基于虚拟机的Rootkit检测系统

张文晓,戴 航,黄东旭
(西北工业大学 自动化学院,陕西 西安 710072)

摘 要:内核级Rootkit位于操作系统核心层,可以篡改内核地址空间的任意数据,对系统安全构成了巨大的威胁。目前基于虚拟机的Rootkit方面应用大都偏重于完整性保护,未对Rootkit的攻击手段和方式进行检测识别。文中在虚拟机框架下,提出了一种新型的Rootkit检测系统VDR,VDR通过行为分析可有效识别Rootkit的攻击位置方式,并自我更新免疫该Rootkit的再次攻击。实验表明,VDR对已知Rootkit的检测和未知Rootkit的识别均有良好效果,能迅速给出攻击信息,为系统安全管理带来很大方便。

关键词:Rootkit;虚拟机;内核;特征码
中图分类号:TP319 **文献标识码**:A **文章编号**:1673-629X(2012)07-0128-04

A New Rootkit Detection System Based on Virtual Machine

ZHANG Wen-xiao, DAI Hang, HUANG Dong-xu
(College of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: Kernel Rootkit runs in the highest system level, can modify all the data of system, so it causes great threat to the security of computer system. At present, facing to the Rootkit, most of methods based on virtual machine focus on protection of kernel's integrity, and ignore to detect the technology of Rootkit. Based on virtual machine, propose a new method to automatically detect and sort Rootkit. This method is named VDR system, can detect Rootkit efficiently and tell the difference between kinds of Rootkit, moreover remember it for agenst the second attack. The VDR system can improve plentiful information for the system administrator.

Key words: Rootkit; virtual machine; kernel; key code

0 引 言

随着信息浪潮的全球化,计算机与网络已成为各国用于管理和传输社会经济、政治、军事和外交等信息的重要工具。信息化在为人们提供便利的同时,也带来了一系列信息安全问题。

Rootkit是一套给入侵者提供后门,收集同一网络上其他主机信息并能掩盖系统已经被入侵事实的工具集合[1]。Rootkit主要分为用户级和内核级两类。用户级Rootkit通常替换或修改一些系统工具,比如/bin/login, /bin/ps等,已被替换的系统文件提供后门给攻击者进行攻击,同时隐藏攻击者的痕迹;而内核级Rootkit运行在操作系统的内核空间,通过劫持系统调用、拦截中断、攻击函数等手段,达到隐藏文件、进程、网络连接,记录密码等非法目的。由于内核Rootkit具有最高的硬件特权级,可以篡改内核地址空间的任意数据,因此对系统安全构成了巨大的威胁。

虚拟机技术近些年来发展迅速,在安全技术方面有较好的前景,但是当前虚拟机在反Rootkit方面的应用,多数集中在通过虚拟机层的高权限对客户系统进行完整性保护[2,3],这种单纯的防守过于粗旷,虽然在一定意义上实现了反Rootkit的效果,但并未对Rootkit的技术手段和攻击目标进行检查,使系统管理员无从得知所受到的攻击状况,甚至于不知道曾经受到攻击。

文中在虚拟机框架下针对内核级Rootkit提出了一种新型的Rootkit检测系统VDR (Rootkit Detection system based on Virtual machine)。VDR利用虚拟机监控器,对客户机操作系统进行监控,通过建立内核代码与攻击手段的映射关系,可以快速检测、定位恶意代码的攻击位置和方式,并通过行为分析方法辨别是否为Rootkit,最后采用类人工免疫的方式进行自我反馈修正,抵抗该Rootkit的再次攻击。实验证明该系统对已知Rootkit的检测和未知Rootkit的识别均有良好效果,能迅速给出攻击的相关信息,为系统管理带来方便。

1 Rootkit检测技术分析

内核级Rootkit主要攻击方法有如下几种[4]:

收稿日期:2011-12-15;**修回日期**:2012-03-20
基金项目:国家自然科学基金资助项目(60803158)
作者简介:张文晓(1987-),男,硕士生,研究方向为网络与信息安全;戴 航,副教授,研究方向为网络控制、网络信息安全。

(1)攻击系统调用函数。

(2)攻击系统调用表。

(3)攻击中断描述符表。

(4)攻击系统调用入口函数。

针对内核级的 Rootkit,近年来提出了许多有针对性的方法,比如:Kern_check[5]将 System. map 文件中的内核符号及其虚拟地址与内存中内核空间对应内核符号及其地址进行比较,如果发现有区别,则认为存在内核态 Rootkit。Checkidt[6]主要是针对中断描述符表进行完整性来检测,从而检测修改中断描述符表这一类型内核态 Rootkit 的存在。St. Michael[7]的方法主要是验证内核关键区域如系统调用表的完整性来达到检测目的。

上述检测工具或方法虽然能够从不同方面针对性地检测到一些 Rootkit,但这些方法依赖于某些关键数据[8],若 Rootkit 针对这些数据进行恶意篡改,那么这类检测方法将会失效。究其原因主要有两点:

(1)无法获取比内核态更高的权限;

(2)无法保证检验基准的安全性,检测方法主要是通过创建检验基准来检测 Rootkit 的存在,这依赖于检验基准和检查器本身的安全。

针对于这个原因,文献[9,10]尝试通过修改系统调用将检测器和检测器的基准备份隐藏并且加密,尽管带来了一定程度的安全,但远远不够, Rootkit 同样可能发现并修改这些备份文件,因为 Rootkit 入侵系统后同样获取了系统的最高权限,系统中的一切对 Rootkit 来说是透明的。

文献[11,12]提出虚拟机来对检验基准和检测器进行保护,如图 1 所示:

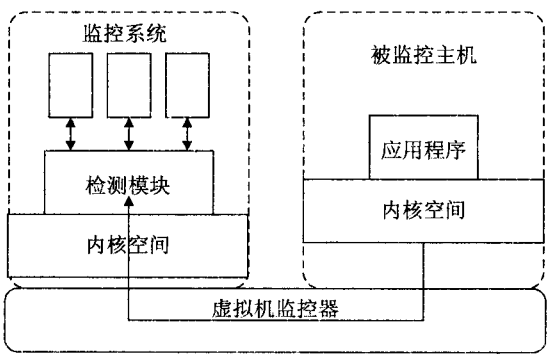

图 1　虚拟机监控器的系统架构

由于 VMM (Virtual Machine Monitor)支持运行在 VMM 上的多个虚拟机(Virtual Machine,VM)间的隔离,运行在虚拟机上的软件不能对运行在 VMM 或其它虚拟机上的软件进行访问或修改,因此即使 Rootkit 已将被监控主机完全控制,监测系统中的数据仍然不会遭到篡改。目前,虚拟机在反 Rootkit 方面有了较多的研究应用,但大都局限于利用虚拟机进行完整性保护[2,3],虽然在一定程度上实现了反 Rootkit 的效果,但并未对 Rootkit 的技术手段和攻击目标进行有效检查和区分,使系统管理员无从得知所受到的攻击状况,不能有针对性的对系统防御进行升级。

2　VDR 系统的设计与实现

一般来说,安全问题可以分为三个方面:预防、检测和恢复[11]。当前基于虚拟机的反 Rootkit 应用在完整性保护方面已经有了相对成熟的方法,但完整性保护只是被动防御,不能实时发现攻击行为。

因此不能单纯靠防范保证系统免受攻击,尽早发现攻击行为与攻击手段,对系统的防护、安全策略的制定,以及防御体系的升级都具有重大的意义。

文中提出一种基于虚拟机的 Rootkit 检测系统 VDR,VDR 可有效定位 Rootkit 的攻击位置和篡改手段,预估带来的风险。并通过机器学习的方法对 Rootkit 的二次攻击进行免疫防御。

VDR 系统的优点在于可以对操作系统进行实时监控,能够尽早地发现攻击或非法行为,并通知用户或者依据一定的安全规则进行自动控制。这对减少损失、增强系统安全具有重要的作用。VDR 系统如图 2 所示:

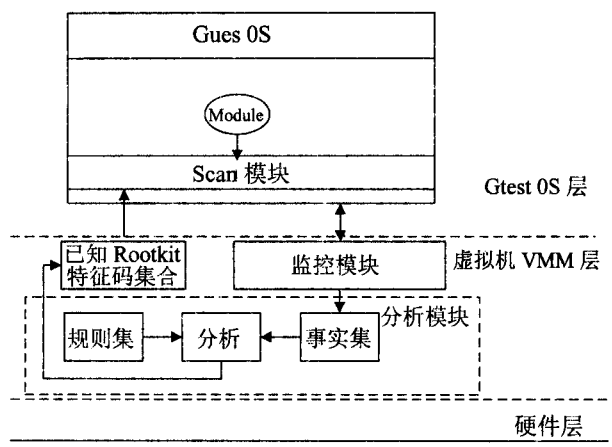

图 2　VDR 系统

系统主要分为三个模块:SCAN 模块,监控模块和分析模块。

SCAN 模块部署在虚拟机内部,这样做的优点是可以直接调用系统内部丰富的功能函数和数据结构,快速有效地在模块加载前对其扫描,扫描模块依赖于虚拟机层的完备的 Rootkit 特征库。只有扫描通过,模块才能在监控模块的监控下继续加载。

监控模块位于虚拟机层次,可以充分利用虚拟机的优势,对模块的行为进行监控并记录,由于虚拟机位于比 Guest OS 更高的层次,所以保证了监控结果的准确可靠。

分析模块根据监控模块的记录结果对加载模块的行为进行解读,并评估所带来的危害风险,如果正常则通过,如果经过分析识别为 Rootkit,则提取该 Rootkit 的特征码加入特征码集合,以抵御该 Rootkit 的再次攻击。

2.1 SCAN 模块

对已知 Rootkit 的检测通过建立完备的 Rootkit 特征库,对加载模块进行扫描的方式来检测。

特征码的生成目前主要有下面两种方式:特征序列方式和信息摘要方式。特征码序列通常是手工分析恶意代码提取特征序列,根据对几家反病毒公司的调查[13],一个熟练的病毒分析师平均每天能够提取 12.8 条病毒特征码。由于 VDR 系统对特征码的提取时间要求较短,因而这种方式不符合需求。

而对于信息摘要方式,通常使用的是 MD5 格式特征码。VDR 系统使用 MD5 作为特征码来匹配 Rootkit,这是一种快速有效而且值得信赖的方法,相比人工分析提取序列特征码,MD5 可做到迅速准确的自动完成,有效是因为现在大部分 Rootkit 都是非感染型的,发放出去后不会变化或者变化有限;值得信赖是因为 MD5 误报正常文件的几率可以忽略不计。

(1)特征码的结构。

SCAN 模块根据 MD5 的匹配情况给出结果和 Rootkit 的相关攻击信息。为系统管理员提供全面充分的攻击信息。特征码结构如下:

名称	MD5	Rootkit 相关攻击信息

(2)特征库的初始建立。

搜集目前已知的所有 Rootkit 标本,然后在 Linux 下编写脚本调用特征码生成模块提取特征码,当然也可以直接在系统下运行 Rootkit 标本,通过 VDR 系统自动来完成已知 Rootkit 的特征生成。

(3)特征库的更新。

分析模块根据监控系统提供的事实集进行分析,如果认定为 Rootkit,则调用特征码生成模块提取特征加入到已知 Rootkit 特征集合中,如图 3 所示。

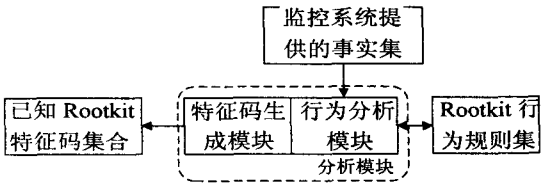

图 3 特征库更新方式

2.2 监控模块

根据前面对 Linux 内核以及 Rootkit 技术的分析可以知道,Linux 系统中有如下关键资源[14]:内核代码(包括基本内核代码和内核模块的代码)、系统调用表和中断向量表、内核函数指针。

如果能够监控这些关键资源不被篡改,就能保证内核的控制流完整性,从而防止绝大多数内核 Rootkit 的攻击。CR0 控制寄存器是 X86CPU 中重要的控制寄存器,它的第 16 位是写保护位,传统的 Linux 内核都利用这种机制对重要的内核数据结构进行写保护,比如系统调用表等。Rootkit 通常先修改寄存器来取消掉写保护,然后再进行内核篡改。

在虚拟机中 CR0 这些寄存器属于敏感寄存器[13],任何对这些特殊寄存器的访问需要陷入到虚拟机监视器中才能执行。

VDR 系统正是基于这一特性,在虚拟机监视器中对某个虚拟机的内核关键资源段的内存进行写保护,那么每当 Guest OS 中的 Rootkit 试图对内存内容进行修改时都会触发页面异常,而异常处理会陷入到虚拟机监视器中,在异常处理函数中设置检测代码便可以监控到 Rootkit 试图对内核的哪些关键资源进行篡改,如图 4 所示。

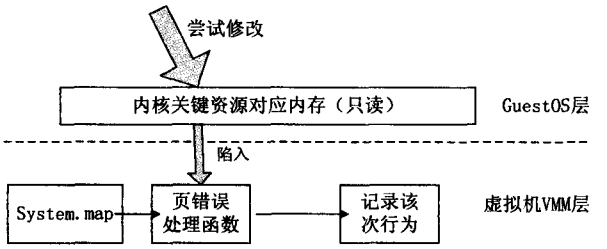

图 4 监控模块实现方式

具体实现步骤:客户机内核编译后生成 System.map 文件,从这个文件中能得到所有内核符号的地址。因此通过监控内核资源关键内存的更改可以达到监控 Rootkit 行为的目的。通过检测模块可以监控内核关键资源的状态变化,从而为接下来的处理模块对 Rootkit 的分析判别提供依据。

2.3 处理模块

对于危险行为的处理如下,通过对行为规则的定义,获取行为的危险系数,然后再通过对一系列模块行为危险系数的计算来定义整个加载模块的危险系数。最后通过一个阈值来得出判断,如果大于该阈值,则认定为 Rootkit,同时利用特征码提取算法提取加载模块的特征码并加入 Rootkit 特征码集合,否则正常返回。

定义行为有如下两种:

1、危险行为:修改寄存器,尝试修改内核关键资源,以及敏感系统调用。

通过对相应的 Rootkit 技术与源码分析,可以知道 Linux 下敏感系统调用,见表 1。

2、异常行为:非敏感系统调用,修改内核其他代码部分。

假定一次监控结果有如下行为:

$\{x_1\cdots x_i\cdots y_1\cdots y_j\cdots\}$,其中 x_i 对应为一次危险行为,y_j 对应为一次异常行为,a_i 为相应的行为判定因子,那么整体危险系数 T 的计算公式如下:

$$T=\frac{\sum_1^n(x_ia_i)}{\sum_1^n x_i}+\frac{\sum_1^n(y_ja_j)}{\sum_1^n y_j}$$

表 1　Linux 系统下敏感系统调用

getuid	setuid	sys_chmod
sys_chown	sys_fchmod	sys_fchown
sys_chroot	sys_execve	sys_init_module
sys_fcnt	sys_open	sys_delete_module
sys_fork	sys_rename	

行为判定因子的界定依赖于对大量 Rootkit 行为的统计分析,通过对一定数量的 Rootkit 抽样分析,对行为判断因子界定一个近似值:对于危险行为,危险因子为 1;对于异常行为,非敏感系统调用的行为因子为 0.6,内核其他代码部分为 0.3。随着检测系统收集的 Rootkit 行为的增加,这一因子将不断趋近于准确值。

对于 $T\geqslant 1$,判定为发现 Rootkit 攻击行为,提取相关信息填入特征结构体中,并更新特征库。

对于 $T<1$,判定为正常范围。

由公式可知,存在危险行为即可判断为 Rootkit,但存在异常行为则不能判断为 Rootkit,只有当积累到一定数量的异常行为时,才认为存在 Rootkit 入侵。

3　实　验

测试环境是 Xen 3.3.0[15] 和 Redhat 9 的 x86 平台。

(1)为了验证 VDR 系统对已知 Rootkit 的检测效果,采集了目前比较常见的 36 种 Rootkit 样本,根据 2.1 中所提到的方法生成初始特征库。现在特征库中有 36 条记录,选取其中 10 个 Rootkit 在客户操作系统中运行,运行结果如表 2 所示。

表 2　已知 Rootkit 扫描实验结果

Rootkit	linspy2	modhide	kis0.9	suckit	Adore
结果	√	√	√	√	√
Rootkit	all-root	maxty	kbdv3	rial	overide
结果	√	√	√	√	√

实验中 SCAN 模块迅速检测到并阻止了 Rootkit 的运行,并给出相应的警告信息(限于篇幅该信息未全部列出)。

(2)对于未知 Rootkit 的检测效果,选择之前没有加入特征库的 Knark 样本在客户系统中运行,结果如图 5 所示。

```
**********************************************************
                *    VDR    *
**********************************************************
2011-1201-15:33:42 warning:Rootkit
                   name:unknown
                   behavior:sys_getdents,sys_kill,sys_read,
sys_ioctl,sys_fork,sys_close,sys_execve,sys_settimeofday
```

图 5　未知 Rootkit 检测实验结果

监控结果显示 Knark 尝试修改 sys_getdents, sys_kill, sys_read, sys_ioctl, sys_fork, sys_clone, sys_execve, sys_settimeofday 等八个系统调用,处理模块判定为 Rootkit 行为并予以拒绝。同时再次安装 Knark,SCAN 模块直接给出了 Rootkit 警告。实验证明该检测器对未知 Rootkit 有相应的防御能力,并可通过自学习,免疫再次 Rootkit 攻击。

(3)对于异常行为的检测,编写了一个简单 LKM 模块,在 sys_select 系统调用中加入一行无意义的代码来模拟异常行为,然后在 Guset 系统中加载该模块,监控模块并未给出 Rootkit 警告。再次对该 LKM 修改,以同样的方式在 sys_poll 中加入无意义代码,该次加载模块后,监控结果给出 Rootkit 行为警告并予以拒绝。试验结果与 2.3 节符合。

4　结束语

文中分析了当前 Rootkit 的检测技术以及相应的局限性,在此基础上引出问题,并提出解决方案 VDR 系统,通过在 Linux 上的实验验证,VDR 系统可以快速检测、定位 Rootkit 的攻击位置和方式,并予以阻拦,同时采用类人工免疫的方式进行自我免疫修正,抵抗该 Rootkit 的再次攻击。对已知 Rootkit 的检测和未知 Rootkit 的识别均有良好的效果,并能迅速给出攻击的相关信息,为系统安全管理带来很大方便。

参考文献:

[1] NSA. Information Security Terms Glossary[S/OL]. 2005. https://www.key.com/html/bank-information-security-glossary.html.

[2] Kruegel C, Robertson W, Vigna G. Detecting kernel-level Rootkits through binary analysis[C]//Proc of the 20th Annual Computer Security Applications Conference. Washington D C: IEEE Computer Society, 2004: 91-100.

[3] Seshadri A, Luk M, Qu N, et al. SecVisor: A Tiny Hypervisor to Provide Lifetime Kernel Code Integrity for Commodity OSes [C]//Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07). [s. l.]: [s. n.], 2007: 335-350.

[4] 石晶翔,陈蜀宇,黄晗辉. 基于系统调用的内核级 Rootkit 技术研究[J]. 计算机技术与发展,2010,20(7):175-178.

[5] Wichmann R. kern_check[CP/OL]. 2006. http://www.la-samhna.de/library/kern_check.c.

[6] Kad. checkidt[CP/OL]. 2007. http://www.phrack.com/archives/59/p59_0x04_Handling%20the%20Interrupt%20Desc.

[7] Branco R R, Correia L J H. StMichael: Protecting the Linux Kernel Integrity[J/OL]. 2006. http://www.thebugmagazine.

(下转第 135 页)

图 3 中,康复率 α 固定为 0.5 时:当不考虑延迟和重连的影响时,$t_r=8$,病毒感染规模稳定在 0.61 附近;当延迟系数 $\gamma=0.4$,重连系数 $\omega=0$ 时,$t_r=11$,而感染规模最终稳定在 0.59 附近;而当延迟和重连两者同时考虑时(即 $\gamma=0.4$,$\omega=0.5$),$t_r=45$,感染规模稳定在 0.35 附近。综上可知,延迟和拓扑重连是抑制病毒传播速率和感染规模的两个重要因素。

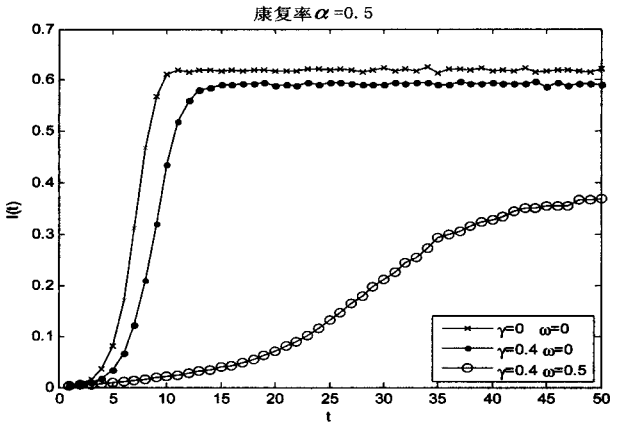

图 3　康复概率一定时,三种情况下病毒在适应网络中的传播趋势曲线

3　结束语

文中提出使用异步元胞自动机在自适应网络中建立病毒传播的 SIS—De 模型,基于计算机网络中经常出现拥塞、延迟的实际情况,考虑网络拓扑结构动态变化的适应网络,扩充异步元胞自动机的传统定义,结合节点状态不同步演化的现实情况,通过提出的模型对初始拓扑为 WS 小世界网络的病毒传播行为进行了研究。

仿真结果表明当康复率为 0 时,由于链路中存在传播时延,会使病毒的传播速率降低,但最后感染规模仍为全部节点都被感染;而当考虑康复概率时,随着延迟系数的增大,病毒的传播速率和最后的爆发规模都降低了;最后当延迟和拓扑重连同时考虑时,由于两者的共同影响,相对于拓扑不变的静态网络,此时病毒的传播速率和爆发规模进一步降低,增益明显。由此可以得出结论,计算机网络中的传播时延和拓扑动态改变两种实际情况是影响病毒在其中传播的重要因素,如果想控制病毒的爆发速度和感染规模,可以从这两方面制定相应的免疫策略,来达到控制和降低计算机病毒危害的目的。

参考文献:

[1] Watts D J. The 'new' science of networks[J]. Annual Review of Sociology,2004,30:243-270.

[2] 宋玉蓉,蒋国平. 基于一维元胞自动机的复杂网络恶意软件传播研究[J]. 物理学报,2009,58(9):5911-5918.

[3] Pastor-Satorras R,Vespignani A. Epidemic spreading in scale-free network[J]. Physical Review Letters,2001,86(14):3200-3203.

[4] Pastor-Satorras R,Vespignani A. Epidemic dynamics and endemic states in complex network[J]. Physical Review E,2001,63(6):66-117.

[5] Gross T,D'lima C J D,Blasius B. Epidemic dynamics on an adaptive network[J]. Phys. Rev. Lett.,2006,96(20):208701.

[6] Shaw L B,Schwartz I B. Fluctuating epidemics on adaptive networks[J]. Phys. Rev. E,2008,77(6):066101.

[7] 叶东海,蒋国平,宋玉蓉. 多局域世界复杂网络中的病毒传播研究[J]. 计算机工程,2010,36(23):130-132.

[8] 陈　昕,宋玉蓉,蒋国平. 二维异步元胞自动机计算机病毒传播[J]. 北京邮电大学学报,2011,34(Sup):90-94.

[9] 王亚奇,蒋国平. 基于元胞自动机考虑传播延迟的复杂网络病毒传播研究[J]. 物理学报,2011,60(8):080510.

[10] Neumann J V,Burks A. Theory of Self-reproducing Automata[M]. Urbana:University of Illinois Press,1966.

[11] Nehaniv C L. Self-reproduction in Asynchronous Cellular Automata[C]//Proceedings of the 2002 NASA/DOD Conference on Evolvable Hardware. [s. l.]:[s. n.],2002:201-209.

[12] Nehaniv C L. Asynchronous Automata Networks Can Emulate Any Synchronous Automata Network[J]. International Journal of Algebra and Computation,2004,14(5-6):719-739.

[13] Gross T,Blasius B. Adaptive coevolutionary networks:a review[J]. J. R. Soc. Interface,2008,5(20):259-271.

(上接第 131 页)

org/magazine/bug02/0x07_stmichael.txt.

[8] 龚　友. Linux 下内核级 Rootkit 检测防护机制的研究[D]. 成都:电子科技大学,2006.

[9] 颜仁仲,钟锡昌. 实时检测 Rootkit 并自动修复系统的研究与实现[D]. 北京:中国科学院研究生院,2006.

[10] Sinch A. An introduction to virtualization[J/OL]. 2006-05-12. http://www. kernelthread. com/publications/virtualization.

[11] Quynh N A,Takefuji Y. Towards a Tamper-resistant Kernel Rootkit Detector[C]//SAC '07 Proceedings of the 2007 ACM Symposium on Applied Computing Table of Contents. [s. l.]:[s. n.],2007:276-283.

[12] 陈　健,范明钰. 基于恶意软件分类的特征码提取方法[J]. 计算机应用,2011(1):83-84.

[13] 金　庆,吴国新,李　丹. 反病毒引擎及特征码自动提取算法研究[J]. 计算机工程与设计,2007(24):5863-5866.

[14] 赵　帅,武延军. 基于虚拟机架构的内核 Rootkit 防范方案[J]. 计算机工程与应用,2011,47(11):72-74.

[15] 石　磊,邹德清,金　海. Xen 虚拟机化技术[M]. 武汉:华中科技大学出版社,2009.