[13]。对于每个进程来说,它们的句柄表是私有的并且以一个双链连接起来的,而 PspCidTable 是一个独立的句柄,不属于任何一个进程,存放的对象是系统中所有的进程和线程指针。该句柄表在 XP 和 2003 下面采用动态扩展表,即先保存一层表结构,如果线程和进程对象数过多,就会启用两层表或者三层表结构。目前在内核层得到 PspCidTable 地址的方法是通过内核未导出的 PsLookupProcessByProcessId 函数。该函数调用时会引用 PspCidTable 的句柄表,可通过暴力搜索该函数的前几十个字节找到 PspCidTalbe 变量,然后获取 PspCidTable 的地址。得到地址后,遍历 PspCidTable 可采用 PsLookupProcessByProcessId 函数和 EnumHandleProcedure 函数直接枚举进程,缺点是两个函数目前已经公开,很容易被钩住,从而影响检测的准确性。文中通过设计自己的函数 EnumProcess 实现枚举进程,在这里把 PspCidTable 的地址和对应的指针传递给 EnumProcess 函数,此方法可以有效地预防 Rootkit 钩子,从而保证检测的准确性。最后遍历句柄指针就可以得到内核层的一组进程映像。

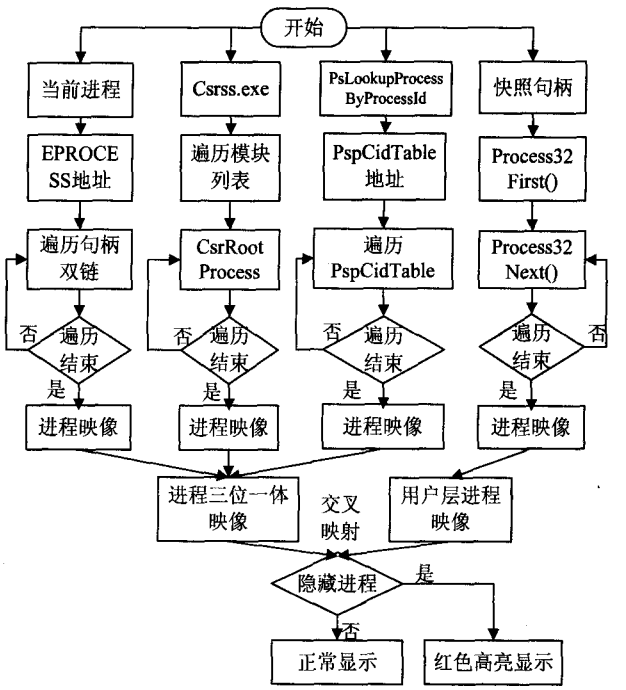

图 2　隐藏进程检测流程

基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测平台工作流程如图 2 所示,在内核层通过获取三种句柄表采集进程映像并运用三位一体分析的方法,以确保内核层得到的是整个系统的所有进程映像。在用户模式下,采用系统快照的方法得到应用层的进程映像,然后交叉映射,使得隐藏的进程红色高亮显示,便于发现隐藏的进程。

3　实验分析

基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测平台是用 C 语言编写并以 MFC 集成的形式编译实现,可视化界面良好。实验环境是 Windows XP SP2,具体实验步骤如下:首先在 Windows 中运行基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测平台;其次利用 SSDT 技术隐藏“SogouExplorer. exe”进程;再次打开任务管理器确定“SogouExplorer. exe”进程已经隐藏;最后点击“进程检测”选项,观察隐藏进程。

通过上述操作,任务管理器中的进程列表如图 3 所示,任务管理器下面没有显示“SogouExplorer. exe”隐藏进程,而在基于句柄表三位一体交叉映射的 Rootkit 隐藏进程检测平台下,可以发现“SogouExplorer. exe”进程已被红色高亮显示,如图 4 所示。实验结果表明,文中设计的基于句柄表三位一体交叉映射的

Rootkit 隐藏进程检测平台可以检测出 Rootkit 隐藏进程。通过进一步测试可以发现，对于钩住 SSDT 表、直接内核对象技术以及 FU_Rootkit 等主流隐藏工具隐藏的进程，本平台都可以一一检测，具有较好的应用效果。

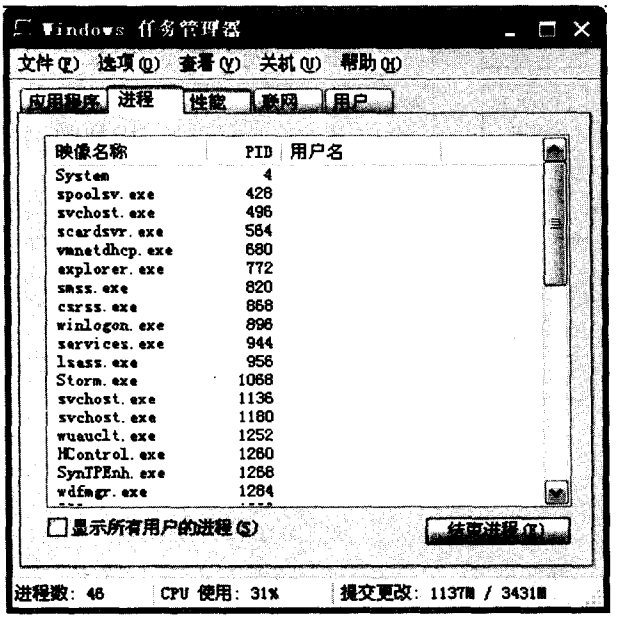

图 3 任务管理器中的进程列表

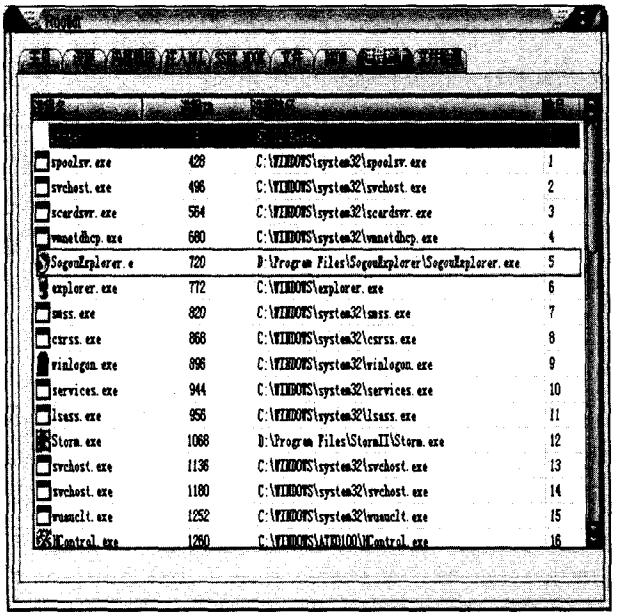

图 4 Rootkit 隐藏进程检测平台的检测结果

4 结束语

Rootkit 进程的隐藏与检测已成为当前网络攻防对抗的重要手段，两种技术在斗争中不断推动着信息安全技术向前发展。通过研究多种 Rootkit 进程隐藏技术的特点，提出了一种基于句柄表三位一体交叉映射的Rootkit隐藏进程检测方案。系统测试表明，本方案能很好地检测通过修改内核层数据结构和进程链表等绝大部分主流 Rootkit 技术实现的进程隐藏，具有较佳的应用价值。

现在的 Rootkit 隐藏技术已经不满足于修改进程句柄和内核层数据结构的层面上，未来的 Rootkit 进程隐藏技术有可能脱离 Windows 系统独立存在于某些硬件当中[14]，例如安装到 BIOS 或处理器的微码中，而依附于操作系统的上层 Rootkit 检测平台则会对这种硬件 Rootkit 毫无察觉。

相应的发掘不依赖于操作系统检测 Rootkit 隐藏进程是下一步的研究方向。

参考文献：

[1] Heasman J. Rootkit threats[J]. Network Security, 2006, 1: 18-19.

[2] Josse S. Rootkit detection from outside the Matrix[J]. Journal in Computer Virology, 2007, 3(2): 113-123.

[3] 何 志，范明钰，罗彬杰. 基于远程线程注入的进程隐藏技术研究[J]. 计算机应用，2008，28(6)：92-94.

[4] 龚 友，范明钰，王光卫. 基于调用门的进程隐藏技术[J]. 计算机应用，2009，29(6)：86-88.

[5] 龚 广，李舟军，忽朝俭，等. Windows 内核级 Rootkits 隐藏技术的研究[J]. 计算机科学，2010，37(4)：59-62.

[6] Levine, Grizzard, Owen. Detecting and Categorizing Kernel-Level Rootkits to Aid Future Detection [J]. IEEE Security & Privacy, 2006, 4(1): 24-32.

[7] 龙 海，郝东白，黄 皓. Windows 下系统服务 Rootkits 的检测与恢复[J]. 计算机工程与设计，2008，29(7)：1612-1615.

[8] Doug W. Methods for Detecting Kernel Rootkits[D]. Louisville: University of Louisville, 2007.

[9] Russinovich M, Solomon D, Ionescu A. Windows Internals [M]. 5th ed. USA: Microsoft Press, 2008.

[10] Kimmo. Detecting Hidden Process by Hooking the SwapContextFunction[EB/OL]. [2004-08]. https://www.Rootkit.com/newsread.php? newsid=170.

[11] Kong J. DESIGNING BSD ROOTKITS[M]. San Francisco: No STARCH PRESS, 2007.

[12] Jurczyk M, Hispasec G C. Windows XP SP3 CSRSS Process Handling Local Privilege Escalation[EB/OL]. [2008-12]. http://vexillium.org/dl.php? HISPASEC_CSRSS_Priv_Esc-al.pdf.

[13] Jimhokin. Detection of Hidden Process[EB/OL]. [2006-01]. www.rootkit.com/newsread_print.php? n-ewsid=434.

[14] David F M, Chan D E, Carlyle J C, et al. Cloaker: Hardware Supported Rootkit Concealment[C]//IEEE Symposium on Security and Privacy. [s.l.]: [s.n.], 2008.