第35卷 第10期 计 算 机 工 程 2009年5月
Vol.35 No.10 Computer Engineering May 2009
·安全技术· 文章编号：1000—3428(2009)10—0118—03 文献标识码：A 中图分类号：TP393

Windows Rootkit 隐藏技术与综合检测方法

左黎明，蒋兆峰，汤鹏志
(华东交通大学基础科学学院，南昌 330013)

摘 要：针对Rootkit具有隐藏、通信、监听等功能但存在典型木马特征对计算机系统危害严重问题，分析近年来Windows操作系统下Rootkit中各种主流隐藏技术(包括DKOM和各种钩子)，指出当前单一检测方法的缺陷，提出综合性检测技术方案。实验结果表明，该方法达到较好的检测效果，可以对目前大多数Rootkit行为进行检测。

关键词：Rootkit技术；系统服务描述符表；隐藏

Concealing Technology of Windows Rootkit and Integrated Detection Method

ZUO Li-ming, JIANG Zhao-feng, TANG Peng-zhi
(School of Basic Science, East China Jiaotong University, Nanchang 330013)

【Abstract】 Rootkit is a program or a set of programs that an intruder uses to hide her presence on a computer system and to allow access to the computer system. This paper analyses the main concealing techniques of Windows Rootkits, including DKOM and Hook, inner Windows system and points out the limitation of these single detection method. An integrated detection method is proposed to detect Rootkits. The main idea and implementation steps are presented. Experimental result shows that it owns satisfied detection effect, and can detect most actions of Rootkit.

【Key words】 Rootkit technology; System Service Descriptor Table(SSDT); concealing

1 概述

Rootkit[1]最早出现于Unix系统中，随着计算机技术的发展，现在多种操作系统平台中都出现了Rootkit。目前木马的危害已经远远超过病毒，木马类恶意程序中Rootkit技术化趋势非常明显，Rootkit越来越多地出现在Windows操作系统(包括9X内核、NT内核甚至CE内核)中。大部分Rootkit的作者对操作系统底层非常了解，因此，使用的技术通常比较高深和复杂，常规检测工具和查杀软件一般难以将其检测出。如何在Windows各种平台下对Rootkit程序进行检测具有非常重要的实际意义。

文献[2]总结了一般的特洛伊木马隐藏技术，文献[3]对近几年中Rootkit技术作了一个全面的介绍，分析了一些Rootkit的技术原理和对应的查杀方法，介绍了目前已有的一些查杀软件的工作原理。文献[4-5]介绍了一些检测Rootkit的特殊方法。本文在前人工作的基础上，针对Rootkit的发展趋势提出了一种综合多种手段的检测方法。

2 实现Rootkit隐藏的常用技术

Windows系列平台下的Rootkit一般具有以下3个主要功能：(1)在目标主机上实现隐藏自身信息，如隐藏Rootkit中相关程序的进程、系统服务、文件路径、通信连接、注册表信息；(2)监视和记录操作系统行为，如截获显示屏输出、键盘输入、监视系统中进程的创建和注册表的变化等；(3)实现通信与后门服务，如执行远端攻击命令、传输文件等为攻击者提供的控制目标主机的功能。以上功能为Rootkit基本功能，但对Rootkit最重要的是隐藏功能。Windows Rootkit隐藏自身的方法按照运行级别可分为用户级(ring3)和内核级(ring0)。按照隐藏方式可以分为挂钩(Hook)方式和非挂钩(Non-Hook)方式。

2.1 实现用户级Windows Rootkit的隐藏功能的常用技术

2.1.1 修改程序导入地址表(Import Address Table, IAT)

IAT是输入地址表，是PE格式程序文件为引入的DLL所使用的结构数组，它记录程序调用了哪些系统函数及输出这些函数的DLL文件。在需要调用API函数的应用程序运行时系统会加载这些DLL文件，同时填写应用程序内存映像(即程序的内存拷贝)中IAT结构的系统调用函数的实际内存地址。在程序调用函数时，系统根据IAT中函数的实际内存地址跳到函数实际执行代码处执行函数调用。当Rootkit进入目标程序的地址空间时，Rootkit会解析PE格式程序文件的内存映像，用Rootkit指定的函数地址替代IAT中所要替换的函数地址。当目标程序调用被替换的函数时，因为IAT被修改，所以执行的是Rootkit指定的函数。

2.1.2 修改函数输出表(Export AddressTable, EAT)

函数输出表指定可执行文件(如DLL文件)的输出函数和数据，包括函数名称和地址。通过替换Windows系统和某些正常程序相关的重要DLL中的输出函数地址，即可实现目标函数的Hook。

2.1.3 覆盖DLL

在Windows平台上，执行合法程序一般都要调用一些重

基金项目：江西省自然科学基金资助项目(2007GZS1054)；华东交通大学校立科研基金资助项目(07JC03)；江西省教育厅基金资助项目(GJJ08256)

作者简介：左黎明(1981－)，男，讲师、硕士，主研方向：信息安全，非线性系统；蒋兆峰，副教授、硕士；汤鹏志，教授、硕士

收稿日期：2008-10-11 **E-mail**：limingzuo@126.com

要的 DLL，攻击者用自编的 DLL 替换常用的系统 DLL 文件，通过函数转发器将正常的调用转发给原系统 DLL，截获并处理特定的消息。通常用于替换的函数只是正常函数的包装，攻击者函数会在执行时调用正常函数，实现特定功能的仍是原来的真正函数，攻击者函数只需完成过滤返回值中特定信息的功能。

2.2 内核级 Windows Rootkit 隐藏的常用技术

在内核模式下，Windows 内核 Rootkit 常用的技术如图 1 所示。

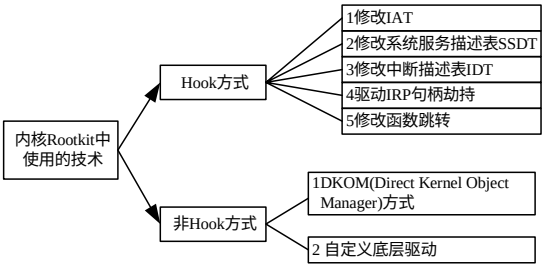

图 1 Windows 内核 Rootkit 常用技术

对于 Hook 方式，修改 IAT 与 2.1.1 节中所述原理基本相同。通过替换 SSDT 中特定类型服务号对应的服务处理函数地址，使其指向 Rootkit 指定的处理函数地址，从而可以实现对系统服务调用的 Hook。IDT 中包含中断类型号和对应处理函数地址的映射关系，通过修改 IDT 中特定类型中断号对应的处理函数地址，可以实现对操作系统中指定中断号处理的接管。内核向驱动中传递信息大部分使用的是各种 IRP，因此，通过劫持这些 IRP 包改变 IRP 包对应的驱动响应处理函数就可以实现一些驱动的劫持。而修改函数跳转，即常说的内嵌函数 HooK(inline function Hooks)的原理是修改某些正常程序中位置比较靠前的跳转(如 jmp 跳转和 call 跳转)，使其指向 Rootkit 所要执行的代码。另外，DKOM 方式是直接修改系统内核中的相关数据结构从而实现对相关信息的隐藏，如直接从内核中包含当前活动进程的 ActiveProcessLinks 链表中摘除指定进程的 EPROCESS 结构来实现进程隐藏。Rootkit 也常通过网络扩展驱动程序和文件扩展驱动程序实现底层的通信隐藏和文件隐藏。

3 Windows Rootkit 的综合检测方法与关键技术

3.1 Windows Rootkit 一般检测方法及其缺陷

Windows Rootkit 的检测较为复杂，目前已有产品中用到的检测方法主要有以下 4 种[3]：

(1)基于特征码的内核内存扫描检测。该检测方法通过在内核内存中搜索含有特征码的内存块来检测 Rootkit，但它通常只能检测出已知特征码的 Rootkit。

(2)启发式扫描检测。这是一种通过发现非常规系统行为寻找 Rootkit 痕迹的检测方法。这种方法的主要缺陷在于通常会挂钩系统频繁调用的处理函数，对操作系统性能有较大影响。

(3)基于内存完整性校验的检测。该检测方法对于采用 Hook 方式的 Rootkit 具有较好的检测效果，但一般只能判断 Rootkit 是否存在，无法判断具体类型。

(4)基于交叉视图的检测。该检测方法通过比较不同途径所枚举到的系统信息，根据其中的差异发现 Rootkit 的痕迹。使用这种方法需要保证枚举系统信息的其中一条途径应为可靠的。

使用单一方法检测的主要缺陷在于 Rootkit 作者可以通过综合使用自定义文件系统底层驱动和系统进程监视驱动的方法避免被检测。通过自定义文件系统驱动，Hook 一些文件操作相关的内核函数，屏蔽或欺骗检测软件对 Rootkit 相关文件的读写操作，使其无法获得正确的文件信息，同时以系统进程监视驱动保护文件系统驱动所关联的进程，使单一的检测方法无法获取正确的系统进程信息，最终导致检测失败。

3.2 基于 DKOM 隐藏的 Rootkit 检测方法

检测使用 DKOM 方式隐藏的 Rootkit 十分困难，目前尚无单一的有效检测方法。因此，检测这种方式隐藏进程的 Rootkit 需要综合使用多种手段，由于系统对各种进程和线程的管理，因此依赖于各种内核链表数据结构，这些链表数据结构之间有一定联系但又有差异，修改其中某一个链表结构可能不会影响其他链表。基于这个原理，利用各个内核链表结构中的一些进程 EPROCESS 结构差异对多种方法获得的进程 EPROCESS 进行交叉对比，就有可能发现 Rootkit 的迹象。本文提出一种新的综合比较方法，见图 2。

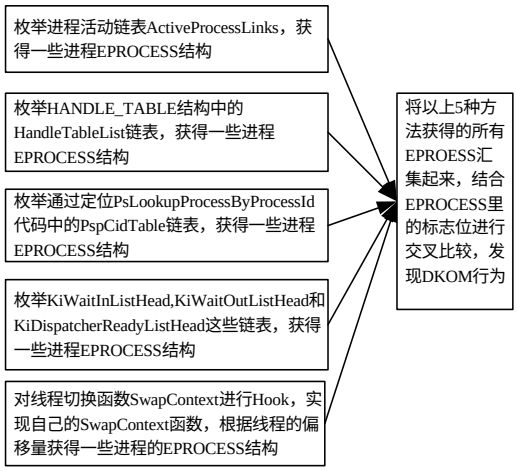

图 2 DKOM 方式 Rootkit 的综合检测方法

3.3 基于交叉视图(Cross View)的检测方法

下面以检测修改系统服务描述符表(System Service Descriptor Table, SSDT)的 Rootkit 来说明这种检测方法。由上可知，所有 SSDT 中对应的内核函数都在 Windows 内核 ntoskrnl.exe 中，当 Rootkit 加载以后，通过修改 SSDT 实现对系统内核函数进行 Hook，已经破坏了系统内存中 SSDT 的真实性。因此，如果需要检测出 SSDT 中哪些项被 Hook 了，必须构建一个原始的干净 SSDT。检测步骤如下：

(1)计算磁盘上 ntoskrnl.exe 文件的 Md5 散列值，并与未被感染情况下的散列值比较，判断其是否已经被 patch。

(2)如果当前系统的 ntoskrnl.exe 文件被 patch，则将未被 patch 的相同版本系统中的 ntoskrnl. exe 载入内存，否则直接载入当前系统的 ntoskrnl.exe，分析其导出表，获取 KeService DescriptorTable 地址。

(3)打开\device\physical memory，将 KeServiceDescriptor Table 所在物理内存的页映射到虚拟内存。

(4)在物理内存中搜索 KeServiceDescriptorTable 结构中 ServiceDescriptor[0].Service Table 的地址，并将该所在地址的物理内存的页映射到虚拟内存，以获取当前 SSDT 的内容。

(5)获取系统中所有已加载模块列表的名称、加载基地址和大小等信息。

(6)分析 ntoskrnl.exe 文件的 PE 结构，将其中的导出函数地址作为基准，和步骤(4)中所获得的 SSDT 进行比较，不同

的地址即是被 Hook 的 SSDT 项，判断修改之后的地址属于系统已加载的哪个模块的地址范围，即可获取挂钩 SSDT 中该表项地址重定位之后所处的模块信息。

值得注意的是，已加载模块信息的可靠获取是一个很重要的问题，因此，Rootkit 一般通过修改 SSDT 中 ZwQuery SystemInformation 服务函数的地址实现自身的隐藏，只要使用了与此相关的函数，得到的结果可能都不真实，这样需要找到另外一种可靠的途径对当前已加载程序的模块进行枚举。本文采用遍历 PsLoadedModuleList 链表的方法获取系统中所有已加载模块的信息。在系统内核中，PsLoadedModule List 链表地址被系统内核函数 MmGetSystemRoutineAddress 引用，所以，可以通过扫描该 API 的实现代码定位 PsLoaded ModuleList。这种方法能够比较可靠地获得已加载模块的详细信息(包括模块名、路径名、加载地址和映像大小等)，之后定位 SSDT 修改后的地址所属模块就比较简单。如果这个地址位于某模块加载起始地址和结束地址(结束地址=加载地址+映像大小)之间，则该地址位于此模块之中。修改 IAT 和 EAT 的 Rootkit 检测方法与此类似。

3.4 基于加载函数范围的检测方法

交叉视图的方法一般是通过直接读函数所在文件，然后和内存中的对比，如果之间存在差异，就有可能存在 Rootkit。本文提出另外一种检查 Hook 的方法，以确定是否存在 Rootkit。其原理如下：Windows 平台下函数的起始地址都应该在其所在的 exe 文件(或者 dll 文件，sys 文件)的内存地址范围内，如果超出了它应该存在的范围，则可能存在 Hook 或者函数转向被劫持的可能。例如在 IAT 中，程序引用的 API 函数起始地址在已加载的某个 DLL 地址范围内。所有的 IRP 包对应的发送和到达起始地址都只允许在某个内核驱动的有效地址范围内，所有在 SSDT 中登记的系统内核函数的起始地址都在 ntoskrnl.exe 的有效地址范围内。下面以检测的 ntoskrnl.exe 中的代码内嵌函数 Hook(inline function hook)为例说明这种方法。

```
/*获取跳转地址不在 ntoskrnl.exe 内的行为*/
DWORD CheckNtoskrnlForOutsideJump(DWORD dw_addr)
{
BYTE opcode=*((PBYTE)(dw_addr));
DWORD hook=0;
WORD desc=0;
// 0xe8 是汇编中的 call 指令，而 0xe9 为 jmp 指令，近地址跳转
if((opcode==0xe8)||(opcode==0xe9))
{
hook|=*((PBYTE)(dw_addr+1))<<0;
hook|=*((PBYTE)(dw_addr+2))<<8;
hook|=*((PBYTE)(dw_addr+3))<<16;
hook|=*((PBYTE)(dw_addr+4))<<24;
hook+=5+dw_addr;
}
else if(opcode==0xea) //0xea 也是 jmp 指令，远地址跳转
{
hook|=*((PBYTE)(dw_addr+1))<<0;
hook|=*((PBYTE)(dw_addr+2))<<8;
hook|=*((PBYTE)(dw_addr+3))<<16;
hook|=*((PBYTE)(dw_addr+4))<<24;
desc=*((WORD*)(dw_addr+5));
}
//检查有没有跳出 ntoskrnl.exe 的行为，如果没有返回 0，如果
//有则返回跳转地址
if(hook!=0)
{
if((hook<g_ntoskrnl.Base)||(hook>g_ntoskrnl.End))
hook=hook;
else
hook=0;
}
return hook;
}
```

3.5 基于特定函数集合的检测方法

Rootkit 要实现特定功能必须使用系统的 API 函数(包括内核级和用户级的)，因此，如果某个程序调用了危险的特定函数集合，有理由怀疑其可能是木马或者 Rootkit 之类的恶意代码。在程序加载之前，对于引入的任何程序文件，扫描其代码获得其系统函数集合(这个过程可能需要代码逆向技术和虚拟机配合)，与本文实现根据对多个木马或者 Rootkit 分析设置好的一系列特征函数集合做交集运算，就可以知道该程序文件使用了哪些危险的函数，并大致估计其功能和属于哪种类型，见表 1。

表 1 危险函数集合例子

集合 A[高危 权重 0.6]	集合 B[可能危险 权重 0.2]
ZwCreateThread, ZwWriteProcessMemory ZWOpenProcess, ZwTerminateProcess NtSetWindowsHookE, ZwLoadDriver ZwCreateFile, ZWOpenFile, ZwOpenSection	ZwSetValueKey, ZwCreateKey ZwDeleteKey, ZwDeleteKey ZwDeleteValueKey, ZwRenameKe

4 结束语

本文提出的综合检测法可以检测大部分 Rootkit 行为，与同类产品的性能比较如表 2 所示。

表 2 产品性能比较

产品	VICE	瑞星 2008	ZoneAlarm Pro	综合检测
灰鸽子免杀版	无法发现	提示恶意代码	可以发现	可以发现
FU DKOM 版	无法发现	可以发现	可以发现	可以发现
Hacker Defender	无法发现	可以查杀	可以发现	可以发现
黑洞	无法发现	提示危险进程	可以发现	可以发现

目前还出现了 Rootkit 固件化的一种技术，将 Rootkit 的部分代码写入了硬件的存储器中。未来更加先进的多态加密技术、抗逆向分析技术以及自动病毒机等将会大量出现在 Rootkit 中，Rootkit 的隐藏技术也将会有新的发展。

参考文献

[1] Hoglund G, Butler J. Rootkits: Subverting the Windows Kernel[M]. [S. l.]: Addison Wesley Professional, 2005.

[2] 张新宇, 卿斯汉. 特洛伊木马隐藏技术研究[J]. 通信学报, 2004, 25(7): 153-159.

[3] Butler J. Windows Rootkits of 2005, Part One & Part Three[DB/OL]. (2007-12-05). http: //www.securitvfocus.com/infocus/1850.

[4] 陈晓苏, 黄文超, 肖道举. 一种基于交叉视图的 Windows Rootkit 检测方法[J]. 计算机工程与科学, 2007, 29(7): 1-3.

[5] 康治平, 向 宏, 胡海波. Windows 系统 Rootkit 隐藏技术研究与实践[J]. 计算机工程与设计, 2007, 28(14): 3337-3343.

编辑 张正兴