第43卷 第6A期
2016年6月

计 算 机 科 学
Computer Science

Vol. 43 No. 6A
June 2016

虚拟化环境下基于职能分离的Rootkit检测系统架构研究

朱智强 赵志远 孙 磊 杨 杰
(解放军信息工程大学三院 郑州 450000)

摘 要 针对现有虚拟化环境下Rootkit检测技术易被绕过、性能开销大的问题,提出了虚拟化环境下基于职能分离的检测系统架构XenMatrix,其在保证检测系统透明性的同时提高了自身的安全性,设计了检测频率的自适应调整策略,实现了Rootkit检测频率的动态调整,有效降低了系统的性能开销。最后对实验结果的分析表明,相比现有检测技术,该原型系统能够有效检测Rookit,具有较高的检测率和较低的性能开销。

关键词 虚拟化,职能分离,Rootkit,自适应
中图法分类号 TP309 文献标识码 A

Research on Rootkit Detection System Architecture Based on Functional Separation in Virtualized Environment

ZHU Zhi-qiang ZHAO Zhi-yuan SUN Lei YANG Jie
(The Third Institute, PLA Information Engineering University, Zhengzhou 450000, China)

Abstract A kind of Rootkit detection system architecture XenMatrix based on duty separation in virtualization environment was proposed in light of the problems of Rootkit detection technology being easy to be avoided and large performance overhead in existing virtualization environment, which can improve the security of its own and at the same time ensure the transparency of the detecting system. A strategy of adaptive adjustment to detect the frequency was proposed, which can achieve dynamic adjustment of Rootkit detecting frequency and reduce the overhead of the system effectively. The analysis of experimental results shows that this prototype system can effectively detect known and unknown Rootkit and has higher success rate of detecting and lower performance overhead compared to existing detecting technology at present.

Keywords Virtualization, Functional separation, Rootkit, Self-adaption

1 引言

云计算因具有动态扩展、按需服务、按量计费等优势而成为继互联网之后又一次信息技术革命[1],但随着越来越多用户将数据业务迁移至云服务端,云上的信息安全成为用户关注的焦点。虚拟化技术是云计算的基础[2],虚拟化环境的安全性直接影响用户业务数据的安全。安全监控技术是增强系统安全性的一个重要的有效手段,但是恶意代码往往借助于Rootkit的隐藏能力来躲避监控系统的检测。因此,实现对Rootkit的有效检测对增强虚拟化环境的安全性具有重要意义。

文献[3]中陈祝红在Xen虚拟化平台上设计了一种入侵检测系统fortis。该系统能够获取被监控系统的内核空间和所有进程空间的所有内存信息,进而利用这些信息对被监控系统状态进行判断,检测是否有恶意代码存在于系统中。基于交叉视图的对比检测技术主要是通过对System.map、CR3寄存器等系统内信息重构出系统的进程列表与系统通过ps等命令获取的进程列表相互对比,查看是否存在隐藏的进程来判断是否存在Rootkit。文献[4-6]通过这种技术实现了对虚拟机中的恶意代码Rootkit的检测。文献[7]研究了一种利用人工免疫算法检测恶意程序的技术,首先对人工免疫进行优化,然后将其用于检测计算机和手机中的恶意程序,在一定程度提高了检测率。文献[8]中采用遗传算法和神经网络相结合的办法,通过预测来达到检测恶意代码的目的,该方法能有效检测CPU数据集中的样本。通过总结,现有虚拟化环境下Rootkit检测技术主要存在以下问题:1)对恶意代码不透明,检测失败率较高;2)对系统性能影响较大。

本文提出一种虚拟化环境下基于职能分离的Rootkit检测系统架构。针对现有的虚拟化环境下Rootkit检测技术易被绕过、性能开销大的问题,提出了虚拟化环境下基于职能分离的检测系统架构XenMatrix,其在保证检测系统透明性的同时提高了自身的安全性;同时,系统架构中的检测频率的自适应调整策略实现了Rootkit检测频率的动态调整,有效降低了系统的性能开销。

2 基于职能分离的Rootkit检测系统架构

针对现有的虚拟化环境下Rootkit检测技术易被绕过、性能开销大的问题,提出了虚拟化环境下基于职能分离的检

本文受国家863计划基金项目(2008AA01Z404),国防预研基金项目(910A26010306JB5201)资助。
朱智强(1961—),男,教授,主要研究方向为云计算、信息安全;赵志远(1989—),男,硕士生,主要研究方向为虚拟化技术、信息安全,E-mail:zzy—taurus@foxmail.com;孙 磊(1973—),男,博士,副研究员,主要研究方向为云计算基础设施可信增强、可信虚拟化技术;杨 杰(1990—),男,硕士生,主要研究方向为云计算、可信虚拟化技术。

测系统架构 XenMatrix，其在保证检测系统透明性的同时提高了自身的安全性；同时，职能分离的特性有效减小了检测系统对 Dom0 的性能影响。

2.1 XenMatrix 系统组成单元

虚拟化环境下的 Rootkit 检测方法中存在的两个主要问题是对未知 Rootkit 检测率不高和检测工具的运行对特权虚拟机的性能影响较大。本文主要针对这两个问题，提出了一种基于职能分离的 Rootkit 检测系统 XenMatrix。该系统主要是将获取行为数据和分析行为数据两个功能部署在不同虚拟机中，以减小获取行为数据对特权虚拟机性能的影响；利用人工智能算法的自学习能力分析行为数据，通过学习已有的行为来判断未知的行为，以此来检测未知的 Rootkit。该检测系统主要包括行为特征捕获模块 LSMBF(Getting Behavior Feature by Scaning Memory Based on Libvmi)、检测频率的自适应调整策略(Auto Monitoring Frequency)、NNDRM(The Detecting Rootkit Model Based Neural Network)模型、用户界面。XenMatrix 系统的组成单元如图 1 所示。

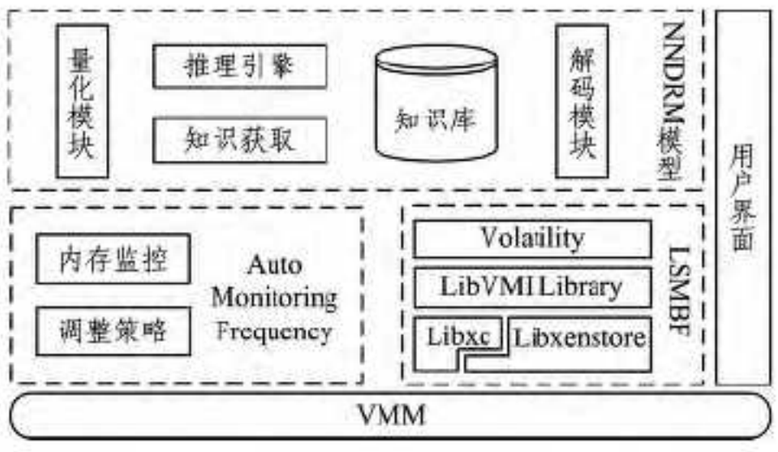


图 1 Rootkit 检测模型

LSMBF 模块的功能是捕获目标操作系统中需要检测的内存区域中与恶意代码 Rootkit 相关的、具有代表性的，并且尽可能详细与准确的信息，如软件执行过程中的行为数据、文件注册表项等信息、软件执行后的遗留痕迹等。

Auto Monitoring Frequency 为自调整检测频率策略，其根据当前系统的负载情况决策扫描内存的时间间隔。检测时间间隔对于最终检测结果的准确性非常重要，若检测时间过小，虽然检测结果会更加准确，但会大大增加系统的负载，使系统性能损失过大；若检测时间太大，虽然给系统增加的性能损失小，但是准确率会大幅下降。在这对矛盾的需求情况下，如何选择一个合适的检测间隔，甚至根据负载情况自动调整检测频率，至关重要。

NNDRM 模型是 XenMatrix 系统架构的核心部分，主要功能是经过训练样本集的学习，在测试样本集测试通过的情况下，完成系统行为特征知识库的建立。这个过程是一个动态的完成知识获取的过程，并且可以通过自学习完成对一些未知 Rootkit 的行为特征的存储。行为特征知识库由神经网络的结构、权值和阈值构成。系统测试之后方可投入到实际应用，将获取的分析数据集与知识库进行推理判断，决策是否存在 Rootkit。

用户界面用于显示最终的推理结果，方便用户及时查看，并做出相应的响应措施。

2.2 XenMatrix 系统架构

本文是在虚拟化环境下检测客户操作系统中是否存在恶意代码 Rootkit，因此在虚拟机中如何部署 XenMatrix 检测系统至关重要。本文选择在 Xen 虚拟机中实现该系统架构的设计。由于 Dom0 具有高特权级，同时为了不修改 VMM，本文将 XenMatrix 检测系统部署在 Dom0 中。但是映射内存、获取行为特征会较大程度影响系统性能，导致 Dom0 的性能下降。为了解决这个问题，本文将分离出一个新的域并对其赋予一定特权，将 LSMBF 模块部署在这个新隔离出来的域中，而 XenMatrix 其余模块依然部署在 Dom0 中，此时 XenMatrix 部署情况如图 2 所示。这种部署方式可以充分利用虚拟化架构的强隔离性和 Dom0 的安全性来保证检测系统安全可靠。

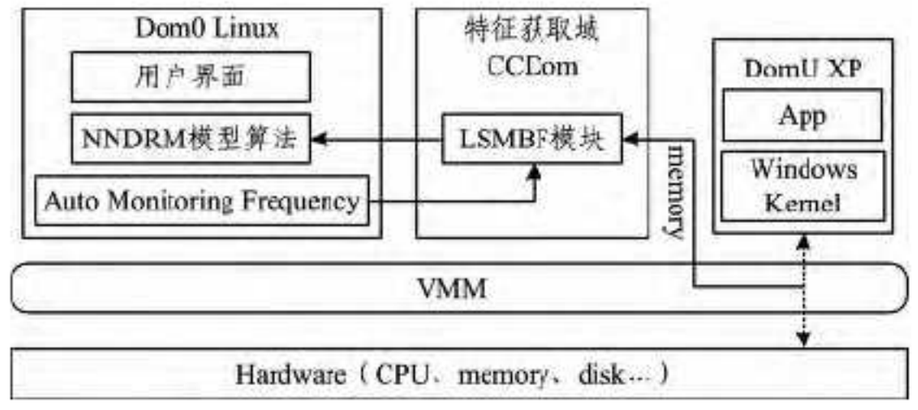


图 2 Xen 环境下 XenMatrix 系统结构图

这种部署模式分离出一个特征获取域(Characteristics of Capture Domain，CCDom)，并且 CCDom 域类似于独立设备驱动域，也是特权域的一种。将行为特征获取的工作从 Dom0 中分离出来将在很大程度上减小对 Dom0 的性能影响。同时这种部署方式也不需要修改 VMM，只是给特征获取域分配一定的特权，让其可以调用 libxc、libxenstore，访问内存等即可。这种部署方式也增强了系统的安全性，因为 Dom0 中的模块变少，所以 Dom0 更加安全，而特征获取域若被攻击或者宕机，Dom0 立刻可以发现并重新启动特征或区域，将其恢复至初始状态即可。

2.3 CCDom 与 Dom0 通讯

XenMatrix 架构中，Dom0 与 CCDom 之间需要信息传递，本文采用共享内存的方式来传递这些信息。Xen 系统中的内存共享机制与 Linux 系统类似，但是也有一些不同之处：共享内存区域在特定时间内只可归属其中一个 Domain；共享内存区域被访问完成后，访问权限将被收回。

Xen 中以内存页为基本单元对内存进行授权，所以内存映射也可以称为页面映射。假设 Domain1 提供共享内存页并授权 Domain2 访问，访问步骤如图 3 所示。

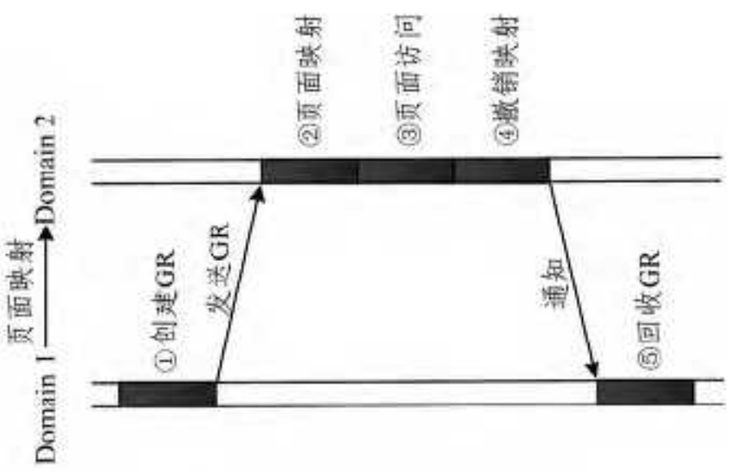


图 3 页面映射流程

(Domain1 和 Domain2 与 Dom0 和 CCDom 对应)

步骤 1 Domain1 首先创建一个授权引用 GR，然后将其传送给 Domain2；

步骤 2 Domain2 把 GR 授权过的内存页映射至自身的地址空间内；

步骤 3 Domain2 访问该内存页；

步骤 4 Domain2 撤销该内存页的映射；

步骤 5　Domain1 撤销授权，回收 GR。

由于虚拟机监视器和 Dom0 具有较高特权，因此本文假设它们是可信的，即客户虚拟机受到攻击不能影响虚拟机监视器和 Dom0 的安全。本文分离出的 CCDom 域具有较高特权，类似于独立设备驱动域，与 Dom0 都运行在 Ring1 环，而其他虚拟机运行在 Ring3 环，所以 CCDom 域不可被攻击，而 CCDom 与 Dom0 域之间的通信也不可能被截获。若二者之间的通信被截获，那么 Dom0 和 CCDom 都将可以被攻击，与假设情况不符，与本文研究情况也不相符。

3　XenMatrix 系统架构关键模块研究

NNDRM 模型作为 XenMatrix 系统的核心部分，其主要设计思想是利用神经网络知识推理问题的能力，实现了自学习、自动更新知识库、自动推理等功能，提高了系统整体的智能水平和对未知 Rootkit 的检测率；自调整检测频率策略可以根据系统内存的使用率情况自动调整检测时间间隔，动态适应环境，提高系统性能；行为特征捕获模块将获取的行为特征输入至 NNDRM 中，是 NNDRM 检测模型的输入来源。

3.1　NNDRM 检测模型结构

针对目前现有的虚拟化环境下 Rootkit 检测效率不高、容易出现误判等问题，提出了基于神经网络的 NNDRM 检测模型，由于神经网络具有自学习的能力，因此利用这种能力可以提高对未知 Rootkit 的检测率。神经网络是对人脑神经元网络的简化和模仿，由大量神经元以拓扑结构相连接[9]。利用神经网络的高效性和自学习能力[10,11]，仅仅需用专家或者事实判断 Rootkit 恶意代码行为的知识来训练神经网络，得出与专家相一致的判断即可。同时利用神经网络的自学习能力可以不断更新知识库，从而提高对未知 Rootkit 的检测率，打破目前检测方法对未知 Rootkit 检测准确性不高的局面。其结构如图 4 所示。

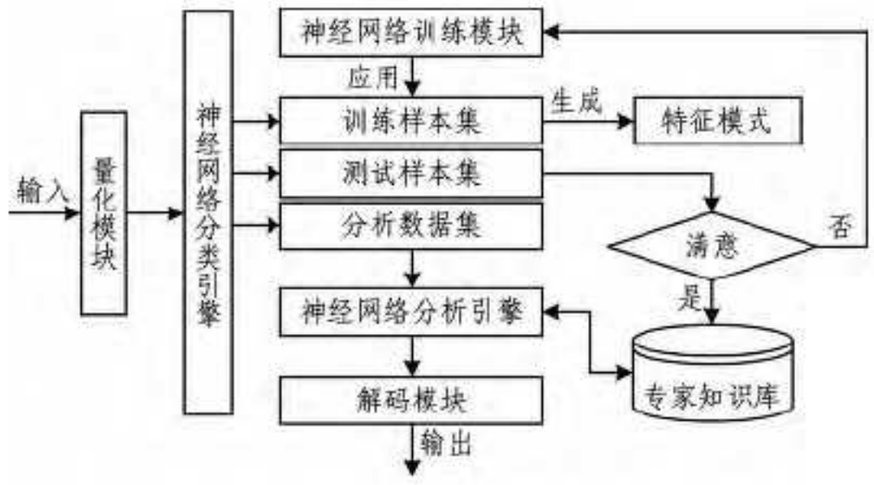


图 4　NNDRM 模型结构图

3.2　检测频率的自适应调整策略

该系统采用定时扫描内存来获取信息，通过对这些信息的分析来检测恶意代码，因此扫描内存的时间间隔对检测结果的准确性和性能开销非常关键。本文采用根据内存使用率自调整检测频率的策略，即扫描内存的时间间隔随着内存的使用率而调整。该策略主要涉及如何实时监控内存的占用率和如何实现自调整检测频率两个关键性问题。

Dom0 控制着 DomU 对硬件资源的访问操作，即经过原生设备驱动访问物理硬件资源。DomU 不能直接访问硬件资源，必须经过 Dom0 的协调实现对硬件资源的访问，因此将该策略模块部署在 Dom0 中，便能有效地监测 DomU 的内存资源使用率。为了实时地监测虚拟机的内存使用率，首先用 Xenstore 来实时地读取其相关信息；然后根据读取的信息判断内存资源占用情况。

在内存使用率较低时，内存资源充足，可以适当减小检测系统的扫描时间间隔；当内存使用率较高时，为减小检测系统对虚拟机的性能开销，应适当增大检测的时间间隔。本文认为当内存使用率低于 30% 时，内存资源充足；当内存使用率高于 80% 时，内存资源缺乏。所以内存使用率低于 30% 时，系统还有很多的资源可以被利用，这时为使系统安全，同样可以减小扫描间隔；当内存使用率在 30%～80% 之间时，为避免使用复杂算法带来的额外开销，这里使用一个线性公式来调整检测频率；当内存使用率高于 80% 时，设置一个固定的且较大的检测时间间隔。这种方法大大提高了检测效果，同时又不会严重影响系统的正常使用。

3.3　行为特征捕获模块

为准确检测 Rootkit，需要获取内存数据并从中分析出与恶意代码 Rootkit 相关的、具有代表性的，并且尽可能详细与准确的信息，因此需要行为特征捕获模块通过内存映射来获取这些信息。LibVMI 基于 libxc 封装了 Xen 提供的一系列接口，可以基于 Xen 的管理程序建立从特权域查看 DomU 内存的函数库。而本文检测系统的部署正好符合 VMI 架构，因此采用 LibVMI 函数库可以很好地在 CCDom 域中获取被检测系统的内存信息，实现 Rootkit 行为捕获的功能。

本文将行为特征捕获模块部署在分离出来的 CCDom 域中，CCDom 域已经获得直接访问内存的特权，因此其可以通过虚拟机监视器获取被检测虚拟机的内存信息，用以检测 Rootkit，同时能够有效防止内存信息在传递过程中被截获或者篡改。行为特征捕获模块原理图如图 5 所示。

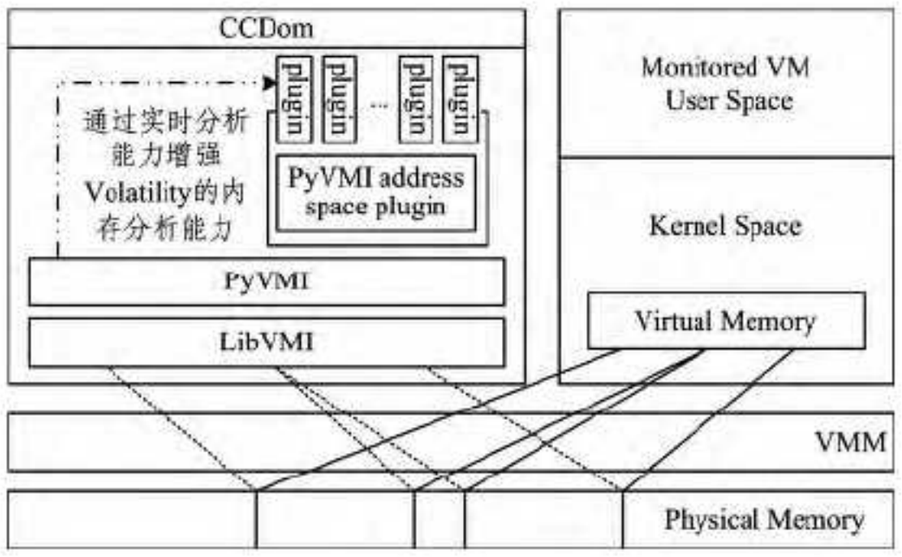


图 5　QPSO 算法流程图

4　实验及结果分析

4.1　实验环境构建

本文实验是在实验室内云桌面基础设施上搭建的，实验平台配置如表 1 所列。

表 1　实验平台配置

硬件平台配置
CPU：Intel(R) Xeon(R) CPU E5-2620 V2 @ 2.10GHz
内存：8GB
硬盘：500GB
软件平台配置
Xen：Xen4.4
Dom0：Fedora Core 20（内核版本：Linux XenHost31 3.11.10-301fc20.）
CCDom：Fedora Core 20（内核版本：Linux XenHost31 3.11.10-301fc20.）
DomU：Windows XP SP3

在该环境下搭建 Xen 虚拟机之后，运行 XenMatrix 系统，用户界面如图 6 所示。

图 6 XenMatrix 用户界面

4.2 检测结果

功能实验包括对正常系统和植入 Rootkit 后系统的检测，每次重复实验前均将系统恢复至纯净状态。选取网上知名 Rootkit：hxdef、futo、ntrootkit、futo-enhanced、badrkdemo，并用 Themida 工具处理模拟未知 Rootkit，共 10 个测试样本。安装 Rootkit 后，由于行为数据中包括大量的隐藏信息、钩挂信息，XenMatrix 系统可以捕获这些信息从而有效地检测出已知和未知的 Rootkit。UI 界面显示如图 7 所示。

图 7 UI 界面显示的测试结果

4.3 性能指标评估

在评估检测系统各项性能指标时，主要考虑其有效性、效率和可用性[12]。有效性是所有指标中最为主要的指标之一，用于分析结果的准确率；效率则是从处理数据速度和经济方面考虑，争取检测系统运行时对整体系统影响较小；可用性是指系统的可扩展性和系统对用户的方便程度。本文主要针对 XenMatrix 系统的有效性和性能效率方面进行分析。

4.3.1 检测系统有效性分析

检测率是指系统被攻击时可以正确判断的概率；误判率是指检测系统在检测时出现错误判断的概率。给出其定义：

$$检测率=\frac{检测出的\ Rootkit}{Rootkit\ 样本集}*100\%$$

$$误判率=\frac{误判断的程序}{样本集总数}*100\%$$

为验证本文方法在检测 Rootkit 时的有效性，选取 QQ、多米音乐等 20 个正常程序，同时选取 Hook Defender、Hacker-Defender、FURootkit、FUTo、NewRootkit11 等 10 个 Rootkit，并用 ACProtect、PE-Armor、Themida 和 Winlicense 4 种保护工具处理，得到 40 个变种 Rootkit，共 50 个恶意 Rootkit 程序，一共 70 个检测程序作为测试样本集。然后使用 Lycosid、XenPHD、Hyperchk、Livewire 和本文方法来检测样本集，检测结果如表 2 所列。

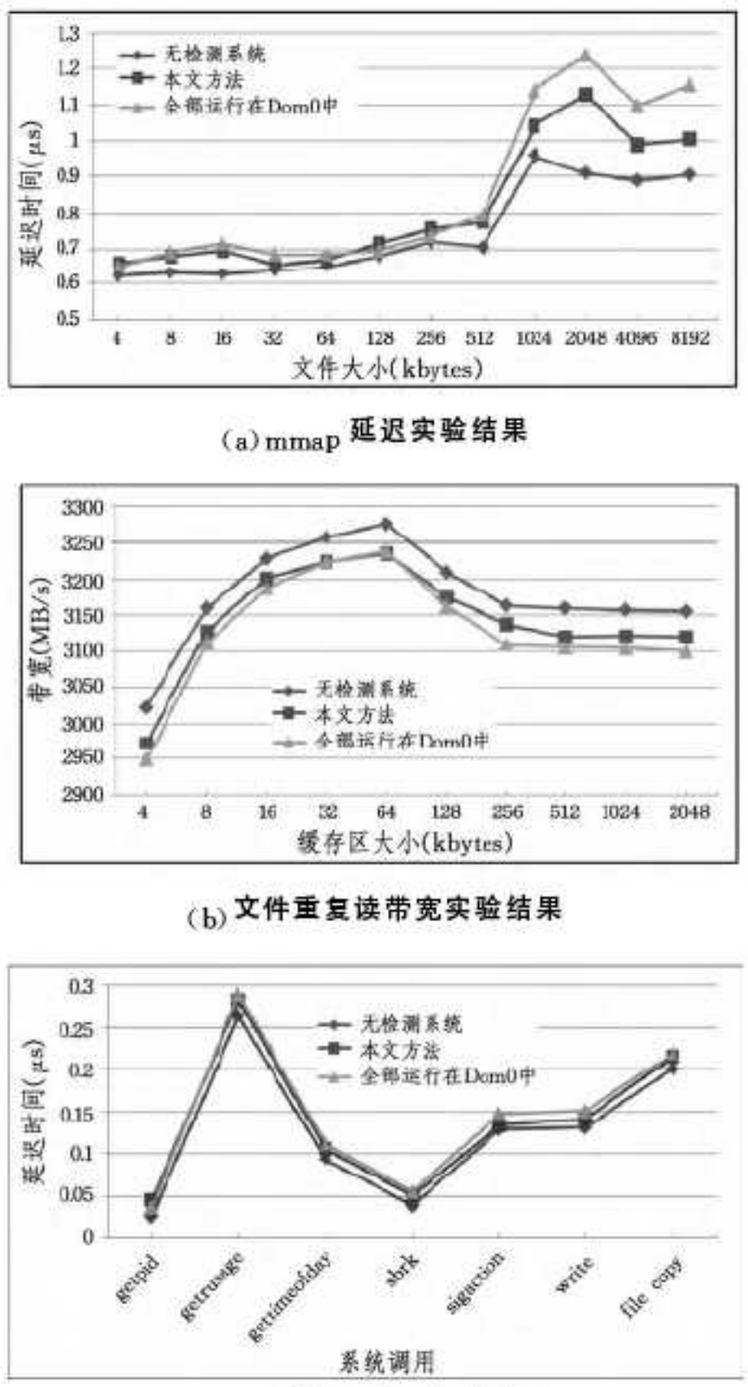
表 2 检测结果对比

	Lycosid	XenPHD	Hyperchk	Livewire	XenMatrix
正常程序	20	20	18	20	20
Rootkit	42	46	46	44	47
检测率(%)	84	92	92	88	94
误判率(%)	11.4	5.7	8.6	8.6	4.3

由表 2 可以看出，XenMatrix 系统的检测效果要优于其它检测系统的效果，其能够检测出未知的恶意代码，但是在一定程度上也有误报的情况，不能完全检测出所有 Rootkit。由于 Lycosid 检测系统不能检测直接内核攻击，检测效率比较底；Hyperchk 检测系统利用完整性检测，虽然检测率比较高，但是误判率也非常高。XenPHD 和 Livewire 在检测率和误判率上较好，但还是略低于本文方法。通过实验对比，以本文方法在有效性方面有一定优势。

4.3.2 检测系统性能分析

本小节主要测试本系统运行时对整体性能的影响，选用 Hbench 测试工具，并且与其他几种检测系统进行对比，以说明本系统性能损失情况。Hbench 是一个操作系统级别的测试工具，其能够测试内存的延迟和带宽。其测试原理是对常用的文件映射系统函数 mmap 和文件读取系统函数 read 进行测试，同时选用 getpid 测试程序和文件拷贝等操作对系统性能进行测试，进而评价 XenMatrix 检测系统对系统整体的性能影响。在进行虚拟机性能测试过程中，分别对不运行 XenMatrix 系统和运行 XenMatrix 系统进行对比，以此来说明该检测系统对整体性能影响不大，本文方法和将所有模块都运行在 Dom0 中进行对比，以此来说明本文基于职能分离的检测系统带来的性能损失更小。测试结果如图 8 所示。



(a) mmap 延迟实验结果

(b) 文件重复读带宽实验结果

(c) 系统调用延迟实验结果

图 8 实验结果

图 8(a) 实验结果中看出，纵坐标代表从 mmap 映射操作开始到 mmap 映射操作结束所耗费的延迟时间，横坐标代表映射磁盘文件的大小。函数 mmap 的工作原理是直接将磁盘文件数据映射到用户进程的地址空间，致使文件的读取无需在内核空间和用户空间切换。mmap 操作延迟时间虽然与映

射文件的大小不是线性关系，但是从整体上来看，映射文件越大，mmap 操作的延迟时间也越长。从图中可以看出文件大小在 1M 大小之后延迟相对较大，但最大处也就是文件大小是 2M 处延时也只增加了 0.219μs，所以总体看来该系统对 mmap 的性能影响较小。

应用程序在调用读函数 read() 时，先把数据读入到内核缓冲区，待其填满后再将文件数据读入变量中。图 8(b) 实验结果中，纵坐标表示带宽，横坐标表示设置的文件系统缓冲区大小。设置文件大小为固定值 2kB。从图中可以看出，带宽随着缓冲区的增大先逐渐增大，然后减小，最终会趋于一个稳定值。带宽最大值出现在缓冲区大小为 64kB 处，这说明缓冲区大小设置为 64kB 时能发挥文件读取的最大性能，并且此时运行 XenMatrix 系统相对于不运行检测系统带宽下降 1.25%。在缓冲区大小为 4k 处带宽下降最大，为 1.65%。由此可以说明，此检测系统对文件重复读的带宽影响较小。

图 8(c) 实验结果中看出，还包含一些其他系统调用函数的延时时间。其中，getpid 测试程序只执行 getpid 系统调用操作，本文通过调用 getpid 程序 1500 次来仿真大量系统调用的情况；getrusage 函数的功能是获取进程的资源使用信息；gettimeofday 获取系统时间；sbrk 增加程序的数据空间；sigaction 改变进程收到某个信号后所执行的行为；write 将缓冲区数据写入文件；file copy 是 I/O 密集型操作，包括非常多的 read 和 write 操作。从整体看来，运行 XenMatrix 系统所引起的延迟处于可以接受的范围。

从图 8 实验结果中看出，本文方法在性能损失方面大部分测试项性能损失都小于将全部模块都运行在 Dom0 中的情况，只有少部分高于全部部署在 Dom0 中，并且本文方法的性能损失较小，结果表明基于职能分离的检测系统有效地减小了对系统性能的影响。

为了更加清楚地展示该检测系统对系统性能的影响，本文对 XenPHD、Hyperchk、Livewire 和所提方法进行性能测试，测试工具为 UnixBench。测试结果以没有任何检测系统为基准，计算各系统的性能损失，如图 9 所示。

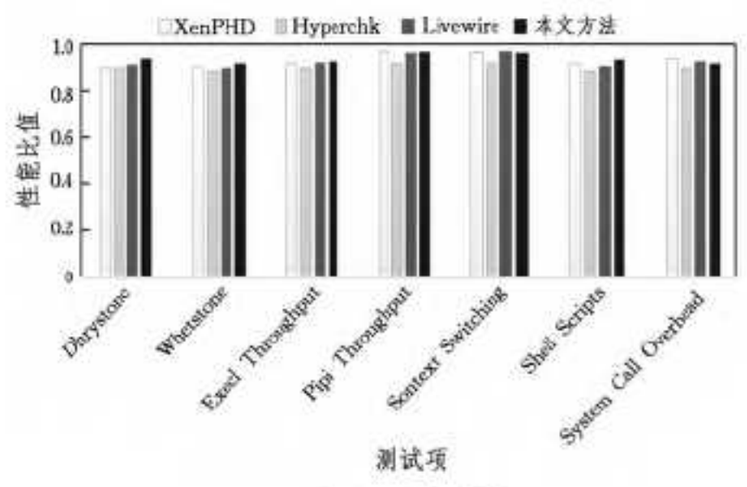


图 9 性能测试对比图

由于 XenPHD 是对 Lycosid 方法的改进，且在有效性分析时 XenPHD 表现更好，因此此处性能测试不再考虑 Lycosid。图 9 实验结果中，纵坐标表示各检测系统运行时与没有运行检测系统时的比值，横坐标表示测试项。从图中可以看出，本文系统在各项测试项目上的比值都比较高，唯有在 System Call Overhead 测试项略低，由于 XenMatrix 系统的职能分离导致系统调用路径更长，因此导致了一定的性能开销。但是从整体上看，本文方法在性能损失方面较其他方法表现更好，且平均计算由于运行 XenMatrix 检测系统所带来的性能损失为 5.7%，处于可以接受的范围。

结束语 针对现有的虚拟化环境下 Rootkit 检测技术易被绕过、性能开销大的问题，提出了虚拟化环境下基于职能分离的检测系统架构 XenMatrix，其在保证检测系统透明性的同时提高了自身的安全性，设计了检测频率的自适应调整策略，实现了 Rootkit 检测频率的动态调整，有效降低了系统的性能开销。本文主要是验证职能分离和检测频率的自适应调整策略的有效性，未对 NNDRM 检测模型中算法做深入研究，这也是下一步的主要研究工作。

参 考 文 献

[1] Kale V. Guide to Cloud Computing for Business and Technology Managers: From Distributed Computing to Cloudware Applications[M]. CRC Press, 2014

[2] 石磊，邹德清，金海，等. Xen 虚拟化技术[M]. 武汉：华中科技大学出版社，2009

[3] 陈祝红. Xen 虚拟化平台下入侵检测系统研究与实现[D]. 合肥：中国科学技术大学，2013

[4] Jones S T, Arpaci-Dusseau A C, Arpaci-Dusseau R H. Antfarm: Tracking Processes in a Virtual Machine Environment[C]// USENIX Annual Technical Conference. General Track, 2006: 1-14

[5] Chen L, Liu B, Zhang J, et al. An advanced method of process reconstruction based on VMM[C]// 2011 International Conference on Computer Science and Network Technology (ICCSNT). IEEE, 2011, 2: 987-992

[6] 陈林. 基于虚拟机的恶意代码检测关键技术研究[D]. 长沙：国防科学技术大学，2012

[7] 芦天亮. 基于人工免疫系统的恶意代码检测技术研究[D]. 北京：北京邮电大学，2013

[8] Dastanpour A, Ibrahim S, Mashinchi R. Using Genetic Algorithm to Supporting Artificial Neural Network for Intrusion Detection System[C]// The International Conference on Computer Security and Digital Investigation (ComSec2014). The Society of Digital Information and Wireless Communication, 2014: 1-13

[9] Negnevitsky M. Artificial intelligence: a guide to intelligent systems[M]. Pearson Education, 2005

[10] Negnevitsky M. Artificial intelligence: a guide to intelligent systems[M]. Pearson Education, 2005

[11] 陈易，张杭，胡航. 基于 BP 神经网络的协作频谱感知技术[J]. 计算机科学，2015, 42(2): 43-45, 64

[12] 陈友，程学旗，李洋，等. 基于特征选择的轻量级入侵检测系统[J]. 软件学报，2007, 18(7): 1639-1651