·计算机工程与应用·

Rootkit研究综述

张 瑜[1], 刘庆中[2], 李 涛[3] , 罗自强[1], 吴丽华[1]

(1. 海南师范大学计算机系 海口 571158; 2. 美国山姆休斯敦州立大学计算机系 德克萨斯州 亨茨维尔 美国 77341;
3. 四川大学计算机学院 成都 610065)

【摘要】Rootkit是一种持久且难以察觉地存在于网络系统中的恶意代码，通过修改操作系统内核或更改指令执行路径，为攻击者提供隐匿自身、维持访问和软件窃听功能，已造成了严重的网络安全威胁。该文首先介绍了Rootkit的基本定义与演化过程，其次剖析了Windows系统中与Rootkit密切相关的内核组件和Rootkit的工作机制；然后讨论了Rootkit防御机制与检测方法；最后探讨了Rootkit的发展趋势和Rootkit防御的进一步研究方向。

关 键 词 隐遁攻击; 取证分析; 恶意代码; 网络安全; rootkit
中图分类号 TP309 **文献标志码** A doi:10.3969/j.issn.1001-0548.2015.04.016

Research and Development of Rootkit

ZHANG Yu[1], LIU Qing-zhong[2], LI Tao[3], LUO Zi-qiang[1], and WU Li-hua[1]

(1. Department of Computer Science, Hainan Normal University Haikou 571158;
2. Department of Computer Science, Sam Houston State University Huntsville, Texas USA 77341;
3. College of Computer Science, Sichuan University Chengdu 610065)

Abstract Rootkit is a set of programs that allows a permanent or consistent, undetectable presence on network systems. Rootkit can cause serious network security threat since it provides stealth access and software eavesdropping for attackers by modifying the operating system kernel data or changing instruction execution path. Firstly, the basic definition and evolution of Windows Rootkit are introduced, and the Rootkit mechanism and the Windows system kernel components are then analyzed. Thereafter, we discuss Rootkit defense mechanism and detection methods. We conclude with prediction of the trends and further research directions of Rootkit and its defense.

Key words evasion attack; forensic analysis; malware; network security; rootkit

近年来，网络攻击者(黑客)利用日益增强的网络依赖性和不断涌现的软件漏洞，通过隐遁技术远程渗透、潜伏并控制目标网络系统，悄无声息地窃取敏感信息、实施网络犯罪并伺机发起网络攻击，获取政治、经济、军事利益，已造成了严重的网络安全威胁[1]。据《2013 Norton Report》报道：全球50%的互联网用户遭受过网络攻击，中国有77%网民遭受过网络攻击，为全球第二大网络攻击受害国；全球因网络攻击造成了1 130亿美元的经济损失，中国因网络攻击造成370亿美元的损失[2]。另据IBM X-Force安全研究小组针对2013年典型攻击情况的分析调查显示，近70%的网络攻击为未知原因的攻击[3]。因此，有理由相信，隐遁攻击技术已被黑客广泛采用。

Rootkit就是在此背景下出现并迅速发展起来的一种隐遁网络攻击新技术[4]。Rootkit是一种通过修改操作系统内核或更改指令执行路径，来隐藏系统对象(包括文件、进程、驱动、注册表项、开放端口、网络连接等)以逃避或者规避标准系统机制的程序[5-6]。攻击者借助Rootkit隐遁技术对已被渗透的目标网络系统发动网络攻击，安全威胁极大[7]。据《McAfee Labs 2014 Threats Predictions》报告预测：网络攻击者将会使用更多的Rootkit隐遁攻击技术，以逃避检测与取证，获取更大利益[8]。

收稿日期：2014 – 12 – 12; 修回日期：2015 – 03 – 26.
基金项目：国家自然科学基金(61462025, 61262077, 61173159, 61463012)；海南省自然科学基金(613161, 614233)；国家级大学生创新创业训练计划资助项目(201211658036).
作者简介：张瑜(1975 –)，男，副教授，博士，主要从事网络安全、恶意代码分析与取证及智能计算等方面的研究.

鉴于Windows系统的普及性，Windows Rootkit已成为网络攻防双方的重点研究对象，本文将主要讨论Windows系统里的Rootkit技术。

Windows系统的Rootkit技术研究始于1999年在著名黑客杂志《Phrack》上发布的《A Real NT Rootkit》[9]。该文创造性地提出了多项Windows系统内核隐遁技术，获得了信息安全社区的极大关注。之后，几乎每期《Phrack》都有与Rootkit相关的技术论文。此外，著名的黑客大会(Black Hat、Def Con等)从2005年也开始了针对Rootkit技术的讨论议题。

工业界和政府部门同样关注Rootkit技术研究与应用。日本索尼公司在2005年使用Rootkit技术保护其BMG CD版权，以防止光碟被非法复制[10]。美国和以色列联合研制的采用了Rootkit隐遁技术的“震网病毒Stuxnet”[11]，于2010年重创伊朗核电设施，严重滞后其核计划。2013年震惊世界的美国“棱镜计划”[12]，隐秘渗透目标系统并植入采用Rootkit隐遁技术的恶意软件，实施暗中监控、窃取政情军情，并发起定向隐遁网络攻击。与此同时，Rootkit检测防御工具也相继出现，如Russinovich编写的Rootkit Revealer，EP_X0FF编写的Rootkit Unhooker，Rutkowska编写的System Virginity Verifier，Linxer开发的PCHunter，Dmitry开发的Tuluka Kernel Inspector，GMER团队开发的GMER，F-Secure公司的Blacklight，McAfee公司的Rootkit Remover，Kaspersky公司的TDSSKiller及Sophos公司的Anti-Rootkit等。

国内对Rootkit及其防御技术的关注和研究相对较早。中国科技大学、上海交通大学、电子科技大学、解放军信息工程大学、北京大学等高校已相继开展Rootkit及其防御技术分析研究。相关Rootkit检测工具主要有：IceSword、DarkSpy及Linxer开发的PCHunter等[13-20]。

作为一种隐遁网络攻防的有力武器，Rootkit及其防御技术已成为信息安全领域研究者所共同关注的热点。关于Rootkit研究综述已取得部分研究成果。如文献[21]从Rootkit产生机制的角度综述了当前的Rootkit技术发展、分类及防御，主要侧重于Linux系统；文献[22]从Rootkit防御机制方面综述了现今Rootkit检测技术的研究进展；文献[23]从取证的视角综述了目前Rootkit技术研究进展；文献[24]综述了Android系统Rootkit的实现原理及检测方法。

但目前国内尚未有详细而全面介绍Windows系统Rootkit机理与研究成果的综述论文。为深入理解Windows Rootkit机理和发展趋势，总体把握Windows Rootkit及其防御研究进展，并促进国内在该方向上的研究，综述Windows系统Rootkit研究进展工作非常必要。

1 Windows系统Rootkit起源与演化

1.1 Windows Rootkit定义

Rootkit一词源于Unix系统。在Unix系统中，Root是指拥有所有特权的管理员，而Kit是管理工具，因此，Rootkit是指恶意获取管理员特权的工具。利用这些工具，可在管理员毫无察觉的情况下获取Unix系统访问权限。

对于Windows Rootkit，尽管在名称上沿用了Unix系统的Rootkit，但在技术上则继承了DOS系统相关隐形病毒技术：拦截系统调用以隐匿恶意代码。最早出现的Windows Rootkit——NT Rootkit，由文献[9]提出并编码实现，且对后来的Rookit研究产生了极大的影响。

文献[6]给出的定义为，Windows Rootkit是能够持久或可靠地、无法检测地存在于计算机上的一组程序和代码。俄罗斯著名的Kaspersky实验室反病毒专家将Windows Rootkit的定义为一种通过使用隐形技术来隐藏系统对象(包括文件、进程、驱动、服务、注册表项、开放端口、网络连接等)以逃避或者规避标准系统机制的程序[25]。尽管上述定义不尽相同，但都刻画出了Windows Rootkit的本质特征[21-22]：隐匿性、持久性、越权性。因此，从本质上分析，Rootkit是破坏Windows系统内核数据结构及更改指令执行流程[26-27]的代码。鉴于此，本文给出如下定义：Windows Rootkit是一种越权执行的程序或代码，常以驱动模块加载至系统内核层或硬件层，拥有与系统内核相同或优先的权限，进而修改系统内核数据结构或改变指令执行流程，以隐匿相关对象、规避系统检测取证，并维持对被入侵系统的超级用户访问权限。

1.2 Windows Rootkit演化

本质上，Rootkit通过修改代码、数据、程序逻辑，破坏Windows系统内核数据结构及更改指令执行流程，从而达到隐匿自身及相关行为痕迹的目的。由于IA-32硬件体系结构缺陷(无法区分数据与代码)和软件程序逻辑错误的存在，导致Rootkit将持续存在并继续发展。回顾Windows Rootkit技术演化历程，其遵循从简单到复杂、由高层向低层的演化趋势。从Rootkit技术复杂度的视角，可将其大致划分为

5代[23,28-29]：1) 更改指令执行流程的Rootkit，2) 直接修改内核对象的Rootkit，3) 内存视图伪装Rootkit，4) 虚拟Rootkit，5) 硬件Rootkit。Rootkit的演化发展时间轴(Rootkit技术典型实例)如图1所示。

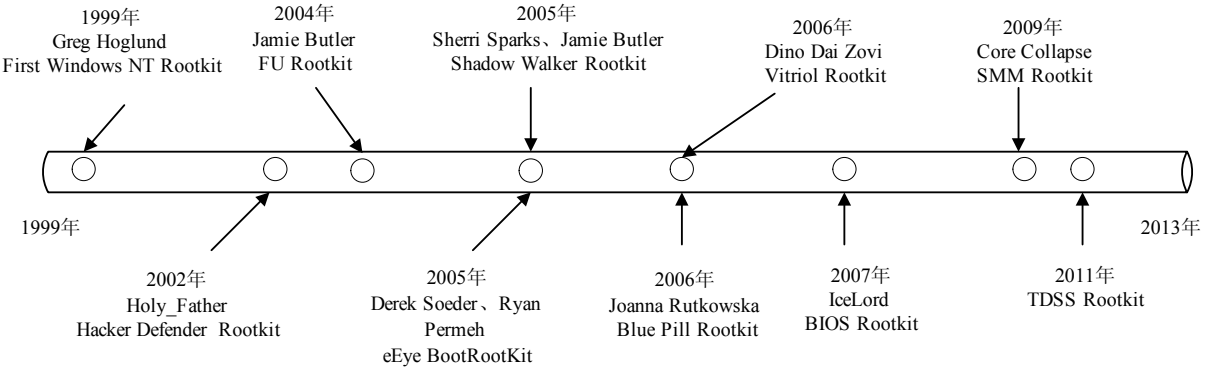

图1 Windows Rootkit起源与发展时间轴

对于更改指令执行流程Rootkit，主要采用Hooking技术。Windows系统为正常运行，需跟踪、维护诸如对象、指针、句柄等多个数据结构。这些数据结构通常类似于具有行和列的表格，是Windows程序运行所不可或缺的。通过钩挂此类内核数据表格，能改变程序指令执行流程：首先执行Rootkit，然后执行系统原来的服务例程，从而达到隐遁目的。此类Rootkit钩挂的内核数据表格主要有：Windows系统的系统服务调度表(system service dispatch table，SSDT)、中断调度表(interrupt dispatch table，IDT)、全局描述符表(global descriptor table，GDT)、局部描述符表(local descriptor table，LDT)、特别模块寄存器(model-specific registers，MSR)、I/O请求包(I/O request packets，IRP)。由于Hooking技术的普及，使此类Rootkit(如NT Rootkit、He4hook、Hacker Defender等)较易被检测与取证分析。攻防博弈的持续，促使Rootkit技术不断向前演化，进入了直接修改内核对象技术发展阶段。

对于直接修改内核对象的Rootkit，与传统Hooking技术更改指令执行流程不同，它直接修改内存中本次执行流内核和执行体所使用的内核对象，从而达到进程隐藏、驱动程序隐藏及进程特权提升等目的。2004年，文献[30]首次展示了该项新技术直接进行内核对象操纵(direct kernel object manipulation，DKOM)，并编写了FU Rootkit。该Rootkit通过修改内核EPROCESS结构中的ActiveProcessLinks双链表来隐藏进程。之后，在FU Rootki基础上的FUTo Rootkit[31]，不是修改进程列表结构，而是修改PspCidTable结构来隐藏进程。2011年出现的TDSS Rootkit[32-33]则直接修改驱动程序(包括防御杀毒软件)来隐匿自身。然而，自Rutkowska编写的SVV (system virginity verifier)[34-35]检测工具后，此类Rootkit也易被检测出来。所以，Rootkit与Rootkit防御技术的魔道之争还将持续，从直接修改内核对象进入内存视图伪装技术阶段。

对于内存视图伪装的Rootkit，主要通过创建系统内存的伪造视图以隐匿自身。此类Rootkit主要利用底层CPU的结构特性，即CPU将最近使用的数据和指令分别存储在两个并行的缓冲器：数据快速重编址缓冲器(data translation lookaside buffers，DTLB)和指令快速重编址缓冲器(instruction translation lookaside buffers，ITLB)。通过强制刷新ITLB但不刷新DTLB引发TLB不同步错误，使读写请求和执行请求得到不同的数据，从而达到隐匿目的[26]。2005年，文献[36]首次演示了该项隐匿技术，即内存伪装技术(memory cloaking)，并编写了ShadowWalker Rootkit。然而，在Microsoft公司为其64位Windows系统引入Patchguard[37]技术后，基本宣告了内核层Rootkit技术的终结。从此，Rootkit技术开始进入虚拟领域以期占据攻击优势。

对于虚拟Rootkit，与直接修改操作系统的Rootkit不同，它专门为虚拟环境而设计，通过在虚拟环境之下加载恶意系统管理程序，完全劫持原生操作系统，并可有选择地驻留或离开虚拟环境，从而达到自如隐匿目的。就其类型而言，可分为3种[26,38]：1) 虚拟感知恶意软件(virtualization-aware malware，VAM)，2) 基于虚拟机的Rootkit(virtual machine based Rootkit，VMBR)，3) 系统管理虚拟Rootkit (hypervisor virtual machine Rootkit，HVMR)。对于虚拟感知恶意软件，它只是增加了检测虚拟环境功能。对于基于虚拟机的Rootkit，能在虚拟机内部封装原生操作系统，如文献[39]于2006年发布了通过修改启动顺序将原生操作系统加载到虚拟环境中的SubVirt。对于系统管理虚拟Rootkit，主要利用CPU

硬件虚拟技术支持，通过定制的系统管理程序替代底层原来系统管理程序，进而在运行中封装当前运行的操作系统。如文献[40]利用AMD Pacifica技术在AMD Athon 64上实现的Blue Pill；文献[41]利用Intel VT-X技术在Intel Core Duo上实现的Vitriol。然而，在防御者利用逻辑差异、资源差异、时间差异等方法[42-44]检测到虚拟Rootkit后，攻防博弈开始向底层硬件方向发展。

对于硬件Rootkit，又称为Bootkit，源于2005年由文献[45]提出的eEye BootRootKit。它通过感染主引导记录(master boot record，MBR)的方式，实现绕过内核检查和启动隐身。从本质上分析，只要早于Windows内核加载，并实现内核劫持技术的Rootkit，都属硬件Rootkit技术范畴。如文献[46-48]发布的SMM Rootkit，它能将自身隐藏在系统管理模式(system management mode，SMM)空间中。由于SMM权限高于虚拟机监控(virtual machine monitor，VMM)，设计上不受任何操作系统控制、关闭或禁用；此外，由于SMM优先于任何系统调用，任何操作系统都无法控制或读取SMM，使得SMM Rootkit有超强的隐匿性。之后，陆续出现的BIOS Rootkit[49-50]、VBootkit[51-53]等都属于硬件Rootkit范畴。

从上述发展轨迹中可以看出，Windows系统Rootkit及其防御技术在网络安全攻防博弈中几乎同步发展、相互促进。因此，随着应用软件技术、操作系统技术和硬件技术的发展，Rootkit技术的攻防博弈还将持续下去。

2 Windows Rootkit工作机制

2.1 Windows系统内核结构及关键组件

Rootkit本质上是破坏Windows系统内核。为理解Rootkit的工作机制，首先需了解Windows系统内核结构和关键组件及其功能。

Windows系统采用层次化设计[54]，自底向上可分为3层：1) 硬件抽象层(hardware abstraction layer)；2) 内核层(OS kernel layer)；3) 应用层(application layer)。硬件抽象层的设计目的是将硬件差异封装起来，从而为操作系统上层提供一个抽象一致的硬件资源模型。内核层实现操作系统的基本机制和核心功能，并向上层提供一组系统服务调用API(application programming interface)函数。应用层通过调用系统内核层提供的API函数实现自身功能。Windows系统层次结构如图2所示。

Windows系统的层次化设计，使其容易扩展、升级相关功能，同时，也给攻击者以可乘之机。Rootkit正是利用Windows系统层次模型中的上下层接口设计，通过修改下层模块返回值或修改下层模块数据结构来欺骗上层模块，从而达到隐匿自身及其相关行为踪迹目的。在Windows系统中，常被Rootkit利用的内核组件主要包括：进程(线程)管理器、内存管理器、I/O管理器、文件管理器、网络管理器、安全监视器和配置管理器[55]。

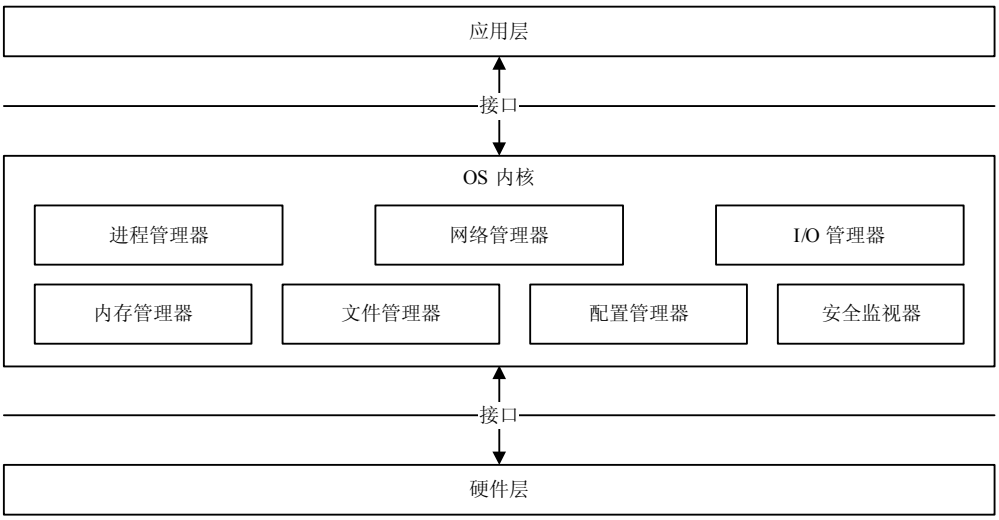

图2 Windows系统层次结构

针对Windows系统内核的不同组件，Rootkit将采取不同技术加以利用。具体而言，进程(线程)管理器负责进程和线程的创建和终止，并使用相关数据结构EPROCESS和ETHREAD记录所有运行的进程与线程。Rootkit通过修改这些数据结构就可隐匿相关进程。

内存管理器实现虚拟内存管理：既负责系统地址空间管理，也负责每个进程地址空间管理，并支

持进程间内存共享。Rootkit通过修改全局描述符表(global descriptor table，GDT)和局部描述符表(local descriptor table，LDT)中的相关值，就能获取特权去修改相关内存页面的读写信息。

I/O及文件系统管理器，负责将IRP分发给底层处理文件系统的设备驱动程序。Rootkit通过在高层钩挂I/O及文件系统管理器所提供的API函数或在低层拦截IRP请求的方式，实现文件和目录隐藏。

网络管理器负责系统的网络协议实现、网络连接管理功能，自底向上主要包括2个接口：1) 网络驱动程序接口规范(network drivers interface specification，NDIS)；2) 传输驱动程序接口(transfer drivers interface，TDI)。NDIS对低级协议进行抽象，TDI在NDIS基础上进一步抽象其细节，并向上提供相关网络API函数。Rootkit可对上述任意一个接口API函数进行代码修改或钩挂，以实现相关网络流量隐藏。

安全监视器负责实施安全策略确保系统安全有序运行。Rootkit通过对内核这部分代码的修改，就可删除所有安全机制，使其畅行无阻。配置管理器负责系统注册表的实现与管理。Rootkit通过修改或钩挂相关API函数即可隐藏相关进程的注册表键值。

2.2 Rootkit技术分类

Rootkit设计目的是成为无法检测与取证的软件，而这主要通过修改Windows系统内核数据结构或更改指令执行流程，从而掩盖其存在的隐身功能来实现，因此，Rootkit的工作机制主要围绕隐身功能而展开。以应用程序调用Windows API函数FindNextFile为例，其调用执行路径上可被Rootkit利用之处如图3所示。

尽管Rootkit数量繁多，但从其所采用技术的视角，可将其划分为5类[56-58]：1) Hooking技术，2) 过滤驱动程序(filter drivers)技术，3) 直接内核对象操纵(direct kernel object manipulation，DKOM)技术，4) 虚拟(virtualization)技术，5) 硬件修改(hardware manipulation)技术。

2.2.1 Hooking技术

Windows系统采用事件驱动机制，通过消息传递来实现其功能。Hooking是Windows系统中非常重要的系统接口，可监控系统或进程中的各种事件消息，截获并处理发送给其他应用程序的消息。Rootkit为实现其隐身功能，需更改指令执行路径，而Windows系统的分层模型和Hooking机制给Rootkit提供了绝佳机会。

只要Rootkit能访问目标进程地址空间，它就可钩挂并修改其中的任何函数，以完成隐藏进程、隐藏文件、隐藏网络端口、防止其他程序访问特定进程句柄等功能。Windows系统中能被钩挂的地方很多，本文将从用户模式到内核模式的顺序介绍Rootkit所使用的Hooking技术。Hooking技术主要包括：IAT Hooking、Inline Hooking、IDT Hooking、SSDT Hooking、GDT/LDT Hooking、IRP Hooking等。

1) IAT Hooking

在Windows系统中，当可执行文件运行时，会将该文件的可移植执行体(portable executable，PE)结构加载入内存；同时，Windows装入程序会将该可执行文件映像的IMAGE_IMPORT_DESCRIPTOR结构中所包含的来自DLL的函数，制作成一个函数指针的表格，称为输入地址表(import address table，IAT)。当Rootkit进入应用程序的地址空间，通过分析内存中目标应用程序的PE格式，可将IAT表中的目标函数地址替换为Rootkit函数地址；之后，当目标函数被调用时，就会执行Rootkit函数而非原始函数。图4描述了IAT Hooking执行后流程的变化。

IAT Hooking技术尽管简单而强大，但存在2个缺点：① 此类Hooking技术易被检测发现，② 如应用程序通过LoadLibrary和GetProcAddress寻找函数地址，则IAT Hooking将失效。

2) Inline Hooking

如果说IAT Hooking是通过在IAT表中更改函数地址实现的话，那么，Inline Hooking则是通过硬编码来改变目标函数头部代码的方式，使其跳转到Rootkit设定好的函数执行。为使Rootkit顺利执行，Inline Hooking通常需完成3个任务：① 重新调整当前堆栈；② 执行被覆盖的指令；③ 过滤信息实现隐身。具体而言，Rootkit一般会向目标函数写入跳转指令JMP，在程序进行跳转时，相关API函数还没执行完；为确保相关API函数在顺利执行后返回至Rootkit函数中，需重新调整当前堆栈；在Rootkit执行之后务必使原来被覆盖的指令顺利执行，并根据需要进行信息过滤以实现隐身功能。Inline Hooking工作机制如图5所示。

根据Inline Hooking所在目标函数位置的不同，可将其分为3类：① 函数头部Hooking；② 函数中间Hooking；③ 函数尾部Hooking。一般而言，在目标函数中Inline Hooking的位置越深，代码重返时所需考虑的问题就越多，如稍有不慎，就会造成蓝屏死机(blue screen of death，BSoD)。因此，从实现与

检测的角度，第1、第3类Inline Hooking较为简单， 而第2类Inline Hooking技术则较为复杂。

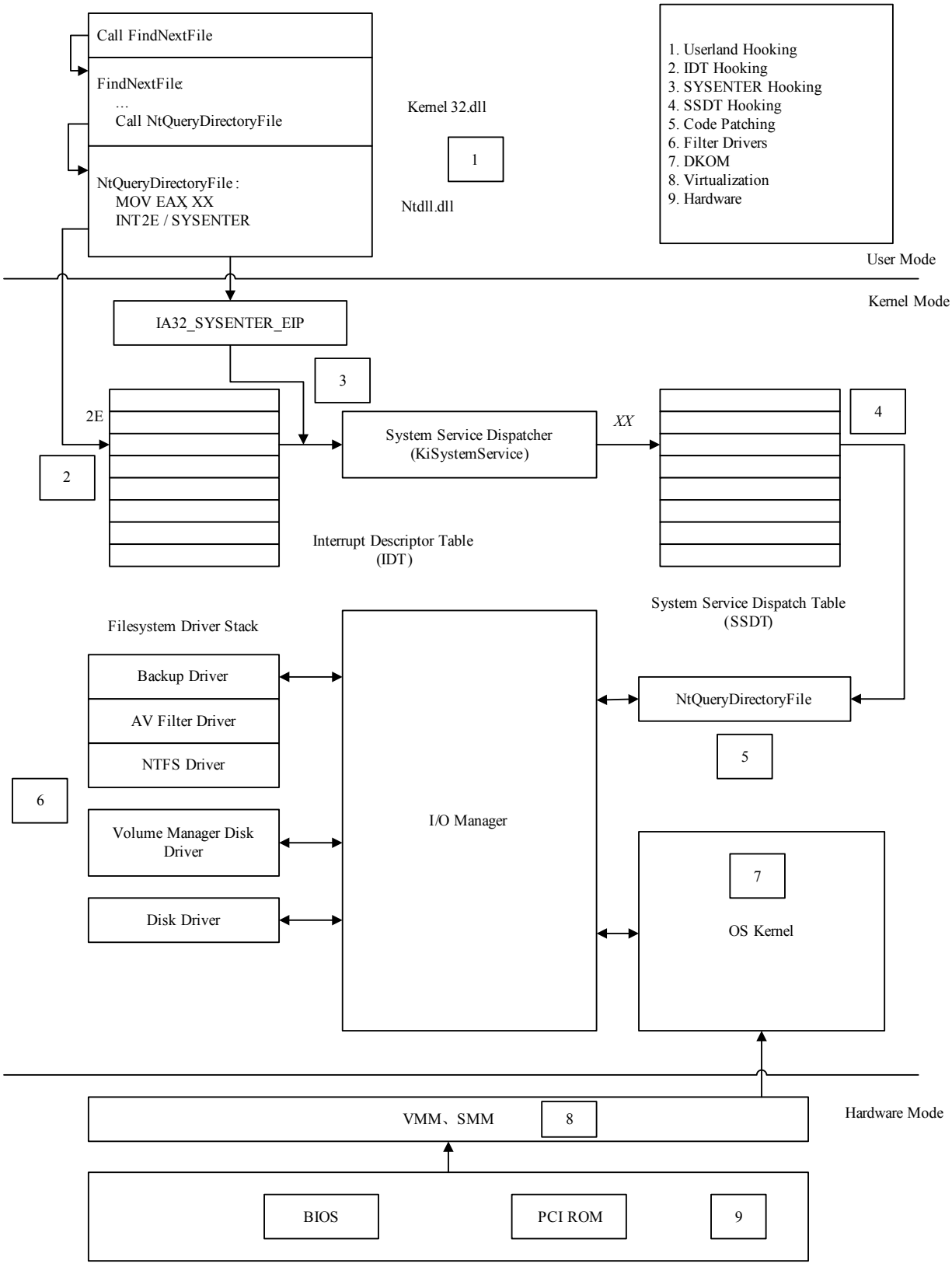

图3 Rootkit藏身之处

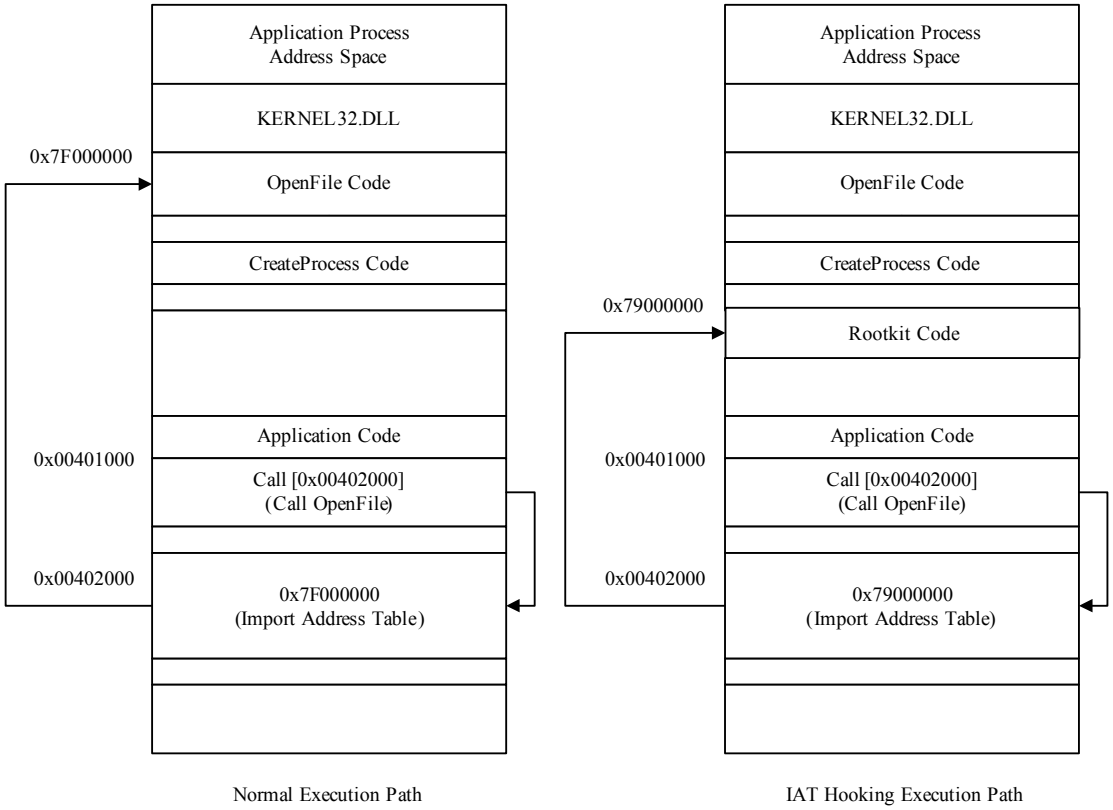

图4 IAT Hooking工作机制

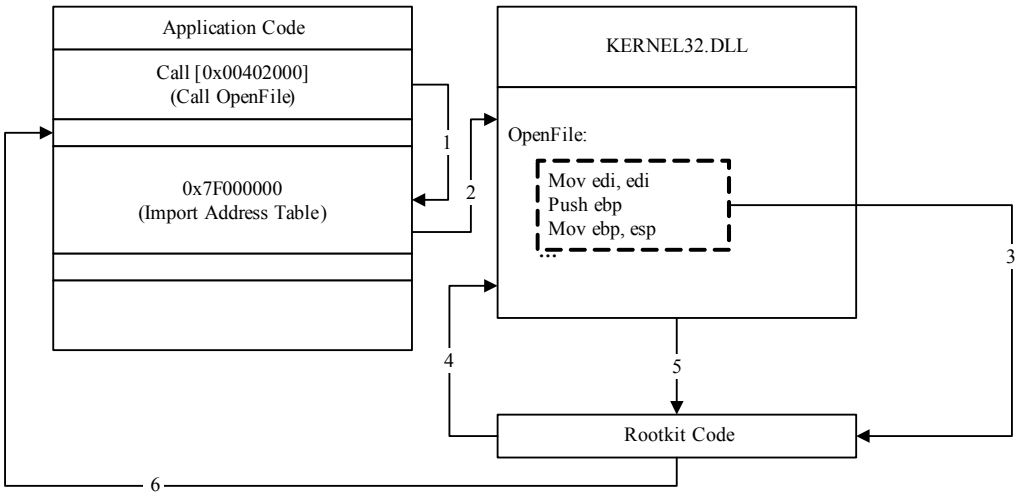

图5 Inline Hooking工作机制

3) IDT Hooking

中断描述符表(interrupt descriptor table，IDT)是一个有256个入口地址的线性表，每个IDT的入口地址为8字节的描述符，每个中断向量关联了一个中断处理例程。操作系统使用IDTR寄存器记录IDT位置和大小，可用SIDT指令读出IDTR寄存器中的信息，LIDT指令将新信息重写入IDTR寄存器。

IDT Hooking通常替换IDT表中的0x2E索引项，即KiSystemService系统服务处理函数。但Windows系统目前多使用快速调用方法：NTDLL将请求系统服务调用号载入EAX寄存器，将当前堆栈指针ESP载入EDX寄存器，再发出SYSENTER指令。SYSENTER指令将控制权传递给MSR(Model-Specific Register，特别模块寄存器)中的IA32_SYSENTER_EIP寄存器。Rootkit通过覆盖这个寄存器，就可改变每个系统服务调用的执行流程，并根据需要拦截和修改信息，从而达到隐身目的。

4) SSDT Hooking

系统服务描述符表(system service descriptor table，SSDT)是Windows系统用于用户模式请求系统服务调用的查找表格。当发出INT 0x2E或SYSENTER指令时，就会激活系统服务调用程序，然后通过查找SSDT表获取相关函数地址并执行。Windows系统内核中的系统服务描述符表有2个：① KeServiceDescriptorTable，由NTOSKRNL.EXE导出；② KeServieDescriptorTableShadow。两者区别是：KeServiceDescriptorTable仅有NTOSKRNL一项，KeServieDescriptorTableShadow包含了NTOSKRNL和WIN32K。一般的Native API服务地址由KeServiceDescriptorTable负责分派，而GDI.DLL/USER.DLL内核API服务地址则由KeServieDescriptorTableShadow负责分派。

SSDT Hooking通过修改KeServiceDescriptorTable地址，使控制权重定向到Rootkit代码，再将被修改后的假消息传回至应用程序，从而有效隐匿自身及相关行为痕迹。SSDT Hooking一般通过2个步骤完成：① 定位KeServiceDescriptorTable；②替换SSDT表中需修改的索引项，使其指向Rootkit代码。然而，防御者会采用持续监控数据结构修改或完全限制访问该结构的方式去对抗此类Rootkit。如Windows在X64系统上已使用Patchguard技术，在系统启动和运行时都要检查NTOSKRNL.EXE内的系统表格，以避免相关表格被Rootkit修改。

5) GDT/LDT Hooking

全局描述符表(global descriptor table，GDT)用于存放描述内存区域的地址和访问特权的段描述符，如数据和代码段描述符、任务状态段TSS描述符等。在保护模式的段式内存管理机制中，段逻辑地址到线性地址的映射是通过GDT表来完成。GDT表中存放着大量段描述符表结构，其结构体中包含一个过程调用函数基地址，该地址加上段偏移地址即可得出线性物理地址，如图6所示。GDT表在每次访问内存时由CPU使用，以确保执行代码有权访问在段寄存器中指出的内存段。寄存器GDTR存放着GDT的入口地址，通过指令LGDT可将GDT的入口地址装入此寄存器，通过指令SGDT可获取该寄存器内容。局部描述符表(local descriptor table，LDT)本质上也一样，只是它是按进程而GDT是按处理器来划分的。寄存器LDTR保存有LDT的入口地址，通过指令LLDT可将LDT的入口地址装入此寄存器，通过指令SLDT可获取该寄存器内容。

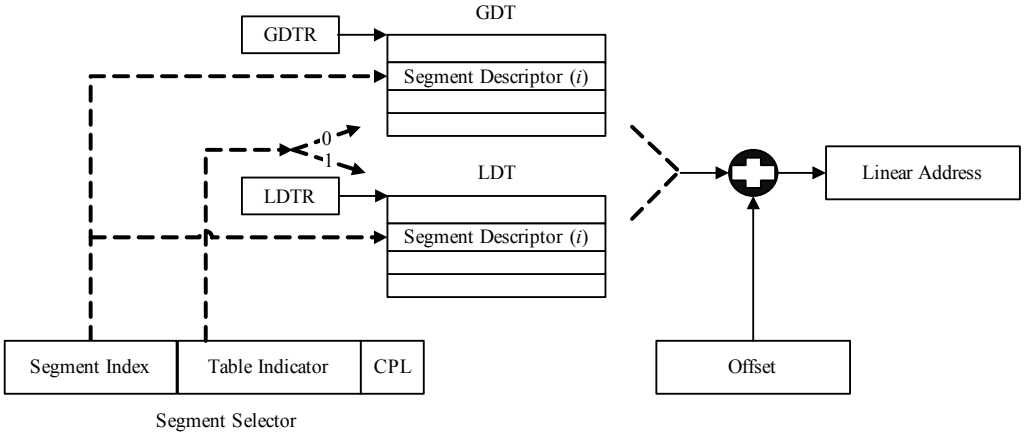

图6 GDT/LGT地址转译

GDT/LDT Hooking通过在GDT/LDT表的保留字段中插入一个调用门描述符来更改内存段的执行特权，再借助诸如DKOM技术来实施进程隐藏。因此，GDT/LDT Hooking的实现步骤是：① 创建一个包含Rootkit代码的调用门，② 读取GDTR寄存器以定位GDT基地址，③ 将创建的调用门插入GDT中的空描述符子项。

6) IRP Hooking

IRP(I/O request packets)是Windows系统内核的一个数据结构[55]。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请求转换成相应的IRP，再根据IRP数据结构中的MajorFunction数组，通过IofCallDriver函数将IRP分派给设备堆栈中的不同派遣例程进行处理。

IRP Hooking是在IRP传递过程中进行拦截并处理有关IRP以隐藏相关信息的一种Rootkit技术。既然IRP在分派过程中先后经过IofCallDriver和设备堆栈中的不同派遣例程，因此，可通过3种方式予以拦截：

① 拦截IofCallDriver函数；② 在设备堆栈中，将一个过滤驱动程序放置在处理欲拦截IRP的那个驱动程序之上；③直接修改处理IRP派遣例程的MajorFunction数组。当IRP被Rootkit函数拦截并处理后，再将其传递给原调度例程完成相关操作。IRP传递及IRP Hooking处理流程如图7所示。

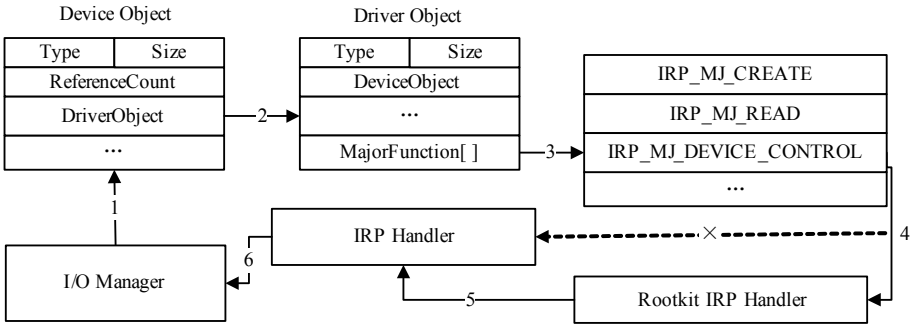

图7　IRP Hooking工作机制

2.2.2　Filter Drivers技术

Windows系统的层次模型也体现在驱动程序设计中，多数Windows驱动程序采用层次结构来传递信息。当I/O管理器接到一个IRP时，会将其传递给一个驱动程序/设备栈。栈顶的驱动程序首先处理该IRP，然后依次将其向下传递，直至该IRP被完成。栈底的驱动程序通常是与硬件关联的，负责直接驱动硬件设备完成IRP任务。

Filter Drivers技术是利用Windows分层驱动程序模型，将Rootkit驱动程序插至驱动程序/设备栈中，拦截、修改并过滤掉合法驱动程序需处理的信息，从而达到隐藏文件、嗅探击键、隐匿网络连接等目的。Filter Drivers工作机制如图8所示。

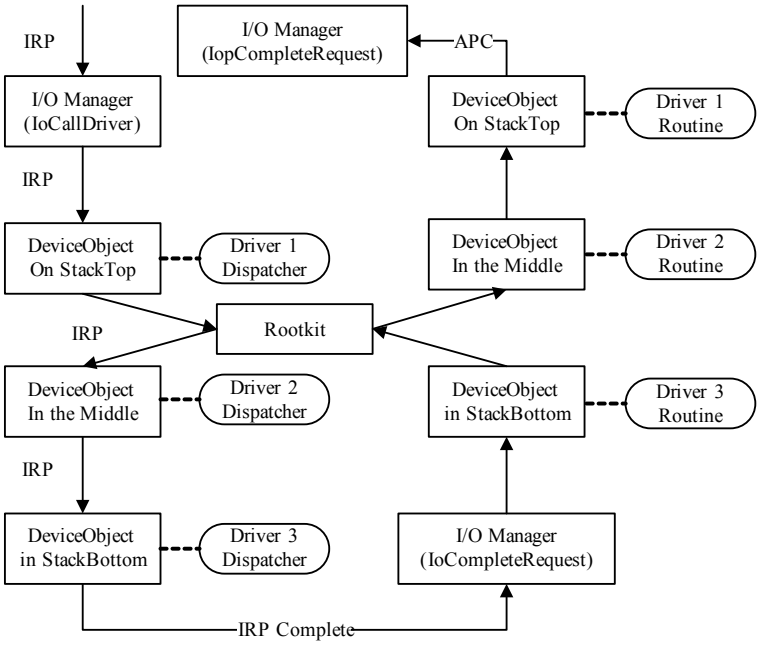

图8　Filter Drivers工作机制

2.2.3　DKOM技术

前面讨论的Hooking技术和Filter Drivers技术，都涉及利用相关机制去更改或重定向指令执行流程。与此不同，DKOM(direct kernel object manipulation)技术通过直接修改Windows系统中代表进程、驱动程序和特权的内核对象，以实现进程隐藏、驱动程序隐藏和进程特权提升。譬如，对于Windows系统中的每个进程，都有一个对应的内核EPROCESS对象，该对象中的ActiveProcessLinks为一个双链表节点，所有活动进程都连接在一起构成一个双链表，Rootkit只须将需要隐藏的进程节点从该双链表中断开，就可达到隐藏进程目的，如图9所示。代表驱动程序的DRIVER_OBJECT对象，也有指向其他驱动程序的指针DriverObject，Rootkit只

要遍历该列表，删除要隐藏的驱动程序项即可将其隐藏。此外，通过修改进程令牌数据结构，可向进程令牌中添加相关权限值以提升该进程的权限。

2.2.4 Virtualization技术

由于传统的Rootkit与检测工具同处操作系统内核层，因而容易被检测出来。与此不同，虚拟Rootkit借助于软件和硬件的虚拟化支持将自身插入至操作系统之下，使传统的位于操作系统这一级别的检测工具失效，从而达到隐身目的[59]。从虚拟Rootkit所处计算机系统层次，可将其划分为2种：① VMM Rootkit；② SMM Rootkit。

1) VMM Rootkit

虚拟机管理器(virtual machine manager，VMM)，是一种运行在物理硬件和操作系统之间的中间软件层，管理从操作系统到共享资源间的映射，可允许多个操作系统和应用程序共享硬件。VMM Rootkit一般在硬件虚拟化技术支持下(如AMD-V或Intel VT-x)，通过在客户操作系统中安装内核驱动程序，创建新的系统管理程序和新的虚拟机来放置原生操作系统，使Rootkit运行于原生操作系统之下并控制整个系统以进行隐身，如图10所示。一旦VMM Rootkit成功加载，它就能透明地截获和修改被放置在虚拟机中的原生操作系统的状态和事件，如截获击键、网络数据包、内存数据、磁盘读写等。此类虚拟Rootkit可细分为2类：① 创建新的系统管理程序并将其自身插至原生操作系统之下的虚拟Rootkit，如SubVirt；② 修改系统管理程序并将原生操作系统封装在虚拟机中的虚拟Rootkit，如BluePill和Vitriol。

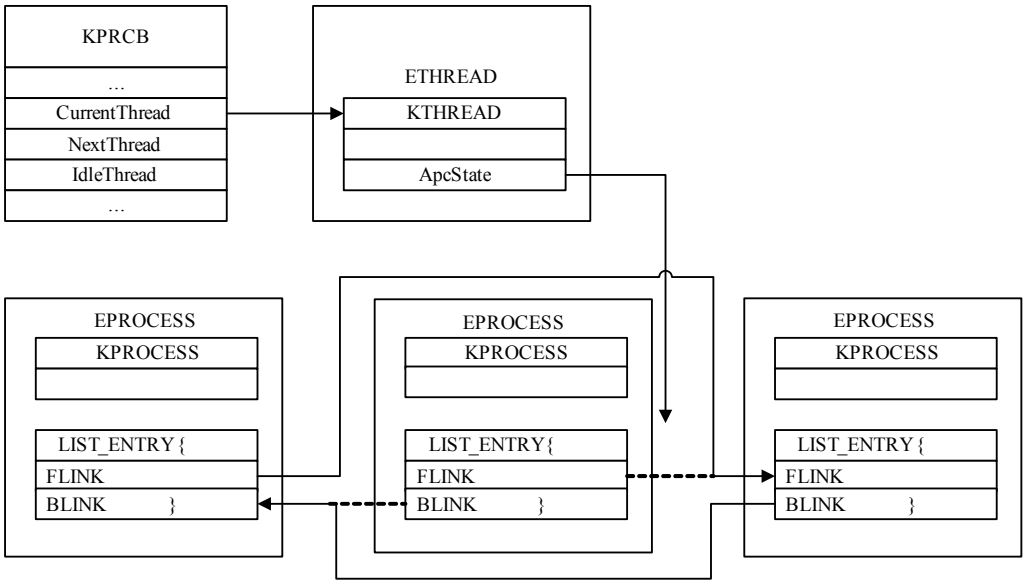

图9 DKOM工作机制

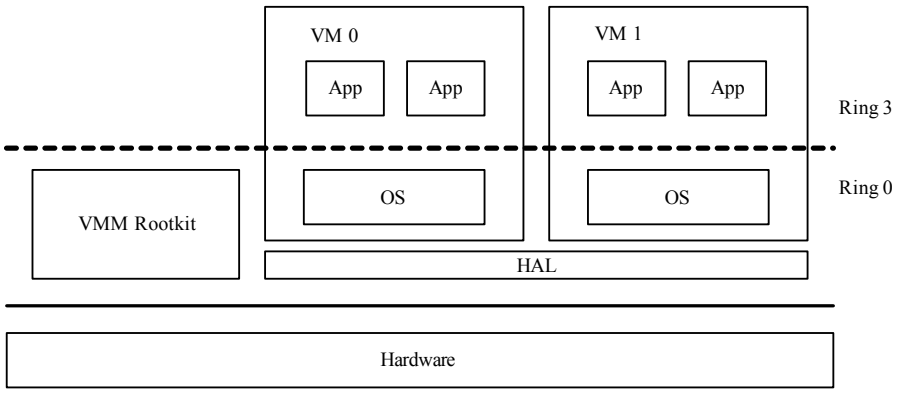

图10 VMM Rootkit所处系统层次

2) SMM Rootkit

Intel系列CPU支持4种运行模式：① 实模式(real mode)；② 保护模式(protection mode)；③ 虚拟8086模式(virtual 8086 mode)；④ 系统管理模式(system management mode，SMM)。其中，实模式是16位地址模式，保持处理器向前兼容，仅在启动时才会用

到；保护模式是32位地址，提供了现代操作系统使用的保护模式；虚拟8086模式用于运行为8086等体系结构编写的程序；系统管理模式是专门用于电源管理和温度调节。Intel构架下的系统管理模式和其他运行模式的转换规则为：任何运行模式在发出SMI(system management interrupt)中断时都可切换至系统管理模式，系统管理模式可通过发出RSM指令返回到前一个运行模式。

重要的是，SMM是一种隔离的、特权的运行环境，其最大特点是：有专用的内存区域和执行环境，外部程序不可见，且SMM不受优先级和内存保护等限制。因此，由于任何操作系统都无法控制或读取SMM，使得SMM RootKit有超强的隐匿性。SMM Rootkit可控制外围硬件设备，与诸如网卡、键盘、鼠标、硬盘、显卡等外围设备交互，从而避免被原生操作系统察觉。文献[47]首先进行了SMM Rootkit研究;文献[48]实现了与键盘和网卡透明交互的概念验证型的SMM Rootkit；文献[46]实现了一个通过劫持SMI的键盘记录型SMM Rootkit。

2.2.5 Hardware Manipulation技术

由计算机系统启动顺序可知，加电后最先启动的是BIOS芯片代码，在其完成对芯片组、内存条及外围设备自检后，才将控制权交给操作系统以完成后续系统启动。因此，Rootkit如能寄存于BIOS芯片中，则将先于操作系统控制整个系统，且能在系统重启、操作系统重装、磁盘格式化等情况下继续运行，更重要的是其他取证和检测工具对其无能为力。

其实，遵循此思路的硬件利用技术由来已久，早在1998年CIH病毒曾成功感染BIOS芯片，使用垃圾数据刷新BIOS，致使机器无法启动。真正意义上的硬件Rootkit是NGS Consulting公司所编写[50]：利用ACPI(advanced configuration and power interface，高级配置和电源接口)功能，禁止传统操作系统进程进入该内存空间，从而达到隐身目的。此外，还可使用PCI扩展ROM作为装入机制来编写相关硬件Rootkit，如eEye BootRoot[45]是一种修改系统启动扇区的Rootkit。

3 Rootkit检测

从宏观的方法论角度，可将恶意代码检测方法分为3类[34]：误用检测、异常检测和完整性监测。误用检测，是按照已分析并提取的代码模式，在内存或文件中去搜寻比对可疑数据，以此判定是否为恶意代码。该检测法的优点是能高效检测已知恶意代码；其不足是不能识别未知或变种恶意代码。异常检测，是通过定义一组系统处于正常情况时的数据(如CPU利用率、内存利用率、文件校验和等)；然后分析是否偏离正常行为以判定是否出现异常。该检测法能有效检测未知恶意代码，但其不足在于不能精确定义正常数据。完整性监测是通过监测可信系统中系统文件和内核内存的变化，来判定系统是否受到恶意代码攻击。该检测法能有效识别未知和变种恶意代码，但实现困难。

对于Rootkit检测，可将这些宏观检测理论具体化为软件检测法和硬件检测法2大类检测方法。

3.1 软件检测法

Rootkit的软件检测法较常见，主要包括：特征码检测法、完整性校验法、Hooking检测法、视图差异检测法等。

1) 特征码检测法

特征码分析与检测法通过查寻文件或内存中的Rootkit特征码来加以判断。该方法首先分析已知Rootkit样本并提取其特征码以建立特征码库；然后扫描文件或内存中的二进制指令代码并判断是否匹配特征码库。该方法尽管能精准检测部分已知Rootkit，但由于Rootkit通常会采取新的Hook技术或DKOM技术，在文件系统或内存中隐匿其静态文件或动态进程，导致特征码检测法根本扫描不到该类Rootkit，很容易被绕过。因此，对于已知Rootkit变种和未知Rootkit，因尚未理解其运行机制提取不到特征码而使该方法无从检测。

特征码检测法是一种通用的检测技术，目前的商业化Rootkit安全产品多采用特征码方法进行检测。但随着变形、加密、加壳等相关技术的出现，需结合其他方法，才能有效检测。

2) 完整性检验法

完整性分析与检测法通过检查系统文件或内存中的系统内核组件的完整性来加以判断。该方法首先建立未感染系统的重要系统文件的Hash 值基准库；然后通过计算运行中系统文件或内存中系统内核组件的Hash值；最后比对两者来判断系统是否受Rootkit感染。应用此方法的检测工具有：Windows系统下的文件完整性检查工具Sentinel，Rutkowska所编写的SVV[33]、PatchFinder[60-61]等。该方法检测算法简单，能检测部分未知Rootkit，但因其Hash 值基准库易被Rootkit伪造或绕过，且必须考虑所有可能被渗透攻击的代码区域，致使该方法执行困难。

3) Hooking检测法

Hooking检测法通过监测系统地址空间中的内

核表格是否被钩挂来加以判断。该方法监测逻辑如下：首先，遍历系统中易被钩挂的重要内核表格(诸如SSDT、IAT、IDT、驱动程序IRP表等)，确保函数指针指向可信的系统模块的地址范围内；其次，对于每个表格项目，反汇编对应的系统服务函数的头几条指令，确保执行转移落在内核模块范围内。使用此方法的检测工具较多，如VICE[62]、RAID[63]、iDefense公司的Hook Explorer、Volatile Systems公司开发的Volatility、Mandiant公司的Intellignet Response及HBGary公司的Responder，均采用此类方法检测Rootkit。但该类方法易受深度Hooking技术和DKOM技术绕过和欺骗。

4) 视图差异检测法

差异检测法又称为交叉视图检测法，是通过比较不同途径(低层视图与高层视图)所枚举到的系统信息，如进程列表、文件目录列表、网络连接列表、SSDT、IDT等，根据所获结果的差异来检测Rootkit。文献[64]首先提出该检测方法并开发了相关工具Patchfinder。之后，利用该检测原理，功能更强的工具相继出现，应用范围逐渐扩大。如微软Russinovich编写的Rootkit Revealer、IceSword，Linxer编写的PCHunter及F-Secure公司研发的Blacklight等。文献[65]提出了检测虚拟Rootkit的视图差异法，该方法的前提是假设，系统信息在一个列表中出现，而在另一个列表中隐藏。因此，在可靠的枚举系统信息情形下，可检测已知或未知的Rootkit。但如果Rootkit修改了第二个列表，使两个列表看起来完全相同，则这种检测假设就不成立，导致不能成功检测。

3.2 硬件检测法

随着更低层的虚拟Rootkit和硬件Rootkit出现，它们先于Rootkit检测软件甚至操作系统启动，导致基于软件的Rootkit检测方法无能为力。这样，基于硬件的Rootkit检测法或许是一种颇有前景的防御方法。

目前，基于硬件的Rootkit检测工具相对较少。美国国防高级研究计划局和国家安全部联合研制了一个基于硬件的Rootkit解决方案CoPilot[66]，能在硬件级别上监控主机的内存和文件系统，通过实时扫描系统寻找异常行为。Tribble[67]及FRED[68]利用直接内存访问(direct memory access，DMA)指令去获取物理内存拷贝，并搜寻异常行为。在拷贝物理内存时，目标系统的CPU将暂停，以避免攻击者执行Rootkit从而非法篡改内存数据。

尽管基于硬件的Rootkit检测法是目前最好的防御方案，但仍没有相关商业产品，且特别编制的Rootkit仍可避开其检测取证。此外，对于硬件卡的更新升级也相对复杂。从这个意义上看，Rootkti检测防御研究依旧任重道远。

3.3 Rootkit检测工具评估

自从Rootkit及内含Rootkit技术的恶意软件不断涌现，研究人员及安全厂商也随之提出了很多检测方法及工具应对。为检验、评价相关检测方法与工具的有效性，开展Rootkit检测工具的评估研究具有重要的理论价值与实践意义[69]。目前，对于Rootkit检测工具的评估研究处于起步阶段，研究者开始对此展开相关研究。如文献[70-71]对部分免费的和商用的Rootkit检测工具进行了评估研究；文献[72]对部分开源Rootkit检测工具进行了评估研究。

根据Rootkit与操作系统交互所采用的相关技术，本文选择了10个有代表性的Rootkit样本，即AFX Rootkit2005、Hacker Defender、NTIllusion、Vanquish、Gromozon、FuTo、Ghost、Shadow Walker、Unreal及Phide_ex；选择了10个有代表性的Rootkit检测工具进行相关检测性能评估。通过以下14个参数[72]评估Rootkit检测工具的性能：进程/线程、加载模块、动态链接库、服务、文件、主引导记录MBR、交换数据流、注册表、SSDT钩子、IDT钩子、IRP钩子、IAT钩子、GDT/LDT钩子及SYSENTER钩子。测评结果如表1所示。

表1 Rootkit检测工具评估

评估参数	Rootkit检测工具									
	Strider GhostBuster	F-Secure BlackLight	Sophos Anti-Rootkit	GMER	IceSword	XueTr	Rootkit Revealer	McAfee Rootkit Detective	Rootkit Unhooker	Helios
Process/Thread	√	√	√	√	√	√		√	√	√
Modules		√		√	√	√		√	√	√
DLL		√		√	√	√		√		√
Services		√		√	√	√		√		√
Files	√	√	√	√	√	√	√	√	√	√
MBR				√						

(续表)

评估参数	Rootkit检测工具									
	Strider GhostBuster	F-Secure BlackLight	Sophos Anti-Rootkit	GMER	IceSword	XueTr	Rootkit Revealer	McAfee Rootkit Detective	Rootkit Unhooker	Helios
Registery	√		√	√	√	√	√	√		√
Alternate Data Stream				√			√			
SSDT Hooks		√		√	√	√		√	√	√
IDT Hooks		√		√	√	√		√	√	√
IRP Hooks		√		√	√	√		√	√	√
IAT Hooks		√		√	√	√		√	√	√
GDT/LDT Hooks						√				
SYSENTER Hooks	√	√		√		√		√	√	√

从测评结果可知，商业Rootkit检测工具(如GMER、McAfee Rootkit Detective、Helios等)表现普遍较好，部分免费或共享工具(如IceSword、XueTr等)也非常不错，个别免费工具检测结果有待进一步提高。

4 Rootkit发展趋势及防御

Rootkit技术自诞生起，一直遵循着与应用软件技术、操作系统技术和底层硬件技术同步发展的模式。应用软件技术、操作系统技术和底层硬件技术的升级与更新，都将促使Rootkit技术的演化与发展。同样地，从攻防博弈的角度，随着Rootkit技术日趋复杂和更臻完美，Rootkit防御技术也呈现“魔高一尺，道高一丈”的发展态势。依据上述分析和掌握的最新资料，将展望Rootkit技术未来发展趋势、探讨Rootkit防御技术进一步研究方向。

4.1 Rootkit发展趋势

Rootkit技术基本遵循与应用软件技术、操作系统技术和底层硬件技术同步发展与演化模式，已呈现日趋复杂、相互融合，并积极扩展至其他新平台、新领域的发展趋势[27]。

1) 从计算机系统纵向层次的角度，Rootkit技术正向纵深方向发展：由高层向低层，由用户层向内核层，由软件向硬件，由磁盘空间向内存空间。从这个意义来说，Rootkit正在充分利用计算机系统的层次模型，竭尽所能地发展新的隐遁技术[73]。

2) 从与其他恶意软件交互的横向视角，Rootkit技术已向相互渗透、相互融合方向发展，导致其日趋复杂和更臻完美。如2012年波及全球的Flame攻击便是一个典型实例：既有病毒的自我复制，又具蠕虫的漏洞利用，还有Rootkit的隐遁潜行功能。此类发展趋势，将会给Rootkit防御带来异常严峻的挑战[27]。

3) 从系统平台的角度，Rootkit已由Windows平台向智能终端系统平台发展。随着Android、iOS等智能终端操作系统的快速发展与普及，Rootkit正向该领域快速扩散与演化发展，开始影响移动互联网安全[74]。如Android平台上的FakeDebuggerd Rootkit，能隐蔽地窃取用户手机号、硬件编号、地理位置等信息；iOS平台上的Carrier IQ Rootkit，能将用户的一切动作(键盘输入、短信，甚至通话)隐蔽地传送至Carrier IQ公司。

4) 从应用领域的角度，Rootkit已开始由信息系统领域转移至与信息技术相关的新领域，且破坏威力惊人。譬如，在工业领域，伊朗核电站所遭受的Stuxnet攻击，就是Rootkit技术的新型应用；在金融领域，2010年著名黑客大会(Black Hat)上，文献[75]利用Rootkit技术攻击ATM机。近年来盛行的隐遁定向APT攻击[76]，就是Rootkit隐遁技术的典型应用。

5) 从反取证的角度，Rootkit已初步具备在诸如数据销毁、数据隐藏、数据转换、数据伪造和数据源擦除等[27]方面的反取证能力，这将给实时取证带来极大挑战。

4.2 Rootkit防御方向

从攻防博弈的视角，Rootkit技术与Rootkit防御技术的魔道之争将会相互促进并一直持续下去。综合上述讨论与分析，本文认为Rootkit防御技术领域的进一步研究方向包括：

1) 硬件级防御技术[65,77]。以系统启动顺序和与操作系统交互的视角，Rootkit会竭尽所能做到最先启动以获取系统控制权限。因此，从防御的角度，硬件级Rootkit检测防御方案通过先于操作系统加载，先期获取控制权限，占据先发制人的技术优势，无疑最具发展应用前景。

2) 操作系统级防御技术[78-79]。从操作系统内核

的视角，造成目前Rootkit泛滥的主要原因在于，当前操作系统缺乏有效区分与隔离自身模块和外来驱动程序的相关机制。一旦Rootkit加载入系统内核，就拥有与系统内核相同的特权，势必造成检测与清除困境。因此，操作系统如能有效区分与隔离自身模块与外来驱动程序，无疑能有效拦截Rootkit。如Microsoft 公 司 的 64 位 Windows 系 统 引 入 的PatchGuard技术已具备相关拦截功能。

3) 内存检测取证技术[80]。从Rootkit进程运行的视角，每个Rootkit进程都需加载至内存中运行，因此，内存无疑是检测防御Rootkit最佳位置。如何有效获取内存数据、分析内存数据、并检测内存中运行的Rootkit，将是未来Rootkit防御技术的进一步研究方向。

4) 免疫云智能防御技术[81-82]。从系统和全局的视角，将计算机系统视为人体系统，Rootkit防御系统等同于人体免疫系统。借鉴人体免疫系统机理和利用云计算的高效性，研究Rootkit免疫云智能防御系统的体系结构及互动机制，能有效防御Rootkit攻击，将是未来Rootkit智能防御技术进一步的研究方向。

5 结 束 语

Rootkit技术的隐遁潜行功能以及与其他恶意软件相互渗透、相互融合的演化发展趋势，已对网络信息系统造成了极大安全威胁。Rootkit技术及其防御技术，已是信息安全社区讨论与关注的热点研究领域。本文从技术的角度探讨了Windows系统的Rootkit技术原理、技术演化过程、Rootkit检测防御方法，并在此基础上展望了Rootkit未来发展趋势和Rootkit防御技术进一步研究方向。从本质上分析，Rootkit技术沿用的是攻防博弈的对抗性思维，充分利用Windows系统的层次模型，通过修改系统内核结构或更改指令执行流程，从而实现其隐遁功能。因此，Rootkit及其防御技术将随着应用软件技术、操作系统技术和底层硬件技术的演化而不断发展与更新。

参 考 文 献

[1] STONE R. A call to cyber arms[J]. Science, 2013, 339(6123): 1026-1027.

[2] Symantec Corporation. 2013 Norton report[EB/OL]. [2014-12-10]. http://www.yle.fi/tvuutiset/uutiset/upics/liitetiedostot/norton_raportti.pdf.

[3] IBM Corporation. IBM X-Force 2013 annual trends and risk report[EB/OL]. [2014-12-10]. http://www-03.ibm.com/security/ xforce/.

[4] RIES C. Inside Windows Rootkits[EB/OL]. [2014-12-10]. http://read.pudn.com/downloads64/sourcecode/windows/freedic/226557/Inside%20Windows%20Rootkits.pdf.

[5] Symantec Corporation. Windows rootkit overview[EB/OL]. [2014-12-10]. http://www.symantec.com/avcenter/reference/windows.rootkit.overview.pdf.

[6] HOGLUND G, BUTLER J. Rootkits: Subverting the Windows kernel[M]. USA: Addison-Wesley Professional, 2007.

[7] Mandiant Corporation. APT1: Exposing one of China's cyber espionage units[EB/OL]. [2014-12-10]. http://intelreport. mandiant.com/Mandiant_APT1_Report.pdf.

[8] McAfee Labs. 2014 threats predictions[EB/OL]. [2014-12-10]. http://www.mcafee.com/uk/resources/reports/rp-threats-predictions-2014.pdf, 2014.

[9] HOGLUND G. A real NT rootkit. PHRACK[EB/OL]. [2014-12-10]. http://phrack.org/issues.html?issue=55&id=5#article.

[10] RUSSINOVICH M. Sony, Rootkits and digital rights management gone too far[EB/OL]. [2014-12-10]. http://blogs.technet.com/b/markrussinovich/archive/2005/10/31/sony-rootkits-and-digital-rights-management-gone-too-far.aspx.

[11] KUSHNER D. The real story of stuxnet[EB/OL]. [2014-12-10]. http://spectrum.ieee.org/telecom/security/the-real-story-of-stu xnet.

[12] BRAUN S, FLAHERTY A, GILLUM J, et al. Secret to PRISM program: Even bigger data seizures[EB/OL]. [2013-06-15]. http://bigstory.ap.org/article/secret-prism-success-even-bigger-data-seizure.

[13] 潘剑锋, 奚宏生, 谭小彬. 一种利用程序行为分析的Rootkit 异常检测方法[J].中国科学技术大学学报, 2010, 40(8): 863-869.
PAN Jian-feng, XI Hong-sheng, TAN Xiao-bin. A method for rootkit anomaly detection using behaviors analysis[J]. Journal of Univeristy of Science and Technology of China, 2010, 40(8): 863-869.

[14] 薛英飞. 基于Windows(2000/2003)内核对象的Rootkit检测[D]. 上海: 上海交通大学, 2008.
XUE Ying-fei. Rootkit detection based on Windows(2000/2003) kernel object[D]. Shanghai: Shanghai Jiao Tong University, 2008.

[15] 徐昊. Win32平台下内核Rootkit技术的研究与应用[D]. 上海: 上海交通大学, 2007.
XU Hao. Research and application of kernel Rootkit technology under Win32 environment[D]. Shanghai: Shanghai Jiao Tong University, 2007.

[16] 赖云一. Windows Rootkit分析与检测[D]. 成都: 电子科技大学, 2009.
LAI Yun-yi. Windows rootkit analysis and detection[D]. Chengdu: Univeristy of Electronic Science and Technology of China, 2009.

[17] 何志. 针对Windows RootKit的安全监测系统的研究与实现[D]. 成都: 电子科技大学, 2008.
HE Zhi. Research and implementation of Windows Rootkit secure detection system[D]. Chengdu: Univeristy of Electronic Science and Technology of China, 2008.

[18] 双世勇. Windows Rootkit检测方法研究[D]. 郑州: 中国人民解放军信息工程大学, 2005.
SHUANG Shi-yong. Research on Windows Rootkit detection methods[D]. Zhengzhou: The PLA Information Engineering University, 2005.
[19] 薛寒. 网络主动防御系统中的rootkit检测与个人防火墙[D]. 郑州: 中国人民解放军信息工程大学, 2007.
XUE Han. Rootkit detection and personal firewall in the network active defense system[D]. Zhengzhou: The PLA Information Engineering University, 2007.
[20] 白光冬, 郭耀, 陈向群. 一种基于交叉视图的Windows Rootkit检测方法[J]. 计算机科学, 2009, 36(8): 133-137.
BAI Guang-dong, GUO Yao, CHEN Xiang-qun. Windows rootkit detection method based on cross-view[J]. Computer Science, 2009, 36(8): 133-137.
[21] BRAVO P. GARCIA D F. Rootkits survey: a concealment story[EB/OL]. [2014-12-10]. http://www.pablobravo.com/files/survey.pdf.
[22] JOY J, JOHN A, JOY J. Rootkit detection mechanism: a survey[J]. Communications in Computer and Information Science, 2011, 203: 366-374.
[23] SHIELDS T. Survey of Rootkit technologies and their impact on digital forensics[EB/OL]. [2014-12-10]. http://www.donkeyonawaffle.org/misc/txs-rootkits_and_digital_forensics.pdf.
[24] 李文新, 王姜博, 慕德俊, 等. Android 系统Rootkit 技术综述[J]. 微处理机, 2011, 32(2): 68-72.
LI Wen-xin, WANG Jiang-bo, MU De-jun, et al. Survey on Android Rootkit[J]. Microprocessors, 2011, 32(2): 68-72.
[25] SHEVCHENKO A. Rootkit evolution[EB/OL]. [2014-12-10]. http://www.securelist.com/en/analysis?pubid=204792016.
[26] DAVIS M A, BODMER S M, LEMASTER A. Hacking explosed: Malware & Rootkits secrets & solutions[M]. USA: The McGraw-Hill Companies, 2010.
[27] BLUNDEN B. The Rootkit arsenal: Evasion in the dark corners of the system[M]. Massachusetts: Jones & Bartlett Publishers, 2013.
[28] BRAVO P, GARCIA D F. Proactive detection of kernel-mode Rootkits[C]//2011 Sixth International Conference on Availability, Reliability and Security. Vienna: IEEE Computer Society, 2011: 515-520.
[29] SPARKS S, EMBLETON S, ZOU C. Windows Rootkits a game of “hide and seek”[EB/OL]. [2014-12-10]. http://www.cs.ucf.edu/~czou/research/Rootkit-BookChapter.pdf.
[30] BUTLER J, HOGLUND G. VICE: Catch the hookers[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-04/bh-us-04-butler/bh-us-04-butler.pdf.
[31] SILBERMAN P, CHAOS. FUTo Rootkit[EB/OL]. [2014-12-10]. http://uninformed.org/?v=3&a=7&t=sumry.
[32] KAPOOR A, MATHUR R. Predicting the future of stealth attacks[EB/OL]. [2014-12-10]. http://www.mcafee.com/us/resources/reports/rp-predicting-stealth-attacks.pdf.
[33] MATROSOV A, RODIONOV E. TDL3: the Rootkit of all evil?[EB/OL]. [2014-12-10]. http://www.eset.com/us/resources/ white-papers/TDL3-Analysis.pdf.
[34] RUTKOWSKA J. System virginity verifier: Defining the roadmap for malware detection on windows systems [EB/OL]. [2014-12-10]. http://www.invisiblethings.org/papers/ hitb05_virginity_verifier.ppt.
[35] RUTKOWSKA J. Rootkit hunting vs compromise detection[EB/OL]. [2014-12-10]. http://www. blackhat.com/presentations/bh-federal-06/BH-Fed-06-Rutkowska/BH-Fed-06-Rutkowska-up.pdf.
[36] SPARKS S, BUTLER J. Shadow walker: Raising the bar for windows rootkit detection [J]. Phrack Magazine, 2005, 11(63): 10-26.
[37] FIELD S. An introduction to kernel patch protection [EB/OL]. [2014-12-10]. http://blogs.msdn.com/b/windowsvistasecurity/archive/2006/08/11/695993.aspx.
[38] RUTKOWSKAJ. Introducing stealth malware taxonomy [EB/OL]. [2014-12-10]. www.net-security.org/dl/articles/malware-taxonomy.pdf.
[39] KING S T, CHEN P M, WANG Y M. SubVirt: Implementing malware with virtual machines[C]//2006 IEEE Symposium on Security and Privacy. Berkeley: IEEE Computer Society, 2006: 314-327.
[40] RUTKOWSKA J. Subverting vista kernel for fun and profit[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Rutkowska.pdf.
[41] DAIZOVI D A. Hardware virtualization rootkits[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Zovi.pdf.
[42] PTACEK T, LAWSON N. Don’t tell Joanna, the virtualized rootkit is dead[EB/OL]. [2014-12-10]. http:// matasano.com/research/bh-usa-07-ptacek_goldsmith_and_lawson.pdf.
[43] BULYGIN Y. Insane detection of insane rootkits: Chipset based approach to detect virtualization malware[EB/OL]. [2014-12-10]. http://me.bios.io/images/2/23/DeepWatch.pdf.
[44] RUTKOWSKA J, TERESHKIN A. IsGameOver() anyone? [EB/OL]. [2014-12-10]. http://invisiblethingslab.com/resources/bh07/IsGameOver.pdf.
[45] SOEDER D, PERMEH R. eEye BootRoot[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-05/bh-us-05-soeder.pdf.
[46] COLLAPSE C. A real SMM rootkit[J]. Phrack Magazine, 2009, 13(42): 56-68.
[47] BSDAEMON, COIDELOKO, DONANDON. System management mode hacks[J]. Phrack Magazine, 2008, 12(41):12-25.
[48] EMBLETON S, SPARKS S, ZOU C. SMM Rootkits: a new breed of OS independent malware[J]. Security & Communication Networks, 2013, 6(12): 1590-1605.
[49] ICELORD. BIOS Rootkit: Welcome home, my lord [EB/OL]. [2014-12-10]. http://blog.csdn.net/icelord/article/details/1604884.
[50] HEASMAN J. Implementing and detecting an ACPI BIOS rootkit[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Heasman.pdf.
[51] KUMAR N, KUMAR V. VbootKit: Compromising Windows vista security[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-europe-07/Kumar/Whitepaper/bh-eu-07-Kumar-WP-apr19.pdf.

[52] KUMAR N, KUMAR V. VBootKit 2.0-attacking Windows 7 via boot sectors[EB/OL]. [2014-12-10]. http://www.securitybyte.org/2009/schedule/Day1_Orchid/Vbootkit2.0a-AttackingWindows7viaBootSectors.pdf.

[53] METULA E. Managed code Rootkits[M]. Burlington: Syngress press, 2011.

[54] RUSSINOVICH M, SOLOMON D, IONESCU A. Windows internals[M]. USA: Microsoft Press, 2012.

[55] 潘爱民. Windows内核原理与实现[M]. 北京: 电子工业出版社, 2010.
PAN Ai-min. Understanding the Windows kernel[M]. Beijing: Publishing House of Electronics Industry, 2010.

[56] BILBY D. Low down and dirty: Anti-forensic rootkits[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-jp-06/BH-JP-06-Bilby-up.pdf.

[57] BUTLER J, ARBAUGH B, PETRONI N. R^2: the exponential growth of Rootkit techniques[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Butler.pdf.

[58] BLUNDEN B. Anti-forensics: the Rootkit connection[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-09/BLUNDEN/BHUSA09-Blunden-AntiForensics-PAPER.pdf.

[59] SUN Bing. Software virtualization based Rootkits[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-europe-07/Bing/Whitepaper/bh-eu-07-bing-WP.pdf.

[60] RUTKOWSKA J. Detecting Windows server compromises with patchfinder 2[EB/OL]. [2014-12-10]. http://jaumingtw.sg1005.myweb.hinet.net/av/anti-rootkits/patch_finder/rootkits_detection_with_patchfinder2.pdf.

[61] RUTKOWSKA J. Execution path analysis: Finding kernel based Rootkits[J]. Phrack Magazine, 2003, 11(59): 65-79.

[62] BUTLER J, HOGLUND G. VICE: Catch the hookers: Plus new Rootkit techniques[EB/OL]. [2014-12-10]. http://www.cs.dartmouth.edu/~sergey/cs258/rootkits/bh-us-04-butler.pdf.

[63] BUTLER J, SIBERMAN P. RAIDE: Rootkit analysis identification elimination v1.0[EB/OL]. [2014-12-10]. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Silberman.pdf.

[64] RUTKOWSKA J. Thoughts about crossview based rootkit detection[EB/OL]. [2014-12-10]. http://invisiblethings. org.

[65] XIE Xiong-wei, WANG Wei-chao. Rootkit detection on virtual machines through deep information extraction at hypervisor-level[C]//IEEE Conference on Communications and Network Security. National Harbor, USA: IEEE, 2013, 498- 503.

[66] PETRONI N L, FRASER T, MOLINA J, et al. Copilot - a coprocessor-based kernel runtime integrity monitor[EB/OL]. [2014-12-10]. http://www.umiacs.umd.edu/partnerships/ltsdocs2004/Active_Systems(Arbaugh).pdf.

[67] CARRIER B D, GRAND J. A hardware-based memory acquisition procedure for digital investigations[J]. Journal of Digital Investigation, 2004, 1(1):50-60.

[68] BBN Technologies. Fred: Forensic ram extraction device [EB/OL]. [2012-12-10]. http://www.ir.bbn.com/vkawadia/, 2014.

[69] RILEY R. A framework for prototyping and testing data-only rootkit attacks[J]. Computers & Security, 2013, 37: 62-71.

[70] ASHRAF. 13 top best free Rootkit removal(anti-rootkit) programs[EB/OL]. [2014-12-10]. http://dottech.org/129897/13-top-best-free-rootkit-removal-anti-rootkit-programs-windows-xp-vista-7-8/.

[71] CARVEY H. Windows forensic analysis[M]. New York : Syngress Publisher, 2009.

[72] ROMANA S, JHA A K, PAREEK H, ESWARI P R. Evaluation of open source anti-rootkit tools[C]//Workshop on Anti-malware Testing Research. Montreal: IEEE, 2013: 1-6.

[73] HILI G, MAYES K, MARKANTONAKIS K. The BIOS and Rootkits, secure smart embedded devices[J]. Platforms and Applications, 2014, 369-381.

[74] PU Shi, CHEN Zhou-guo, HUANG Chen, et al. Threat Analysis of smart mobile device[C]//2014 General Assembly and Scientific Symposium. Beijing: IEEE, 2014: 1-3.

[75] JACK B. Jackpotting automated teller machines redux [EB/OL]. [2014-12-10]. http://www.blackhat.com/html/bh-us-10/bh-us-10-briefings.html.

[76] McAfee Corporation. Combating advanced persistent threats [EB/OL]. [2014-12-10]. http://www.mcafee.com/us/resources/white-papers/wp-combat-advanced-persist-threats.pdf.

[77] 施江勇, 王会梅, 鲜明, 等. 硬件虚拟化Rootkit检测方法研究综述[J]. 计算机应用研究, 2014, 31(1):1-5.
SHI Jiang-yong, WANG Hui-mei, XIAN Ming, et al. Summarize of detection methods on hardware-based virtualization machine Rootkit[J]. Application Research of Computers, 2014, 31(1): 1-5.

[78] 辛知, 陈惠宇, 韩浩, 等. 基于结构体随机化的内核Rootkit防御技术[J]. 计算机学报, 2014, 37(5): 1100-1110.
XIN Zhi, CHEN Hui-yu, HAN Hao, et al. Kernel Rootkit defense based on automatic data structure randomization [J]. Chinese Journal of Computers, 2014, 37(5): 1100-1110.

[79] WANG Xue-yang, KARRI R. Detecting kernel control-flow modifying Rootkits[J]. Advances in Information Security, 2014, 55: 177-187.

[80] KORKIN I, NESTEROV I. Applying memory forensics to rootkit detection[C]//Proceedings of the Conference on Digital Forensics, Security and Law. Virginia, USA: ADFSL, 2014: 115-143.

[81] LIANG Jun-jie. Key security technologies of cloud computing platforms[J]. Advances in Intelligent Systems and Computing, 2014, 250: 411-417.

[82] CARRETERO J, BLAS J G. Introduction to cloud computing: Platforms and solutions[J]. Cluster Computing, 2014, 17(4): 1225-1229.

编 辑 蒋 晓