Windows 内核级 Rootkits 隐藏技术的研究

龚 广[1] 李舟军[1] 忽朝俭[1] 邹蕴珂[1,2] 李智鹏[1]
(北京航空航天大学计算机学院 北京 100191)[1] (海军装备研究院 北京 100161)[2]

摘 要 随着 Rootkits 技术在信息安全领域越来越受到重视,各种 Anti-rootkits 新技术不断出现。在各种 Anti-rootkits 工具的围剿下,常规的 Rootkits 隐藏技术难以遁形。在系统分析和深入研究传统内核级 Rootkits 隐藏技术的基础上,提出了一个集驱动模块整体移位、内核线程注入、IRP 深度内联 Hook 3 种技术为一体的 Rootkits 隐藏技术体系。实验结果显示,基于该隐藏技术体系所实现的 Rootkits 能够很好地躲避专业的 Anti-rootkits 工具(如 Rootkit Unhooker 和冰刃)的检测,从而充分表明了这种三位一体的 Rootkits 隐藏技术体系的有效性。

关键词 Rootkits,Anti-rootkits,驱动模块整体移位,内核线程注入,IRP 深度内联 Hook

中图法分类号 TP393 **文献标识码** A

Research on Stealth Technology of Windows Kernel-level Rootkits

GONG Guang[1] LI Zhou-jun[1] HU Chao-jian[1] ZOU Yun-ke[1,2] LI Zhi-peng[1]
(School of Computer, Beihang University, Beijing 100191, China)[1]
(Navy Equipment Academe, Bejing 100161, China)[2]

Abstract With more and more attention being paid to the Rootkits technology in the fields of cyber-security, various new Anti-rootkits technologies have emerged continually. Under the detection of various Anti-rootkits tools, the conventional Rootkits stealth technology is difficulty to play its role. Based on systematic analysis and research of traditional kernel-level Rootkits stealth technology, this paper presented a three-in-one rootkits stealth technical architecture on the basis of driver module integral transposition, kernel threads injection and IRP inline Hook in depth. Experimental results show that the Rootkits based on this stealth architecture can well bypass the detection of some well-known Anti-rootkits tools (such as Rootkit Unhooker and IceSword), which fully demonstrates the effectiveness of this three-in-one Rootkits stealth technical architecture.

Keywords Rootkits, Anti-rootkits, Driver module integral transposition, Kernel threads injection, IRP inline Hook in depth

Rootkits 源自 Unix 系统,最早出现在 1994 年 2 月的一篇安全咨询报告中[1]。1999 年 Greg Hoglund 开发了针对 Windows 的 Rootkits(NT Rootkit),此后 Windows Rootkits 技术得到了迅速发展。Windows 系统由于其普及性成为 Rootkits 攻击的主要目标。2008 年 8 月卡巴斯基实验室发表技术类文章 Rootkit evolution[2],图 1 是该文中关于 Rootkits 发展史上重要 Rootkits 出现时间的事件时间表,曲线纵轴表示 Rootkits 产品出现的数量。由图可以看出,到 2008 年,安全领域对 Rootkits 的研究热度达到了一个新的顶峰。

Rootkits 是一种技术,病毒在使用,杀毒软件也在使用。对 Rootkits 隐藏技术的研究,主要有以下 3 方面的意义:首先,Rootkits 隐藏技术可用于计算机取证人员收集计算机犯罪证据;其次,只有深入研究 Rootkit 技术,才能检测出使用了 Rootkits 技术的恶意程序;在网络安全的对抗中,只有做到知己知彼,方能百战百胜,只有掌握了 Rootkits 这种技术,才能在与国内外敌对势力的信息对抗中占据优势。

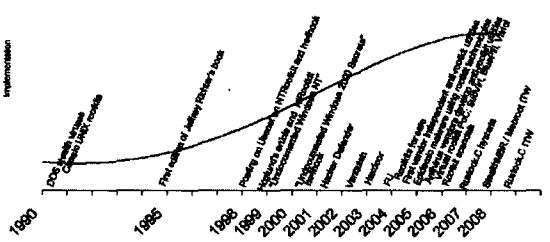

图 1 事件时间表

1 Rootkits 相关概念

Rootkits 是能够持久并且无法检测地存在于计算机上的一组程序和代码,攻击者利用它能够对计算机系统保持最高

到稿日期:2009-05-05 返修日期:2009-07-17 本文受国家自然科学基金(60473057,90604007,90718017)和教育部博士点基金(20070006055)资助。

龚 广(1985—),男,硕士生,主要研究方向为网络与信息安全,E-mail:higongguang@163.com;李舟军(1963—),男,教授,主要研究方向为高可信软件技术、网络与信息安全技术、智能信息处理技术;忽朝俭(1982—),男,博士生,主要研究方向为网络与信息安全;邹蕴珂(1982—),女,硕士生,主要研究方向为网络与信息安全;李智鹏(1985—),男,硕士生,主要研究方向为网络与信息安全。

权限的访问，以便进一步收集信息。攻击者并不能直接通过Rootkits获取系统管理员权限，但只要获得权限，种植了Rootkits，就能维护一个后门，允许攻击者一直以管理员权限控制系统，并且通过隐藏文件、进程、注册表项、端口等来隐藏攻击行为，逃避用户和安全软件的检测。

Intel X86微芯片系列使用环（ring）的概念来实施访问控制。环有4个级别：环0的权限最高，环3的权限最低[3,4]。Windows操作系统只使用了环0和环3，所有内核代码都在环0级别上运行。根据对操作系统的入侵层次，Rootkits分为用户级Rootkits和内核级Rootkits两种。用户级Rootkits依赖于对用户层程序的修改，在环3级别上执行，而内核级Rootkits会修改操作系统内核，在环0级别上执行。用户级Rootkits相对更容易实现，通用性好，而内核级Rootkits则更隐蔽，更难检测。本文只讨论内核级Rootkits隐藏技术。

2 传统隐藏技术

Rootkits作为一种强大的后门工具，隐藏性是其首要特征。它被成功植入后，利用各种手段来隐藏踪迹，依据隐藏手段的不同，当前的内核级Rootkits隐藏技术主要分为两类：DKOM和修改内核执行路径[5,6]。修改内核执行路径的实现依赖两种技术：HOOK技术和过滤驱动技术。

2.1 修改内核执行路径

2.1.1 HOOK技术

（1）IAT HOOK

IAT(Import Address Table)导入地址表是PE文件里的一个表项。常见的PE文件包括EXE，DLL，SYS文件。内核层的IAT HOOK主要作用于SYS文件。SYS文件通常要调用ntoskrnl.exe中的导出函数，在调用了ntoskrnl.exe中的某一函数后，必须导入该函数的地址。导入函数的名称存在于PE结构IMAGE_IMPORT_DESCRIPTOR中，当操作系统将模块加载到内存时，会分析IMAGE_IMPORT_DESCRIPTOR结构，在内存中定位每个导入函数，并使用函数的实际地址来重写一个IMAGE_IMPORT_DESCRIPTOR结构（要学习Windows PE格式中各种数据结构的更多知识，参见Matt Pietrek的文章[7]）。

当Rootkits被植入内核后，就可以分析加载模块所对应的PE格式，并将IAT中目标函数的地址替换为钩子函数的地址。然后，当调用目标函数时，就会执行钩子函数而不是原始函数。对这种技术的最新研究是可以使用其来隐藏SSDT钩子[8,9]。这种方法功能强大且实现简单，但通过将ntoskrnl.exe的导出表与修改后的模块导入地址表做比较就可以很容易地检测出这种钩子。

（2）IDT/SYSENTER HOOK

中断描述符表寄存器[10]（Interrupt Descriptor Table Register，IDTR）存储了IDT在内存中的基地址。IDT用于查找处理中断所用的函数。计算机中大量底层功能都使用了中断。例如，敲击键盘就会产生中断信号。IDT是一个由256项组成的数组，每个中断对应一项。因此每个CPU最多处理256个中断。当中断发生时，可以从中断指令或可编程中断控制器中获取中断编号，然后通过中断表寻找要调用的中断处理函数。该函数称为中断服务例程（Interrupt Service Routine，ISR）[11]。通过修改IDTR的内容，Rootkit可以创建一个新的中断表并替换CPU原来的中断表，从而控制中断处理行为。这种HOOK可以通过读取中断寄存器里的地址，判断其是否在内核模块地址范围内来进行检测；另一种IDT HOOK不是替换整个中断表，而是修改中断表的一些特殊项。在Windows 2000系统中，通过中断描述符表进行系统调用。Rootkits通过修改IDT的第0x2e项来拦截系统调用，这种方法的缺点是仅能简单地阻止请求，不能修改返回数据。Intel x86平台上的Windows XP系统用SYSENTER指令取代中断，使系统陷入系统服务调用程序，相应产生了新的HOOK方法，即把IA32_SYSENTER_EIP寄存器的值修改为要运行的Rootkits代码的地址。要查找出系统中的此种HOOK比较困难，因为并不知道大多数中断的模块范围。但可以确定的是INT 2E处理程序，它应该指向内核的ntoskrnl.exe，而此项是Rootkits最经常修改的。

（3）SSDT HOOK[12,13]

系统服务调度表（System Service Dispatch Table）中存放了所有系统服务函数的入口地址。当程序使用INT 2E或SYSENTER指令后，执行系统调用会导致在内核中调用KiSystemService（系统服务调度函数），该函数通过查找SSDT来调用相应的系统服务。因此，Rootkits可以将SSDT中的系统服务地址替换为自己代码的地址。这样，当调用系统服务时，实际运行的是Rootkits代码。例如，ZwQuerySystemInformation是系统枚举进程时须调用的系统服务，通过挂钩它可实现进程的隐藏。SSDT HOOK是最方便也是最常见的隐藏办法，各种Anti-Rootkits工具对SSDT修改的检测都非常严格。最新的检测办法是通过读取磁盘上的ntoskrnl.exe文件来得到一个干净的SSDT的拷贝，然后将其与内存中的SSDT做比较。这种检测方法让简单的SSDT HOOK无所遁形[14]。

（4）IRP HOOK

在内核中，有一种数据结构叫做IRP（I/O Request Package），即输入输出请求包。上层应用程序与底层驱动程序通信时，应用程序会发出I/O请求。操作系统将I/O请求转化成相应的IRP数据[15]。内核中的IRP派遣函数会对各种不同类型的IRP进行处理。在驱动安装的时候，会初始一个IRP派遣函数的指针表，这个指针表包含的就是IRP处理函数的地址。通过修改相应IRP的某类派遣函数指针（如图2所示），就可以将对此类IRP的相应处理截获到Rootkits中来，从而在Rootkits中对其处理逻辑进行修改。对此种HOOK的检测主要是扫描模块的IRP派遣函数表，看其是否在驱动的内存地址范围内[16]。

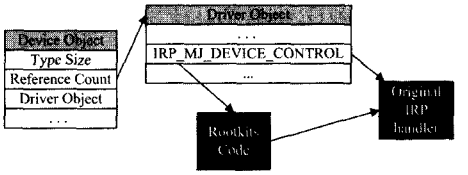

图2 IRP HOOK

（5）INLINE HOOK

INLINE HOOK即内联HOOK。在实现内联函数钩子时，Rootkits实际上重写了目标代码的字节。实现内联函数钩子时通常会保存钩子要重写的目标函数的多个起始字节和起始地址。保存了原始字节后，常常在目标函数的前几个字

节放置一个 JMP 指令，该跳转将程序执行流程转向 Rootkits 钩子。然后该钩子函数可以通过保存的内容跳转到原始函数。通过这种方法，原始函数将执行的控制权交给 Rootkits 钩子，此时钩子能够更改由原始函数返回的数据（如图 3 所示）。当前的 Anti-Rootkits 工具对 INLINE HOOK 检测主要采取对函数的前几个字节做完整性检测的方法，而对深度的内联 HOOK，则检测难度很大[17]。

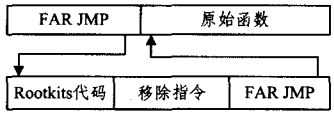

图 3　INLINE HOOK

2.1.2　过滤驱动技术

Windows 驱动模型假定硬件设备可以有多个驱动程序，每个驱动程序都有自己管理设备的方法。WDM 根据设备对象堆栈来完成驱动程序的分层，对驱动程序进行分层时，会将驱动程序注册到一个链中。如果向链接起来的驱动程序发出 I/O 请求，就会创建一个 IRP 并将其传递给该链中的所有驱动程序[18]。如果向这个链中插入一层新的过滤驱动程序，它不仅可以截获数据，也可以在传递数据之前对其进行修改。通过这种方式几乎可以截获系统上的所有设备，过滤许多重要的信息，例如键盘驱动的信息，文件系统驱动的信息，网络驱动的信息等。过滤驱动方式提供了一种相当简便而又底层的修改数据方式，这正是 Rootkit 技术所需要的。这种技术的缺点主要在于，要插入一层新的过滤驱动，就必须创建一个新的设备，并将其挂载到设备栈中的某一下层设备上，这导致这一下层设备的设备对象中将有一个指针指向我们创建的新设备对象。通过遍历设备栈可以找出异常的设备，然后根据异常设备对象可以找出其驱动对象，这样加载的 Rootkits 就无法隐藏了。

2.2　DKOM[19]

操作系统通过创建相关内核对象来登记和审计系统资源，采用 DKOM(Direct Kernel Object Manipulation)的 Rootkits 则通过修改这些内核对象来隐藏特定资源。Windows 系统经常要创建、打开和操作各种各样的内核对象，比如存取符号对象、事件对象、文件对象、文件映射对象、I/O 完成端口对象、管道对象、进程对象等。每个内核对象只是内核分配的一个内存块，并且只能由内核访问，应用程序无法在内存中找到内核对象的数据结构，也就不能直接修改它们的内容。系统提供的很多 API 都是通过查询相应内核对象，从而返回相应结果的。直接修改 API 访问的内核对象会影响 API 的返回结果，实现某些隐藏功能就是 DKOM 技术的实质。

3　三位一体的新 Rootkits 隐藏体系

Rootkits 要隐藏得很深，就应该尽量减少对系统的改动。在 Rootkits 检测方面，Anit-Rootkits 在进程与注册表的检测上花了很大的功夫。其实，实现一个 Rootkits 完全可以做到不添加新进程，不改动注册表项，这两项的隐藏自然也不用操心。但 Rootkits 要植入内核，就必须加载驱动模块，驱动模块的隐藏是必须要做的。在我们的隐藏体系中，使用驱动模块整体移位的方法来隐藏驱动模块。Rootkits 在保证深度隐藏的前提下，为完成某些特定功能（键盘信息和磁盘信息的获取），就需要使用内核线程注入技术和 IRP 深度内联 HOOK 技术。

下面分别介绍这 3 种技术。

(1) 驱动模块整体移位

驱动装载后，会在很多位置留下踪迹，如注册表、驱动链表等。要消除这些踪迹，最好的办法就是将驱动卸载，但是其功能模块仍需有效。我们的做法是在系统内存中另外申请一块空间，将驱动模块整体拷贝到该空间，然后修改相应的系统回调函数指针，使其指向拷贝后的相应位置。这时要注意的是，拷贝后的模块代码中有很多指针指向最初的模块加载位置，这就需要对拷贝后的模块进行重定位。需要重定位的位置多而杂，幸好在 PE 文件格式中有一张重定位表[20]。依据这张表对拷贝后的代码进行重定位，然后我们的驱动就可以安全卸载了。

(2) 内核线程注入

一般来说，Rootkits 除隐藏自身外，还要完成某些特定的功能。Anti-Rootkits 工具对隐藏进程的检测方法越来越多，进程隐藏的难度也越来越大。启动新进程来实现这些特定的功能很容易被发现和检测出来。但现有的 Anti-Rootkits 工具对 Rootkits 的检测还没有细化到线程级别，相对于进程而言，系统中的线程更是多而杂，有的进程（如 system 进程）就有几十个线程。要去分析每个线程的作用更是难上加难，故 Anti-Rootkits 工具也很难将检测细化到这一级别。因此，可通过调用 PsCreateSystemThread 来创建内核线程，再与驱动模块整体移位相结合，将线程的入口点设为移位后驱动模块的相应位置，这样就可以创建不与任何内核模块直接关联的内核线程，在线程中就可以实现我们想要的某些特定功能。

(3) IRP 函数深度内联 HOOK

当前 Anti-Rootkits 工具对 HOOK 的检查格外严厉，一般的 IRP HOOK、内联 HOOK 都难逃其法眼。本文提出了一种叫 IRP 函数深度内联 HOOK 的方法：当想截获某些 IRP 并对其设置完成例程时，最直接的办法就是采用 IRP HOOK，这种方法确实很好用，然而很不幸的是，Anti-Rootkits 工具早已在这里重重设防。若直接进行 HOOK，它立马就可以检测出来，故我们不直接修改 IRP 指针的值，而是去搜索此 IRP 指针所指向的函数的代码。在此 IRP 函数中，一般会调用其他的系统函数，我们修改这条调用指令，同样可以截获到想要的 IRP，实现特定功能。

4　实验结果

为了验证本文提出的三位一体的 Rootkits 隐藏体系的有效性，实现了一个使用该体系的 Rootkits 原型。在该原型中实现了键盘信息和磁盘信息的获取。在我们的 Rootkits 运行后，使用专业的 Anti-Rootkits 工具 Rootkit Unhooker 和冰刃对其进行检测。结果发现，从驱动模块和钩子的检测中，均检测不出其踪迹。图 4 和图 5 是检测结果。

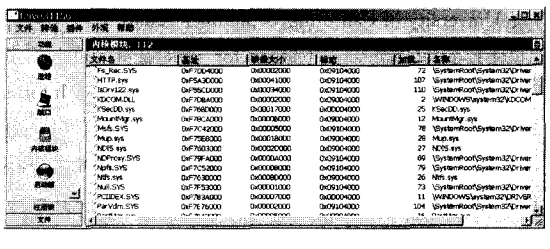

图 4　冰刃内核模块检测结果

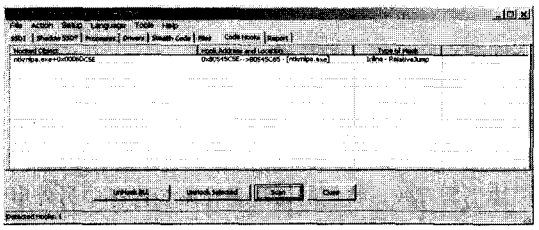

图 5 Rootkit Unhooker 代码钩子检测结果

结束语 网络攻防技术日新月异，隐藏技术也在攻防交替中不断发展和演变。本文在对传统 Rootkits 隐藏技术分类研究的基础上，提出了一个集驱动模块整体移位、内核线程注入、IRP 深度内联 HOOK 3 种技术为一体的 Rootkits 隐藏技术体系，并实现了一个基于该隐藏技术体系的 Rootkits 原型系统。实验结果显示，本 Rootkits 能够很好地躲避专业的 Anti-Rootkits 工具(如 Rootkit Unhooker 和冰刃)的检测，充分表明了这种三位一体的 Rootkits 隐藏体系的有效性。

Rootkits 技术是计算机系统安全的关键技术之一，其本身是中性的。在杀毒软件、防火墙、入侵检测系统等安全软件中，Rootkits 技术同样得到了广泛应用，发挥了强大的作用。随着网络安全的重要性日益显现，Rootkits 技术也必定会有更广阔的应用前景。

参 考 文 献

[1] CERT/CC. Ongoing Network Monitoring Attacks[EB/OL]. http://www.cert.org/advisories/CA-1994-01.html,1997-09-19

[2] Alisa S. Rootkit evolution [EB/OL]. http:// www. viruslist.com/en/analysis? pubid=204792016,2008-08-28

[3] Greg H, James B. Rootkits: Subverting the Windows kernel [M]. Addison Wesley Professional,2005:46-47

[4] Intel. Intel 6 4 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture, chapter 6[EB/OL]. http://download.intel.com/design/processor/manuals/253665.pdf,2006:5-11

[5] Rutkowski J K. Advanced Windows 2 0 0 0 Rootkit Detection [Z]. http://www.securitytechnet.com/resource/rsc-center/presentation/black/vegas03/bh-us-03-rutkowski.pdf. ,2003-07-01

[6] Levine J G. A Methodology for Detecting and Classifying Rootkit Exploits[D]. School of Electrical and Computer Eng. ,Georgia Inst. of Technology,2004

[7] Pietrek M. Peering Inside the PE: A Tour of the Win32 Portable Executable File Format[J]. Microsoft Systems Journal,1994

[8] xrayn. Hide your SSDT hooks[EB/OL]. http://www.rootkit.com/newsread.php? newsid=922,2008-11-08

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

[10] Barry B. Intel 微处理器[M]. 北京：机械工业出版社，2008：45-47

[11] Russinovich M, Solomon D. 深入解析 Windows 操作系统[M]. 潘爱民，译. 北京：电子工业出版社，2007：87-90

[12] Levine J G, Grizzard J B, Hutto P W, et al. A Methodology to Characterize Kernel Level Rootkit Exploits that Overwrite the System Call Table[C]// Proceedings of IEEE. SoutheastCon, IEEE,2004:25-31

[13] Levine J G, Grizzard J B, Owen H L. A Methodology to Detect and Characterize Kernel Level Rootkit Exploits Involving Redirection of the System Call[C]// Second IEEE International Information Assurance Workshop. 2004

[14] Levine J G, Grizzard J B, Owen H L. Detecting and categorizing kernel-level rootkits to aid future detection[M]. IEEE Security & Privacy,2006

[15] 张帆，史彩成. 驱动开发技术详解[M]. 北京：电子工业出版社，2008：186-187

[16] Molina D, Zimmerman M, Roberts G. Timely Rootkit Detection During Live Response [M]. Springer Boston,2008:139-148

[17] Hoglund G, Butler J. Rootkits: Subverting the Windows kernel [M]. Addison Wesley Professional,2005:290-291

[18] Microsoft Corporation. Windows 2000 驱动程序开发大全(第 1 卷)[M]. 北京：机械工业出版社，2001：27-36

[19] Greg H, James B. Rootkits: Subverting the Windows kernel [M]. Addison Wesley Professional,2005:173-183

[20] 罗云彬. Windows 环境下 32 位汇编语言程序设计(第 2 版)[M]. 北京：电子工业出版社，2007：667-670

(上接第 48 页)

[4] Werner-Allen G, Swieskowski P, Welsh M. MoteLab: A Wireless Sensor Network Testbed[C]// Proceedings of the 4th International Symposium on Information Processing in Sensor Networks. 2005

[5] Cerpa A, Busek N, Estrin D. Scale: A Tool for Simple Connectivity Assessment in Lossy Environments[R]. CENS Technical Report 0021. Los Angeles: Center for Embeded Networked Sensing, University of California, September 2003

[6] Emre E, et al. Kansei: a testbed for sensing at scale[C]// Proceedings of the 5th International Symposium on Information Processing in Sensor Networks. 2006

[7] Altare. Cyclone II Data Sheet CII5V1-3.3[EB/OL]. www.altera.com,2008

[8] Atmel, AT91 ARM Data Sheet[EB/OL]. www.atmel.com, 2007

[9] IEEE 802.3af. Data Terminal Equipment (DTE) Power via Media Dependant Interface(MDI)[M]. IEEE Computer Society, June 2003

[10] Dwelley D. Linear Technology Inc. New Power for Ethernet-The LTC4255 Delivers[J]. Linear Technology Magazine, Augst 2002

[11] 刘滨，王琦，刘丽丽. 嵌入式操作系统 FreeRTOS 的原理及实现[J]. 单片机与嵌入式系统应用，2005(07)

[12] Dunkels A. Design and Implementation of the LwIP TCP/IP Stack[J]. Swedish Institute of Computer Science, February 2001

[13] Labrosse J J. 嵌入式实时操作系统 μC/OS-Ⅱ(第 2 版)[M]. 邵贝贝，译. 北京：北京航空航天大学出版社，2003

[14] Chipcon. CC2420 Data Sheet[EB/OL]. www.ti.com,2004