第34卷 第12期 计 算 机 工 程 2008年6月
Vol.34 No.12 Computer Engineering June 2008
·安全技术· 文章编号：1000—3428(2008)12—152—02 文献标识码：A 中图分类号：TP393.08

Windows Rootkit 隐藏技术研究

杨 彦[1]，黄 皓[1,2]
(1. 南京大学计算机科学与技术系，南京 210093；2. 南京大学软件新技术国家重点实验室，南京 210093)

摘 要：Rootkit是恶意软件用于隐藏自身及其他特定资源和活动的程序集合。该文分析和研究现有的针对Windows系统的代表性Rookit隐藏技术，将其总结为2类：通过修改系统内核对象数据实现隐藏和通过修改程序执行路径实现隐藏。说明并比较了相应的技术原理，展望了Rootkit隐藏技术未来的发展趋势。

关键词：Windows Rootkit技术；Hook技术；系统内核；系统调用；中断描述符表

Research on Concealment Technology of Windows Rootkit

YANG Yan[1], HUANG Hao[1,2]
(1. Dept. of Computer Science and Technology, Nanjing University, Nanjing 210093;
2. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093)

【Abstract】Rootkit is a program set which malicious software uses to conceal itself and other specific resources and actions. This paper analyzes and researches on the concealment technologies which representative rootkits on Windows platform commonly use, and classifies them into two categories: modifying kernel object data structures and changing execution paths. The technical principles are described and compared in detail. The future development directions are discussed.

【Key words】Windows Rootkit technology; Hook technology; system kernel; system call; Interrupt Descriptor Table(IDT)

1 概述

随着网络技术的飞速发展，针对网络计算机的各种攻击层出不穷，网络安全问题显得由为突出。通过网络传播的恶意程序是攻击者实施网络攻击的利器，常见的有病毒、蠕虫和木马等。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，单纯以破坏系统可用性为目标的恶意程序逐渐消失，而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。它使恶意程序能在目标主机中长期潜伏，窃取信息而不被察觉，从而造成更大危害。Rootkit源自Unix系统，1999年Greg Hoglund开发了针对Windows的Rootkit (NT Rootkit)，此后Windows Rootkit技术得到了迅速发展，Windows系统由于其普及性成为了Rootkit攻击的主要目标。

2 Rootkit的相关概念

Rootkit是一个程序的集合，用于实现自身及系统中特定资源和活动的隐藏，破坏可信任计算机的完整性[1]。攻击者并不能直接通过Rootkit获取系统管理员权限，但只要获得权限，种植了Rootkit，它就能维护一个后门，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。从运行环境看，Rootkit可分为用户模式Rootkit和内核模式Rootkit。内核模式Rootkit工作于ring0，能直接干预所有用户和系统内核的内存空间，破坏能力强，不能被工作于用户模式的监控软件发现，但工作需要ring0权限，且兼容性较差。用户模式Rootkit工作于ring3，功能虽不及内核模式Rootkit强大，却具有轻便、通用性强的优点。从采用的隐藏手段看，Rootkit主要可归结为2类[2]：(1)通过修改程序执行路径，截获所有访问隐藏资源的请求；(2)直接修改操作系统用于监控系统资源的数据，称为DKOM技术。

3 DKOM隐藏技术

操作系统通过创建相关内核对象来登记和审计系统资源，DKOM(Direct Kernel Object Manipulation)Rootkit[3]则通过修改这些内核对象来隐藏特定资源。Fu Rootkit是DKOM Rootkit的代表，它通过修改活动进程列表实现进程隐藏。

Windows系统枚举进程使用的是活动进程列表PsActive ProcessList，它是一个双向链表，每个结点对应一个进程的EPROCESS数据结构，所有结点通过EPROCESS结构中的ActiveProcessLinks双向指针链在一起。要隐藏某个进程，只需修改对应EPROCESS的ActiveProcessLinks，将其从链表中摘除即可。由于系统执行线程调度使用的是其他的数据结构，因此这样的修改不会影响进程运行。

Windows 2000/XP提供了一种在用户模式下直接访问物理内存的方法，使得Rootkit无须安装附加的驱动程序就可以操作内核数据，具体做法是调用ZwOpenSection打开\Device\PhysicalMemory内存节对象，调用MapViewOfSection将目标物理内存页映射到进程的内存空间，访问映射后的内存。Rootkit首先获取进程的EPROCESS的虚拟地址，换算成物理地址，然后利用上述方法修改内存数据，把要隐藏的进程从活动进程列表上删除，实现隐藏。

基金项目：国家自然科学基金资助项目“操作系统的安全结构及其在单向网关上的应用”(60473093)
作者简介：杨 彦(1982－)，女，硕士研究生，主研方向：计算机安全；黄 皓，教授、博士生导师
收稿日期：2007-08-11 **E-mail：**yangyantj@hotmail.com

4 修改程序执行路径

4.1 Win32 程序执行路径

Windows 系统是一个分层的操作系统，为了保护内核，用户模式程序不能直接访问内核。要与操作系统交互，程序必须首先调用 Win32 子系统动态库(包括 User32.dll, Gdi32.dll, Advapi32.dll, Kernel32.dll 等)提供的 Win32 API，Win32 API 又去调用 Ntdll.dll 导出的、能为内核所理解的 Native API。

从用户模式进入内核模式，ntoskrnl.exe是Windows中最重要的内核文件之一，包括 2 个关键部分：(1)函数Executive，负责管理内核函数、数据结构、内核状态等；(2)更底层的部分称为“内核”，包括系统调用的具体代码。Ntdll.dll通过调用Executive来确定使用“内核”中的哪部分代码。在Windows 2000 中，Ntdll.dll通过触发 0x2e号中断陷入内核模式，调用系统服务调度程序；Windows XP中则用SYSENTER指令取代中断完成相同功能。系统服务调度程序查看系统服务调度表(System Service Dispatch Table, SSDT)，确定要调用的系统服务代码地址，从Executive转向底层内核，执行相应的系统调用[4]。Windows程序执行路径如图 1 所示。这一从上到下的程序执行路径涉及了很复杂的函数调用关系，Rootkit正是通过修改这一路径来插入自己的代码，从而破坏系统返回的结果，实现资源的隐藏。Hook是安全领域的重要技术，即拦截特定的函数，使得当此函数被调用时，实际运行的是钩子代码，作额外处理后再转回原函数。修改程序执行路径类的Rootkit大多基于Hook技术。

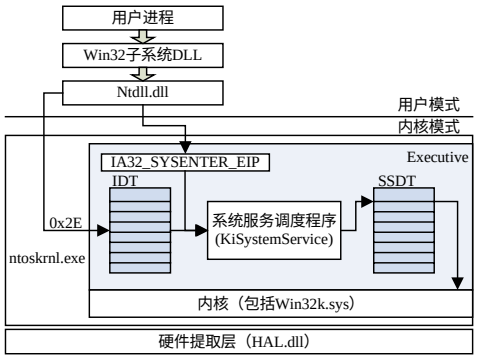

图 1 Windows 程序执行路径

4.2 隐藏的实现

4.2.1 用户模式下的 API Hook

在用户模式下，应用程序会调用 Win32 API 和 Native API，Rootkit 可以拦截这些 API，插入自己的代码。

一种常用方法是修改PE文件的导入函数表(Import Address Table, IAT)。IAT中记录着该文件运行中需调用的所有链接库，以及链接库中需调用的所有库函数。PE文件运行时，文件自身和IAT中记录的链接库都加载到内存，然后在库函数符号名和相应代码在当前进程空间的地址间建立映射，以后就通过IAT来调用这些函数。Rootkit采取的手段是在进程加载时修改IAT，将要Hook的API函数地址改为Rootkit代码的地址。这样，进程每次调用此函数时，其实执行的是Rootkit代码。另一种方法称作Inline Hook[5]，将被拦截函数的前几个字节内容修改为一个无条件跳转指令JMP，使程序的执行跳转到Rootkit代码。Rootkit进行额外处理后，执行事先保存下来的被改写部分的原指令，并跳转回原函数相应位置，这样，函数的原有功能就得到了完整的调用。原函数的返回结果转由Rookit函数处理，整个过程如图 2 所示。

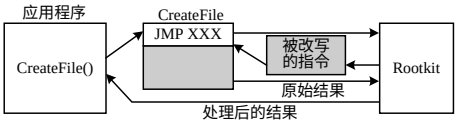

图 2 用户模式下的 Inline Hook

动态库陷阱方法直接用重写的 DLL 文件来替换系统中原有的 DLL，并将原 DLL 改名。重写的 DLL 的 IAT 和原 DLL 的 EAT(Export Address Table)之间存在一个一一映射。系统调用到需 Hook 的 API，就执行相应的 Rootkit 代码，否则转而调用原 DLL 中的 API。该方法的难点在于 Windows 文件保护机制(Windows File Protection, WFP)对系统中关键文件的完整性作了保护，当 WFP 发现这些文件被修改，会自动恢复原文件。然而通过改写 SFCDisable 注册表项和修改 Sfc.dll，Rootkit 可以骗过粗心的系统管理员，中止 WFP。

4.2.2 IDT Hooks 和 SYSENTER Hook

在 Windows 2000 系统中，通过中断描述符表进行系统调用。IDT 有 256 个入口，每个入口说明对应的中断处理程序的地址。Rootkit 通过修改 IDT 的 0x2e 入口来拦截系统调用，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(AMD 平台上的 Windows XP 使用 syscall 指令实现相同功能)，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 Rootkit 代码的地址。

4.2.3 SSDT Hook

系统服务调度表(SSDT)中存放了所有系统服务函数的入口地址。系统服务调度程序通过查找 SSDT 来调用相应的系统服务。因此，Rootkit 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 Rootkit 代码，由 Rootkit 代码调用真正的系统服务并对结果作相应处理。系统中存在 2 个 SSDT，位于文件 ntoskrnl.exe 中，分别由指针 KeServiceDescriptorTable 和 KeServiceDescriptor Table Shadow 导出，对应了 2 类不同的系统服务。前者对应的系统服务，功能代码在 ntoskrnl.exe 中实现，用户模式下的访问接口由 NTDLL.dll 提供，通常 kernel32.dll/advapi32.dll 导出的 API 最终调用的都是这类系统服务；后者对应的是 USER 和 GDI 系统服务，访问接口由 User32.dll/Gdi32.dll 提供，功能代码在 Win32k.sys 中实现。Rootkit 关注的主要是对进程、文件、注册表等的操作，对应的系统服务属于第 1 类，需修改 KeService Descriptor Table 导出的 SSDT。KeServiceDescriptorTable 指向一个 KSERVICE_TABLE_ DESCRIPTOR 结构，其中的 Service TableBase 指针指向真正的系统服务入口地址表。如要 Hook 某个系统服务，将该表中相应的代码入口地址修改为钩子函数的地址即可。例如，ZwQuerySystemInformation 是系统列举进程时须调用的系统服务，通过挂钩它来实现进程的隐藏。

4.2.4 中间层过滤驱动

Windows 的驱动程序模型采用分层栈式结构。对设备的访问请求从驱动栈顶开始，处理后再传递到下一层驱动。这就方便了系统的扩展，当需要新功能时，只要编写新的驱动程序，并指定其在驱动栈中的位置即可，无须修改原有系统。很多动态加密、病毒实时监控、防火墙等软件都是基于此项技术。Rootkit 通过安装文件过滤驱动，可以拦截到所有的

(下转第 156 页)