软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(8):2173−2187 [doi: 10.3724/SP.J.1001.2012.04219]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于虚拟化的安全监控*

项国富[1+], 金 海[1], 邹德清[1], 陈学广[2]

[1](华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

[2](华中科技大学 控制科学与工程系,湖北 武汉 430074)

Virtualization-Based Security Monitoring

XIANG Guo-Fu[1+], JIN Hai[1], ZOU De-Qing[1], CHEN Xue-Guang[2]

[1](School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

[2](Department of Control Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China)

+ Corresponding author: E-mail: whxgf1984@163.com

Xiang GF, Jin H, Zou DQ, Chen XG. Virtualization-Based security monitoring. *Journal of Software*, 2012, 23(8):2173−2187 (in Chinese). http://www.jos.org.cn/1000-9825/4219.htm

Abstract: In recent years, virtualization technology is the novel trendy of computer architecture, and it provides a solution for security monitoring. Due to the highest privilege and the smaller trusted computing base of virtual machine monitor, security tools, deployed in an isolated virtual machine, can inspect the target virtual machine with the help of virtual machine monitor. This approach can enhance the effectiveness and anti-attack ability of security tools. From the aspect of the implementation technologies, existing research works can be classified into internal monitoring and external monitoring. According to the different targets, the related works about virtualization-based monitoring are introduced in this paper in detail, such as intrusion detection, honeypot, file integrity monitoring, malware detection and analysis, security monitoring architecture and the generality of monitoring. Finally, this paper summarizes the shortcomings of existing works, and presents the future research directions. It is significant for virtualization research and security monitoring research.

Key words: virtualization; virtual machine monitor; security monitoring; virtual machine introspection

摘 要: 近年来,虚拟化技术成为计算机系统结构的发展趋势,并为安全监控提供了一种解决思路.由于虚拟机管理器具有更高的权限和更小的可信计算基,利用虚拟机管理器在单独的虚拟机中部署安全工具能够对目标虚拟机进行检测.这种方法能够保证监控工具的有效性和防攻击性.从技术实现的角度来看,现有的研究工作可以分为内部监控和外部监控.根据不同的监控目的,详细地介绍了基于虚拟化安全监控的相关工作,例如入侵检测、蜜罐、文件完整性监控、恶意代码检测与分析、安全监控架构和安全监控通用性.最后总结了现有研究工作的不足,并指出了未来的研究方向.这对于从事虚拟化研究和安全监控研究都具有重要意义.

关键词: 虚拟化;虚拟机管理器;安全监控;虚拟机自省

中图法分类号: TP316 **文献标识码**: A

* 基金项目: 国家自然科学基金(60973038, 61142010); 国家高技术研究发展计划(863)(2012AA012600); 武汉市科技攻关项目(201010621211); 信息网络安全公安部重点实验室开放课题(C11602)

收稿时间: 2011-05-04; 修改时间: 2011-11-02; 定稿时间: 2012-03-23

目前,虚拟化技术(virtualization)[1−4]在企业中得到了广泛的应用,并具有巨大的市场前景.据 IDC (International Data Corporation)预计,2011 年,虚拟化服务市场将达到 117 亿美元[5].在 2009 年和 2010 年连续两年公布的 Gartner 技术发展趋势报告中,虚拟化技术都成为十大 IT 技术之一[6].虚拟化技术改变了系统软件与底层硬件紧耦合的方式,可以更加灵活地配置与管理计算系统.虚拟化技术最早由 IBM 在 20 世纪 60 年代提出,并成功应用于大型机 VM370[7].根据 Popek 和 Goldberg 在文献[8]中提出的可虚拟化体系架构的 3 个条件:等价性(equivalence)、资源控制(resource control)和有效性(efficiency),目前广泛使用的 X86 架构是不支持虚拟化的[9].随着个人计算机(主要是 X86 架构)处理能力的增强,同时,VMware 将虚拟化技术引入到 X86 架构[10],从此,虚拟化技术及其应用得到了蓬勃的发展,常见的虚拟化软件有 VMWare ESX/GSX/Workstation[11],Xen[12−15].根据虚拟层次的不同,虚拟化技术可以分为指令级虚拟化、硬件抽象级虚拟化、操作系统级虚拟化、运行库级虚拟化和编程语言级虚拟化[16].每个虚拟操作环境(包括操作系统和其上运行的应用程序)被称为虚拟机(virtual machine,简称 VM);将底层硬件资源进行抽象,从而供多个虚拟机使用的虚拟化软件被称为虚拟机管理器(virtual machine monitor,简称 VMM 或 hypervisor).

图 1 将传统架构与虚拟化架构进行了比较.从安全的角度来看,传统架构的操作系统具有最高权限,负责管理整个硬件平台.而在虚拟化架构中,虚拟机管理器位于操作系统和真实硬件平台之间,比操作系统的特权级更高,而且代码量更少.

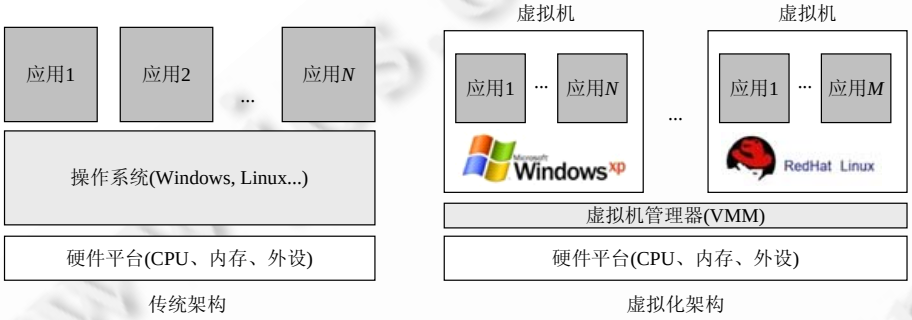


Fig.1 Comparison between traditional architecture and virtualization architecture

图 1 传统架构和虚拟化架构的比较

与传统架构相比,虚拟化架构的优势在于[17]:

(1) 更小的可信计算基(trusted computing base,简称 TCB):虚拟机管理器的代码量一般在 10 万行左右,远远小于操作系统的代码量.例如,Linux 2.6.27 大概有 1 000 万行[18],而 Windows XP 大概有 3 500 万行[19].这就意味着虚拟机管理器自身具有的 Bug 数量更少,也比操作系统更健壮.

(2) 更好的隔离性(isolation):在传统架构中,应用程序通过进程的虚拟地址空间来进行隔离,进程之间可能相互干扰.例如,某个进程出现故障导致整个系统崩溃,从而影响到其他进程的正常运行.而在虚拟化架构中,应用程序是以虚拟机为粒度进行隔离的,因此,虚拟机架构提供更好的隔离性.

由于虚拟化架构具有上述优势,基于虚拟化架构的安全工具能够有效地监控虚拟机的内部状态,同时抵御被监控系统中可能发生的攻击.因此,在虚拟机管理器层提供服务,可以在一定程度上增强计算系统的安全性(security)和可移动性(mobility)[20].

文献[20]列出了基于虚拟化架构的 3 种服务实例:安全日志(secure logging)、入侵防御和检测(intrusion prevention and detection)、环境迁移(environment migration).

目前,基于虚拟机架构来增强安全工具的安全性已经成为研究趋势,例如,入侵检测、系统日志、蜜罐、完整性监控、恶意代码检测与分析、安全监控架构、监控通用性等.

1 基于虚拟化安全监控的分类

近几年来,由于虚拟机管理器自身具有更小的可信基和更好的隔离性的优势,基于虚拟机的安全监控都是利用虚拟机管理器隔离和保护特定的安全工具.因此,从安全监控实现技术的角度来看,基于虚拟化安全监控的相关研究工作可以分为两大类:

(1) 内部监控:在虚拟机中加载内核模块来拦截目标虚拟机的内部事件,而内核模块的安全通过虚拟机管理器来进行保护.

(2) 外部监控:通过在虚拟机管理器中对虚拟机中事件进行拦截,从而在虚拟机外部进行检测.

1.1 内部监控

图 2 说明了内部监控的架构,其典型的代表系统是 Lares[21]和 SIM[22].被监控的系统运行在目标虚拟机中,安全工具部署在一个隔离的虚拟域(安全域)中.这种架构支持在虚拟机的客户操作系统的任何位置部署钩子函数,这些钩子函数可以拦截某些事件,例如进程创建、文件读写等.由于客户操作系统不可信,因此这些钩子函数需要得到特殊的保护.当这些钩子函数加载到客户操作系统中时,向虚拟机管理器通知其占据的内存空间.由内存保护模块根据钩子函数所在的内存页面对其进行保护,从而防止恶意攻击者篡改.在探测到虚拟机中发生某些事件时,钩子函数主动地陷入到虚拟机管理器中.通过跳转模块,将虚拟机中发生的事件传递到管理域的安全驱动.安全工具执行某种安全策略,然后将响应发送到安全驱动,从而对虚拟机中的事件采取响应措施.跳转模块的功能是在虚拟机和管理域之间通信的桥梁.为了防止恶意攻击者篡改,截获事件的钩子函数和跳转模块都是自包含的(self-contained),不能调用内核的其他函数.同时,它们都必须很简单,可以方便地被内存保护模块所保护.

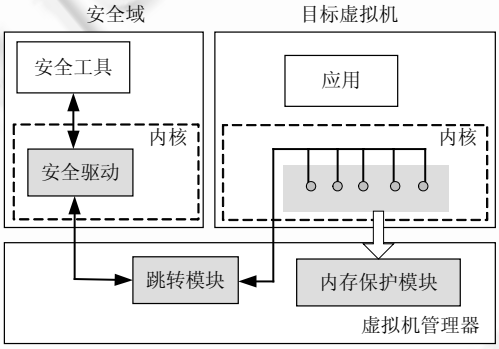


Fig.2 Architecture of internal monitoring[21]

图 2 内部监控的架构[21]

这种架构的优势在于,事件截获在虚拟机中实现,而且可以直接获取操作系统级语义.由于不需要进行语义重构,因此减少了性能开销.然而,它需要在客户操作系统中插入内核模块,对其不具有透明性.而且,内存保护模块和跳转模块是与目标虚拟机紧密相关的,不具有通用性.

1.2 外部监控

与内部监控相比,外部监控具有一定的优势,因此相关的研究工作也就更多.图 3 说明了外部监控的架构,其典型的代表系统是 Livewire[23]等.

外部监控是指在目标虚拟机外部,由位于安全域的安全工具按照某种策略对其进行检测.从图 3 中可以看出,监控点部署在虚拟机管理器中,它是安全域中的安全工具和目标虚拟机之间通信的桥梁.监控点拦截目标虚拟机中发生的事件,重构出高级语义并传递给安全工具.安全工具根据安全策略产生的响应,通过监控点来控制目标虚拟机.虚拟机管理器将安全工具与目标虚拟机隔离开来,增强了安全工具的安全性.由于虚拟机管理器位

于目标虚拟机的底层,因此监控点可以观测到目标虚拟机的状态(例如 CPU 信息、内存页面等).在虚拟机管理器的辅助下,安全工具能够对目标虚拟机进行检测.一般来说,外部监控一般包含两种基本功能:事件截获和语义重构.事件截获是指拦截虚拟机中发生的某些事件,从而触发安全工具对其进行检测.由于虚拟机管理器位于目标虚拟机的下层,因此只能获取低级语义(例如寄存器和内存页面).而监控工具是针对操作系统层的语义,因此两者之间存在语义鸿沟(semantic gap).为了使监控工具能够"理解"目标虚拟机中的事件,因此必须进行语义重构(semantic reconstruction).语义重构是指由低级语义重构出高级语义(操作系统级语义).语义重构的过程与客户操作系统的类型和版本密切相关,通过某些寄存器或者内存地址来解析出内核关键的数据结构.

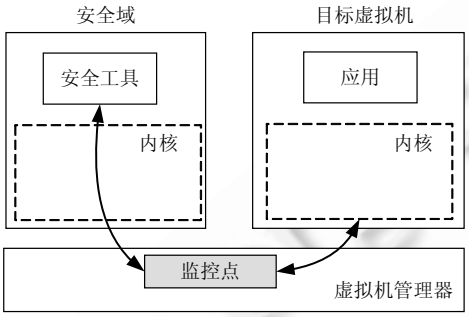


Fig.3 Architecture of external monitoring

图 3 外部监控的架构

2 基于虚拟化安全监控的相关工作

安全监控主要是对计算系统实现相应的安全功能,保证计算系统的正常运行.本文将主要从入侵检测、蜜罐、文件完整性监控、恶意代码检测与分析、安全监控架构、监控通用性等方面介绍相关的研究工作.

2.1 入侵检测

入侵检测系统(intrusion detection system,简称 IDS)[24,25]是指发现在非授权的情况下,试图存取信息、处理信息或者破坏系统,以使系统不可靠、不可用的故意行为的安全工具.根据收集信息的来源不同,IDS 可以分为基于网络的 IDS(network-based IDS,简称 NIDS)和基于主机的 IDS(host-based IDS,简称 HIDS)[26].NIDS 部署在局域网中,实时地分析网络中的流量(例如 Snort[27]);而 HIDS 则是分析系统的内部状态和日志,从而发现入侵行为(例如 OSSEC[28]).入侵防御系统(intrusion prevention system,简称 IPS)[29−31]则是在入侵检测的基础上实现动态响应.

现有的入侵检测架构给系统管理员带来了两难的选择:如果将入侵检测系统部署在主机上,则它可以清晰地观察到主机的系统状态,但是容易遭到恶意攻击或者被屏蔽;如果将入侵检测系统部署在网络上,则它可以更好地抵御攻击,但是对主机的内部状态一无所知,因此可能让攻击者逃脱.因此,Garfinkel 和 Rosenblum 首次在文献[23]中提出了一种能够观察到被监控系统的内部状态、同时与被监控系统隔离的入侵检测架构.该架构利用虚拟化技术,将入侵检测系统从被监控系统中转移出来.虚拟机管理器能够直接观察到被监控系统的内部状态,通过直接访问其内存来重构出客户操作系统的内核数据结构,而通过单独运行的入侵检测系统来进行检测.这种在虚拟机外部监控虚拟机内部运行状态的方法称为虚拟机自省(virtual machine introspection,简称 VMI).为了证明该架构能够抵御攻击以及防止攻击者逃脱,他们通过修改 VMware Workstation 实现了原型系统 Livewire[23],如图 4 所示.图 4 右侧为被监控的系统,左侧是基于虚拟机自信机制的入侵检测系统,操作系统接口库通过虚拟机管理管理器拦截的状态来恢复出操作系统级语义.

类似地,Laureano 等人[32,33]提出将被监控系统封装在虚拟机中,同时从外部监控其中的系统调用序列[34].外部的监控器根据系统调用序列来判断进程行为是否异常,从而采取相应的响应措施.基于 User-Mode

Linux(UML)[35],他们实现了原型系统,当检测到异常时,采用软件防火墙来阻断网络连接或者关闭网络端口.

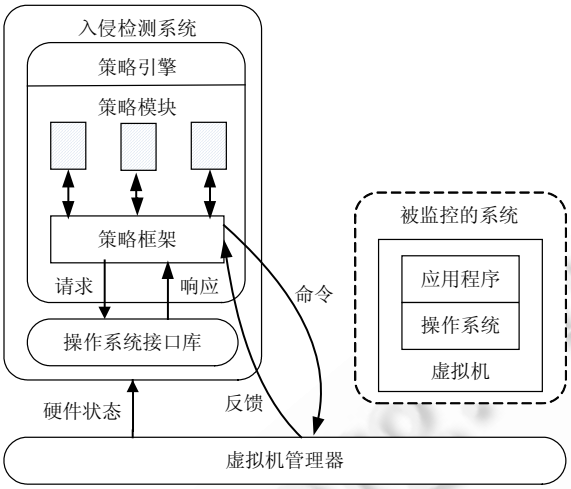


Fig.4 Architecture of Livewire[23]

图 4 Livewire 系统架构[23]

与上述方法不同,VNIDA[36]通过建立一个单独的入侵检测域(intrusion detection domain,简称 IDD)来为其他虚拟机提供入侵检测服务.虚拟机管理器层的事件传感器(event sensor)拦截虚拟机中的系统调用,并通过虚拟机管理器接口(VMM interface)传递到 IDD 中的入侵检测系统.根据不同的安全策略,虚拟机管理层的入侵检测域助手(IDD helper)能够针对入侵作出相应的响应.

此外,Zhang 等人[37]基于虚拟化和智能磁盘技术(smart disk technology)提出了面向存储的入侵检测系统.虚拟机管理器能够避免入侵检测系统受到攻击,智能磁盘技术则能够让入侵检测系统观测到被监控系统的整个文件系统.Pennington 等人[38]提出将入侵检测嵌入到网络文件系统(network file system,简称 NFS)服务器中.当客户端进行文件操作时,基于存储的入侵检测系统能够对可疑的文件修改操作进行在线的、基于规则的检测.

在分布式计算系统中,HyperSpector[39]是一种面向虚拟计算环境的入侵监控系统.分布式环境中的多个入侵检测系统能够发现攻击者,同时也增加了不安全的因素.因此,HyperSpector 将入侵检测系统部署在虚拟机中,与被监控的系统隔离.此外,各个节点上的入侵检测虚拟机(IDS VM)通过虚拟网络进行互联.为了对被监控系统进行有效地检测,它提供了 3 种虚拟机内部(inter-VM)监控机制:软件端口镜像(software port mirroring)、虚拟机内磁盘挂载(inter-VM disk mounting)和虚拟机内进程映射(inter-VM process mapping).此外,Roschke 等人[40]提出了面向分布式计算环境,集成了管理虚拟机和入侵检测系统的架构.该架构通过事件收集器(event gatherer)来获取各个 IDS 传感器(IDS sensor)的信息,并记录在事件数据库(event database)进行分析.因此,该架构满足通用分布式入侵检测系统可伸缩性(extensibility)的需求.

2.2 系统日志

系统日志(Log)[41]是记录系统中硬件、软件和系统的问题信息,同时还可以监视系统中发生的事件.系统管理员可以通过它来检查错误发生的原因,或者寻找攻击者留下的痕迹.例如,Linux 系统中,日志一般存放在文件 */var/log/messages* 中,而 Windows XP 系统中则可以运行 *eventvwr* 命令来查看系统日志.

当前的系统日志存在两个问题:

(1) 它依赖于记录日志的操作系统的完整性.如果入侵已经发生了,那么攻击者很有可能通过删除日志记录来隐藏其踪迹,这样,系统日志也就不再值得信赖了.

(2) 它没有记录充足的信息(例如非确定性事件)来重放和分析攻击.系统日志只记录系统中发生的事件,

而这些事件不能充分确定攻击者如何入侵到系统,因此给系统管理员分析攻击带来了极大的负担.

ReVirt[42]在虚拟机管理器层将客户操作系统的行为记录到系统日志中.它主要记录了影响进程运行的非确定性事件(non-deterministic event):外部输入和定时器.它根据日志记录来进行系统重放,从而分析攻击行为.

通过记录操作系统中发生的系统调用和进行系统重放,BackTracker[43,44]能够自动定位入侵发生的序列.通过在线记录系统日志,当入侵发生后,分析日志信息从而获得入侵过程中进程和文件的依赖关系图(dependency graph).为了在软件漏洞发布之前发现入侵,IntroVirt[45]利用虚拟机自省机制来监控操作系统和应用程序的执行过程.在系统运行或者回放过程中,通过执行特定的脆弱性断言(vulnerability-specific predictate)来检测入侵.与之类似,通过记录和回放系统中的相关事件,ExecRecorder[46]基于 Bochs 实现了全系统回放.

上述方法都是在系统运行时记录系统日志,在发生入侵后通过从某个检查点(checkpoint)回放系统来重现攻击行为.与上述方法不同的是,Xenlog[47]提出了一种安全地记录系统日志的方法.在获得被监控系统的日志之后,通过 Xen 提供的共享内存机制,将日志信息传输到一个单独的隔离域中.这种方法可以防止日志被攻击者嗅探到.同时,与网络传输日志的方式相比,它提高了传输效率.类似地,吴佳民等人[48]提出了一个运行于 Xen 上的实时日志备份系统.通过该系统来传输日志信息,可以避免其在传输过程中暴露在网络上的风险.

2.3 蜜 罐

蜜罐(honeypot)[49]是通过构造类似于真实的系统环境,从而引诱恶意攻击,并分析攻击者的行为特征.目前,蜜罐是研究最新恶意代码的有效手段,从中可以提取恶意代码的行为特征.根据传感器部署的位置,蜜罐可以分为内部蜜罐和外部蜜罐[50].内部蜜罐部署在被监控系统内部.它能够提供丰富的语义,但可能会被恶意攻击者所破坏.外部蜜罐部署在被监控系统外部.它对恶意攻击者透明,但是不能获取系统内部事件.

由于在物理机器上部署蜜罐通常耗时而且代价大,Honeyd[51]是一种模拟计算机的虚拟蜜罐框架.它不仅能够模拟不同操作系统的网络栈,而且可以为大量虚拟系统提供任意的路由拓扑(routing toplogy).通过部署 Honeyd 可应用于蠕虫检测、垃圾邮件防御等系统安全领域.与之类似,基于虚拟化技术,Potemkin[52]通过内存共享(memory sharing)和推迟绑定资源(late binding of resources)技术来支持在单个物理节点上同时模拟上万个高度逼真的蜜罐,从而增强了蜜罐的可扩展性(scalability).为了探测本地的蠕虫,HoneyStat[53]通过修改蜜罐来提高检测的准确性,同时具有低误报率.与入侵检测系统的报警信息相比,它可以提供更多的信息,例如二进制特征(binary signature)、攻击向量(attack vector)、攻击速率(attack rate)等.文献[54]比较了在 User-Mode Linux 和 Xen 上实现基于虚拟机的蜜罐,并通过实验表明,少量的传感器就能探测到大量的入侵行为.由于单个蜜罐或者多个独立的蜜罐只能发现攻击的局部视图,而且蜜罐的部署也会带来安全风险,Collapsar[55,56]通过管理专用网络中的大量高交互式虚拟蜜罐来建立分布式网络攻击拘留中心(network attack detention center).这种方法能够提供网络攻击的多样化视图,同时便于进行管理.

由于内部蜜罐容易受到攻击,而外部蜜罐无法探测到被监控系统的内部状态,VMscope[57]基于虚拟化实现了从蜜罐外部查看内部系统状态.由于客户端攻击急剧地增加,PhoneyC[58]通过模拟客户端的应用(例如网页浏览器)来分析最新的客户端攻击方式.针对网络中传播的恶意代码,Amun[59]在服务器端模拟系统漏洞,从而主动收集蠕虫[60]和僵尸网络[61]等恶意代码的行为特征.

2.4 完整性保护

完整性是指能够保障被传输、接收或存储的信息是完整的和未被篡改的.它是信息安全的重要属性之一[62].目前,关于完整性保护已经有一些相关的研究工作,主要集中在保证文件完整性、内核代码完整性和虚拟机管理器代码的完整性.

文件完整性保护主要分为周期性文件保护和实时文件保护:周期性文件保护是在不同的时间点来比较文件的 Hash 值,从而判断文件是否被篡改,例如 Tripwire[63],AIDE[64],Samhain[65]等;实时文件保护通过拦截文件操作从而保护其完整性,例如 I3FS[66].Quynh 等人[67,68]提出基于 Xen 虚拟机来实时地监控文件系统的完整性,并将操作文件的报告通过共享内存的方式保存在单独的虚拟机中.然而,文件监控器可能被屏蔽,同时对被监控系统

不具有透明性.RFIM[69]系统通过在虚拟机管理器层截获虚拟机中的系统调用,并对文件操作进行语义恢复.因此,RFIM 系统对被监控系统完全透明.

Copilot[70]通过一个运行在协处理器中的内核完整性监视器来探测篡改操作系统内核的恶意代码.它不需要修改被保护系统上运行的软件,即使内核完全被"攻陷"了,也能正常工作.Pioneer[71]是在不可信的主机上实现代码执行过程的可验证性.它不需要协处理器或扩展 CPU 架构.SecVisor[72]通过修改 Linux 来创建一个轻量级虚拟机,从而保护主机操作系统的完整性;KOP[73]则是通过映射动态的内核数据来保证内核完整性.此外,HUKO[74]是基于虚拟化的完整性保护系统,从而避免对操作系统内核进行不可信的扩展.

在虚拟化得到广泛应用的同时,虚拟机管理器自身的安全问题也不容忽视.由于虚拟机管理器是在真实硬件之上运行的软件,因此近年来也出现了保护虚拟机管理器完整性的相关文献.Hypersafe[75]的目标是使虚拟机管理器具有自保护性(self-protection).为了保证虚拟机管理器运行时的完整性,文献[75]中提出了两种技术:不可绕过的内存锁(non-bypassable memory lockdown)和受限制的指针索引(restricted pointer indexing).不可绕过的内存锁通过设置页表中的某些位(例如 NX,R/W,U/S,WP)来实现恶意程序修改时导致的缺页故障(page fault),而正常的页表更新则通过原子操作来实现.此外,通过构造控制流程图(control flow graph)来限制指针的位置,从而保证控制流完整性.不可绕过的内存锁保证了虚拟机管理器代码的完整性,不可绕过的内存锁和受限制的指针索引保证了虚拟机管理数据的完整性.不可绕过的内存锁直接通过扩展虚拟机管理器的内存管理模块来实现;受限制的指针索引则是扩展了开源的 LLVM 编译器来重新编译虚拟机管理器的代码.

HyperSentry[76]提出了对运行时虚拟机管理器的完整性度量框架.与现有系统都是保护特权软件不同的是,HyperSentry 不需要在被度量的目标底层引入更高特权级的软件,如果引入更高特权级的软件,则将引起恶意攻击者获取系统最高特权的竞赛.与之相反,HyperSentry 引入一种软件组件,它能够与虚拟机管理器隔离,从而实现对虚拟机管理器进行秘密的和实时的完整性度量.然而,秘密性是确保受到攻击虚拟机管理器不会隐藏攻击踪迹,实时性是完整性度量的必须条件.

HyperSentry 通过使用外部通道(intelligent platform management interface,简称 IPMI)来触发秘密地度量,并通过系统管理模式(system management mode,简称 SMM)来保护代码和关键数据.HyperSentry 的贡献是突破了系统管理模式的限制,提供了完整性度量代理.其主要功能包括:

(1) Hypervisor 的上下文信息;

(2) 完整的执行保护;

(3) 证明输出.

与之类似,HyperCheck[77]是基于硬件辅助的探测篡改框架,用于保证虚拟机管理器的完整性.HyperCheck 也是利用 CPU 的系统管理模式来检测被保护机器的状态,并发送给远程用户.除了 BIOS 以外,HyperCheck 不依赖于目标机器上运行的任何软件,而且能够抵御屏蔽和阻止其功能的恶意攻击.

2.5 恶意代码检测与分析

恶意代码[78]是指按照攻击者的意图在系统中执行的程序,一般具有隐蔽性,不易被监控工具发现.如果将检测工具都部署在被监控的系统中,则容易遭受到攻击.为了解决上述问题,VMwatcher[79]提出在被监控系统外部检测内部的恶意软件.由于检测工具与被监控系统隔离在不同的虚拟机中,因此需要解决语义鸿沟(semantic gap)的问题.通过语义重构,在虚拟机外部的检测软件能够发现虚拟机中的恶意软件.利用硬件辅助虚拟化技术,Ether[80]提出了一种外部的、透明的恶意代码分析方法.该方法利用 Intel VT 技术对可疑程序程序进行跟踪,同时对被监控系统完全透明.由于在虚拟机内部和外部进行观测时,两者之间具有一定的时间差,如果在这段时间内进程创建和撤销很频繁,则会对检测结果带来一定程度的误差.因此,Lycosid[81]系统利用统计学的方法消除了在两者之间观测带来的干扰,从而提高了检测的精确性.

现有的恶意软件检测机制都是探测恶意代码的存在,而不是动态地分析恶意代码的行为.K-Tracer[82]是基于 QEMU 开发的恶意代码行为分析工具.它能够动态地收集 Windows 内核的执行路径信息,并采用后向和前向的切片(backward and forward slicing)技术来提取恶意代码的行为.与之类似,Rkprofiler[83]是一个基于沙盒

(sandbox)的恶意代码分析系统.它能够监控和报告客户操作系统中运行的恶意代码的行为.由于分析恶意代码的行为是一个非常枯燥而且耗时的工作,Moser 等人[84]基于 QEMU 实现了多执行路径(multiple execution path)的恶意代码分析系统,从而识别出某些特定条件才会出现的恶意行为.这种方法能够在可疑行为发生时,自动地提取应用程序的全面行为.此外,Crandall 等人[85]提出基于虚拟机的虚拟时间来自动分析与时间相关的恶意代码.该方法不需要假定客户操作系统内核的完整性,同时能够探测恶意代码的定时炸弹(malware timebomb).Wang 等人[86]利用虚拟机中运行有漏洞的浏览器(vulnerable brower)来分析各个 Web 站点上可能存在的恶意软件,从而降低了为了保护 Internet 用户而监控恶意网站的开销.

2.6 安全监控架构

安全监控架构是指安全工具为了适应虚拟计算环境而采用不同的架构.Livewire[23]采用了虚拟机自省的监控架构,即在将安全工具放在单独的虚拟机中来对其他虚拟机进行检测.Xenaccess[87]是在 Xen 的管理域中实现的虚拟机监控库,它基于 Xen 提供的 *libxc* 和 *libblktap* 库.Xenaccess 提供了高级接口,并实现对目标虚拟机内存和磁盘查看.由于 Xenaccess 的前提假设是操作系统内核的完整性,当恶意攻击者篡改内核关键数据结构时,检测功能会失效.而且,磁盘监控具有一定的延迟,因此可能遭到定时攻击.Wizard[88]是一个基于 Xen 的内核监控器.它能够发现高级的内核事件和低级的硬件设备事件之间的相互关系.因此,它能够安全而有效地截获应用级和操作系统级行为.

上述系统都是采用虚拟机自省机制.与之不同的是,Lares[21]是基于虚拟化的安全主动监控(secure active monitoring)框架.它通过在被监控的虚拟机中插入一些钩子函数(hook),从而截获系统状态的改变,并跳转到单独的安全虚拟机中进行处理.因此,Lares 既进行主动监控,又通过隔离的虚拟机来提高安全性.SIM[22]则是利用硬件辅助虚拟化来实现虚拟机内部的通用安全监控框架.监控工具部署在不可信的客户操作系统中,并通过虚拟机管理器进行保护.由于事件截获在被监控的系统中,不存在语义鸿沟的问题,因此降低了监控开销.然而,由于钩子函数或者监控工具部署在被监控系统中,因此需要通过虚拟机管理器来保护其所在的内存页面.

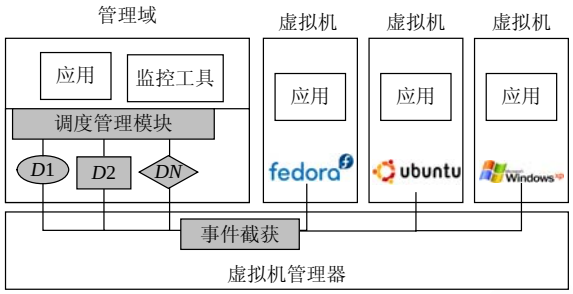
2.7 监控的通用性

由于单个物理节点上同时运行多个不同类型的虚拟机,虚拟机自身具有动态性,虚拟机可以动态地创建、撤销以及在各个物理平台之间进行迁移.传统的监控工具针对的目标系统是相对稳定的,而在虚拟化架构中,监控工具需要对各种虚拟机进行有效的监控.因此,现有的安全监控方法不能满足要求,需要采用通用的监控机制来保证有效性.

目前,所有上述监控工具都是针对特定的客户操作系统的类型来实现特定的安全功能.然而单个物理节点上虚拟机中的客户操作系统是多种多样的(例如 Linux,Windows 等).当某个物理节点上创建一个新的虚拟机,或者从另外一个物理节点上迁移新的虚拟机时,监控工具就会失效.因此,现有的监控工具不能满足监控通用性的需求.

文献[89]首次提出了基于虚拟化安全监控的通用性问题,同时提出了一种基于驱动的通用监控系统——VMDriver.VMDriver 通过两种新的设计策略来实现细粒度监控:(1) 将客户操作系统中的事件截获和语义恢复相分离.事件截获在虚拟机管理器层,而语义恢复在管理域中.(2) 语义恢复通过管理域中的监控驱动来实现.采用 Linux 中设备驱动的方式,语义恢复以内核模块的方式加载到管理域中,从而实现动态地屏蔽虚拟机中客户操作系统的差异性.监控驱动与虚拟机中客户操作系统的类型和版本相对应,并为上层的监控工具提供统一的接口,从而保证了监控系统的通用性.

图 5 说明了 VMDriver 系统的总体架构.事件截获模块位于虚拟机管理器层,语义恢复模块位于管理域的内核态.事件截获模块实现对虚拟机中客户操作系统发生的系统调用进行拦截.不同的形状(例如椭圆、矩形、菱形)分别代表了不同类型和版本的客户操作系统.语义恢复模块与虚拟机中客户操作系统的类型相对应,因此用不同的形状来表示.语义恢复模块以内核模块的方式进行加载,从而动态地对新的虚拟机进行有效的监控,因此也被称为监控驱动.调度管理模块只是对各种监控驱动进行管理,例如加载、卸载等.



DX:对应于不同类型操作系统的监控驱动*X*(*X*=1,2,...,*N*)

Fig.5 Architecture of VMDriver

图 5 VMDriver 系统架构

VMDriver 系统的工作流程:用户态监控工具需要对某个虚拟机进行监控,通过调度管理模块加载相应的监控驱动.当虚拟机中发生系统调用时,虚拟机管理器中的事件截获模块进行拦截,并向相应的监控驱动报告.监控驱动根据内核数据结构重构出被监控系统中的操作系统级语义,并传输给用户态监控工具.通过这种方式,监控驱动屏蔽了虚拟机中客户操作系统的差异性.

3 展 望

综上所述,现有的工作主要集中在利用虚拟机管理器来保护目标虚拟机中的钩子函数(内部监控)或者从目标虚拟机外部查看其内部状态(外部监控),并针对不同类型的安全工具都进行了相应的研究.然而,现有的研究工作主要存在以下两个方面的不足:

(1) 现有研究工作的前提条件过强,在实际情况中会存在某些问题

现有的研究工作都假定虚拟机管理器和管理域自身是安全的.事实上,虚拟机管理器是一层运行在真实硬件之上的系统软件,其自身也可能存在一定的安全漏洞.虽然与操作系统相比,虚拟机管理器由于其代码量少和功能简洁而具有较高的安全性,但是,CVE 网站已经公布出来的漏洞显示,VMware ESX 存在 23 个漏洞[90],Xen 存在 6 个漏洞[91],这与操作系统本身的漏洞相比而言是相当少的.但是,虚拟机管理器作为整个虚拟计算平台的基石,一旦其出现某种安全问题,就会导致整个虚拟化平台上的其他虚拟机遭到非常严重的后果,因此其危害性也就更大.此外,管理域是一个完整的系统环境(包括操作系统和应用程序),平台管理员可以利用管理域中的管理工具来操作虚拟机,例如创建、暂停、重启、迁移等.由于管理域比较庞大而且具有较高的权限,这也会给虚拟计算环境带来新的安全风险.例如,恶意攻击者通过入侵管理域从而控制整个虚拟化平台.相对于破坏虚拟机管理器,这种攻击方式难度更低而且不易发现.

(2) 安全监控与现有的安全工具的融合

现有的研究工作都是利用虚拟机管理器实现某种安全功能,例如入侵检测、恶意代码分析等.目前,学术界和工业界已经开发或使用了大量安全工具.这些安全工具都是基于系统级语义的.安全监控没有提供统一的接口标准,现有的安全工具不能对其直接调用.因此,如何将基于虚拟化的安全监控与现有的安全工具进行融合,是主要存在的问题.

我们认为,基于虚拟化的安全监控还有以下一些问题值得进行深入研究:

(1) 增强虚拟机管理器自身的安全性

现有针对虚拟机管理器的攻击主要是破坏其隔离性,从而导致虚拟机逃脱(VM escape),即从虚拟机中渗透到管理域中,从而影响到同一平台上的其他虚拟机.目前,为了保护虚拟机管理器自身的安全性,HyperSentry[76]和 HyperCheck[77]都是提出利用 CPU 的系统管理模式(system management mode,简称 SMM)来定期对虚拟机管理器进行扫描.这种方法将会引入对新的特权级的竞争,即虚拟机管理器和恶意代码都来争抢底层硬件的控制

权.因此,需要充分地研究如何保证虚拟机管理器自身的安全性.一方面,尽量简化虚拟机管理器的设计,即保证虚拟机管理器只实现底层硬件抽象接口的功能,降低其实现的复杂度;一方面,在整个虚拟化平台启动之前,利用可信计算[92]技术对虚拟机管理器进行度量,从而保证其完整性;另一方面,简化管理域的功能,管理域只实现管理和控制虚拟机的功能,而不提供其他具体服务(例如 Web 服务、文件服务).

(2) 基于监控信息来评估虚拟计算环境的安全风险

虽然不能保证虚拟计算环境100%的安全,但是安全监控能够观察到虚拟机的详细内部状态信息,因此根据全局监控信息可以对其进行风险评估.在分布式计算环境中,通常采用隐马尔可夫模型(hidden Markov model,简称 HMM)来对所有节点进行入侵风险评估[93].在虚拟计算环境中,所有物理机器上运行的虚拟机状态都是动态变化的(例如创建、暂停和迁移等),而且永远不会达到相对稳定的状态.现有的安全风险评估模型都是针对相对稳定的系统状态和网络环境,并不适合于虚拟计算环境.因此,需要建立新的理论模型,对现有的虚拟计算环境进行安全风险评估,并为应用虚拟化技术提供理论依据.

(3) 与现有安全工具的融合

在传统环境下,为了提高计算系统的安全性,研究者已经开发了大量的安全工具.在虚拟计算环境下,基于虚拟机管理器可以更好地监控虚拟机的内部运行状态.然而,虚拟机管理器获取的信息是二进制级的语义,传统的安全工具是无法直接使用的.因此,为了更好地利用已有的安全工具,基于虚拟化的安全监控需要与现有的安全工具进行有效的融合.一方面,利用语义恢复来实现从二进制级的语义转换为系统级语义,同时为安全工具提供标准的调用接口,从而使安全工具直接或者稍作修改来适应于虚拟计算环境;另一方面,语义恢复给安全工具带来了额外的性能开销,为了使安全监控具有更大的实用价值,研究者需要考虑在语义信息的全面性和系统开销之间进行综合权衡.

(4) 适应于云计算环境

云计算[94,95]作为一种新的计算模式和资源使用方式,深刻地改变了人们使用资源的方式.云服务提供商(cloud service provider,简称 CSP)将按照云用户需求为其资源分配和环境配置,从而免除了云用户购买硬件和系统维护的开销.虚拟化技术将云中的资源更加灵活地呈现给云用户,因此虚拟化在云平台中具有广泛的应用前景.然而,现有的安全监控机制不能直接应用于云平台:一方面,由于云计算具有不同的服务类型[96],包括软件即服务(software as a service,简称 SaaS)、平台即服务(platform as a service,简称 PaaS)、架构即服务(infrastructure as a service,简称 IaaS),需要研究针对不同的服务类型来对实现安全监控的方法;一方面,由于云平台同时为海量云用户提供服务,而各个云用户需要的系统环境可能是千差万别的,这在一定程度上增加了安全监控的复杂性,需要研究对各种系统环境实现通用的安全监控;另一方面,由于云用户的使用环境和数据都部署在云端,其中可能包含云用户的隐私信息,现有的技术手段不能保证云服务提供商是否提供承诺的服务质量或者窃取云用户的个人隐私信息,以及云中的恶意用户是否对其他正常的云用户发起攻击.因此,需要研究如何保证云平台监控信息的全面性和可信性.解决方法之一是,在云端建立独立于云服务提供商和云用户的可信第三方,对云端的服务进行有效的监控.除了获取云平台内部的全局状态信息以外,可信第三方主要的功能是:1) 它向云用户报告其服务的运行状态(例如服务使用的资源、内部状态等);2) 它还可以通过对全局的监控信息进行关联分析,从而及时发现云平台内部可能的攻击,从而提高服务云用户的质量.总而言之,云平台中的安全监控具有巨大的研究价值和应用前景.

4 结束语

近年来,基于虚拟化的安全研究成为热点.本文分别从入侵检测、蜜罐、文件完整性监控、恶意代码检测与分析、安全监控架构、监控通用性等方面介绍了基于虚拟化安全监控方面的研究现状,其核心思想都是利用虚拟机管理器的高特权级来实现某种安全功能.在虚拟机管理器的辅助下,安全工具将会更好地发挥其作用.然而,现有的研究工作都假定虚拟机管理器和管理域是安全的,在实际应用当中存在问题,同时还需要考虑与现有安全工具的融合.本文说明了现有研究工作的不足之处,同时指出了未来研究的发展趋势.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是华中科技大学服务计算技术与系统教育部重点实验室和集群与网格计算湖北省重点实验室(SCTS&CGCL)的老师和同学表示感谢.

References:

[1] Smith JE, Nair R. The architecture of virtual machines. IEEE Computer, 2005,38(5):32−38. [doi: 10.1109/MC.2005.173]

[2] Rosenblum M, Garfinkel T. Virtual machine monitors: Current technology and future trends. IEEE Computer, 2005,38(5):39−47. [doi: 10.1109/MC.2005.176]

[3] Whitaker A, Cox RS, Shaw M, Gribble SD. Rethinking the design of virtual machine monitors. IEEE Computer, 2005,38(5):57−62. [doi: 10.1109/MC.2005.169]

[4] Jin H, *et al*. Computer System Virtualization—Theory and Application. Beijing: Tsinghua University Press, 2008. 1−26 (in Chinese).

[5] IDC report. http://www.virtualization.info/2007/07/idc-predicts-virtualization-services.html

[6] Gartner report. http://www.gartner.com/it/page.jsp?id=777212

[7] Creasy RJ. The origin of the VM/370 time-sharing system. IBM Journal Research and Development, 1981,25(5):483−490. [doi: 10.1147/rd.255.0483]

[8] Popek G, Goldberg R. Formal requirements for virtualizable third generation architectures. Communications of the ACM, 1974, 17(7):413−421. [doi: 10.1145/361011.361073]

[9] Robin JS, Irvine CE. Analysis of the Intel Pentium's ability to support a secure virtual machine monitor. In: Proc. of the 9th USENIX Security Symp. Berkeley: USENIX Association, 2000. 129−144.

[10] Waldspurger CA. Memory resource management in VMware ESX server. In: Proc. of the 5th Symp. on Operating Systems Design and Implementaion. New York: ACM Press, 2002. 181−194.

[11] VMware homepage. http://www.vmware.com

[12] Xen homepage. http://www.xen.org

[13] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfiedld A. Xen and the art of virtualization. In: Proc. of the 19th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2003. 164−177. [doi: 10.1145/1165389.945462]

[14] Clark B, Deshane T, Dow E, Evanchik S, Finalyson M, Herne J, Matthews JN. Xen and the art of repeated research. In: Proc. of the 2004 USENIX Annual Technical Conf. Berkeley: USENIX Association, 2004. 135−144.

[15] Pratt I, Fraser K, Hand S, Limpach C, Warfield A. Xen 3.0 and the art of virtualization. In: Proc. of the 2005 Linux Symp. New York: ACM Press, 2005. 65−77.

[16] Nanda S, Chiueh T. A survey on virtualization technologies. Technical Report, TR-179, Stony Brook University, 2005. 1−42.

[17] Jin H. The key problem of cloud security. The Communications of the Computer Conf. Foundation, 2009,5(2):47−48 (in Chinese with English abstract).

[18] The linecount of Linux 2.6.27. http://doexcel.com/node/6572

[19] The linecount of Windows XP. http://blog.csdn.net/syf442/article/details/4459229

[20] Chen PM, Noble BD. When virtual is better than real. In: Proc. of the 8th Workshop on Hot Topics in Operating Systems. Washington: IEEE Computer Society, 2001. 133−138.

[21] Payne BD, Carbone M, Sharif M, Lee W. Lares: An architecture for secure active monitoring using virtualization. In: Proc. of the 29th IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 2008. 233−247. [doi: 10.1109/SP.2008.24]

[22] Sharif M, Lee W, Cui W, Lanzi A. Secure in-VM monitoring using hardware virtualization. In: Proc. of the 16th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2009. 477−487. [doi: 10.1145/1653662.1653720]

[23] Garfinkel T, Rosenblum M. A virtual machine introspection based architecture for intrusion detection. In: Proc. of the 10th Network and Distributed System Security Symp. Berkeley: USENIX Association, 2003. 191−206.

[24] Lian Y. Survey on intrusion detection. Network Security Technology and Application, 2003,(1):46−48 (in Chinese with English abstract).

[25] Axelsson S. Intrusion detection systems: A survey and taxonomy. Technical Report, 99-15, Department of Computer Engineering, Chalmers University of Technology, 2000. 1−27.

[26] Sun J. Research on intrusion detection and antivirus model in large-scale network [Ph.D. Thesis]. Wuhan: Huazhong University of Science and Technology, 2005 (in Chinese with English abstract).

[27] Roesch M. Snort-Lightweight intrusion detection for networks. In: Proc. of the 13th USENIX Large Installation System Administration Conf. Berkeley: USENIX Association, 1999. 229−238.

[28] Hay A, Cid D, Bray R. OSSEC Host-Based Intrusion Detection Guide. Burlington: Syngress Press, 2008. 149−174.

[29] Fuchsberger A. Intrusion detection systems and intrusion prevention systems. Information Security Technical Report, Elsevier, 2005. 134−139.

[30] Locasto ME, Wang K, Keromytis AD, Stolfo SJ. FLIPS: Hybrid adaptive intrusion prevention. In: Proc. of the 8th Int'l Symp. on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2005. 82−101. [doi: 10.1007/11663812_5]

[31] Lam LC, Li W, Chiueh T. Accurate and automated system call policy-based intrusion prevention. In: Proc. of the 2006 Int'l Conf. on Dependable Systems and Networks. Washington: IEEE Computer Society, 2006. 413−424. [doi: 10.1109/DSN.2006.10]

[32] Laureano M, Maziero C, Jamhour E. Intrusion detection in virtual machine environments. In: Proc. of the 30th Euromicro Conf. Washington: IEEE Computer Society, 2004. 520−525. [doi: 10.1109/EUROMICRO.2004.48]

[33] Laureano M, Maziero C, Jamhour E. Protecting host-based intrusion detectors through virtual machines. The Journal of Computer and Telecommunications Networking, 2007,51(5):1275−1283. [doi: 10.1016/j.comnet.2006.09.007]

[34] Hofmeyr S, Forrest S, Somayaji A. Intrusion detection using sequences of system calls. Journal of Computer Security, 1998,6(3): 151−180.

[35] Dike J. User-Mode Linux. In: Proc. of the 5th Annual Linux Showcase and Conf. Berkeley: USENIX Association, 2001. 21−28.

[36] Zhang X, Li Q, Qing S, Zhang H. VNIDA: Building an IDS architecture using VMM-based non-intrusive approach. In: Proc. of the 14th Int'l Workshop on Knowledge Discovery and Data Mining. New York: ACM Press, 2008. 594−600. [doi: 10.1109/WKDD. 2008.135]

[37] Zhang Y, Gu Y, Wang H, Wang D. Virtual-Machine-Based intrusion detection on file-aware block level storage. In: Proc. of the 18th Int'l Symp. on Computer Architecture and High Performance Computing. Washington: IEEE Computer Society, 2001. 21−28. [doi: 10.1109/SBAC-PAD.2006.32]

[38] Pennington AG, Strunk JD, Griffin JL, Soules CAN, Goodson GR, Ganger GR. Storage-Based intrusion detection: Watching strorage activity for suspicious behavior. In: Proc. of the 12th USENIX Security Symp. Berkeley: USENIX Association, 2003. 1−15.

[39] Kourai K, Chiba S. HyperSpector: Virtual distributed monitoring environments for secure intrusion detection. In: Proc. of the 1st ACM Int'l Conf. on Virtual Execution Environments. New York: ACM Press, 2005. 197−207. [doi: 10.1145/1064979.1065006]

[40] Roschke S, Cheng F, Meinel C. An extensible and virtualization-compatible IDS management architecture. In: Proc. of the 5th Int'l Conf. on Information Assurance and Security. Washington: IEEE Computer Society, 2009. 130−134. [doi:10.1038/nchembio0309-130]

[41] Yang Z, Deng X. A behavior mining and pattern recognition for operation system Log. Computer and Information Technology, 2010,18(1):53−55 (in Chinese with English abstract).

[42] Dunlap GW, King ST, Cinar S, Basrai MA, Chen PM. ReVirt: Enabling intrusion analysis through virtual-machine logging and replay. In: Proc. of the 5th Symp. on Operating Systems Design and Implementation. New York: ACM Press, 2002. 211−224. [doi: 10.1145/844128.844148]

[43] King ST, Chen PT. Backtracking intrusions. In: Proc. of the 19th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2003. 223−236. [doi: 10.1145/945445.945467]

[44] King ST, Chen PT. Backtracking intrusions. ACM Trans. on Computer Systems, 2005,23(1):51−76. [doi: 10.1145/945445.945467]

[45] Joshi A, King ST, Dunlap GW, Chen PM. Detecting past and present intrusions through vulnerability-specific predicates. In: Proc. of the 20th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2005. 91−104. [doi: 10.1145/1095809.1095820]

[46] Oliveira DAS, Crandall JR, Wassermann G, Wu SF, Su Z, Chong FT. ExecRecorder: VM-Based full-system replay for attack analysis and system recovery. In: Proc. of the 1st Workshop on Architectural and System Support for Improving Software Dependability. New York: ACM Press, 2006. 66−71. [doi: 10.1145/1181309.1181320]

[47] Quynh NA, Takefuji Y. A central and secured log data solution for Xen virtual machine. In: Proc. of the 24th Int'l Multi-Conf. Parallel and Distributed Computing and Networks. Washington: IEEE Computer Society, 2006. 218−224.

[48] Wu J, Peng X, Gao D. On Xen virtual machine-based system Logs security. Computer Applications and Software, 2010,27(4): 125−127 (in Chinese with English abstract).

[49] Spitzner L. Honeypots: Tracking Hackers. New York: Addison Wesley Press, 2002. 1−35.

[50] Tang Y, Lu X, Hu H, Zhu P. Honeypot technique and its applications: A survey. Journal of Chinese Computer Systems, 2007,28(8): 1345−1351 (in Chinese with English abstract).

[51] Provos N. A virtual honeypot framework. In: Proc. of the 13th USENIX Security Symp. Berkeley: USENIX Association, 2004. 1−14.

[52] Vrable M, Ma J, Chen J, Moore D, Vandekieft E, Snoeren AC, Voelker GM, Savage S. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. In: Proc. of the 20th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2005. 148−162. [doi: 10.1145/1095810.1095825]

[53] Dagon D, Qin X, Gu G, Lee W. HoneyStat: Local worm detection using honeypots. In: Proc. of the 7th Int'l Symp. on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2004. 39−58.

[54] Asrigo K, Litty L, Lie D. Using VMM-based sensors to monitor honeypots. In: Proc. of the 2nd ACM Int'l Conf. on Virtual Execution Environments. New York: ACM Press, 2006. 13−23. [doi: 10.1145/1134760.1134765]

[55] Jiang X, Xu D. Collapsar: A VM-based architecture for network attack detention center. In: Proc. of the 13th USENIX Security Symp. Berkeley: USENIX Association, 2005. 15−28.

[56] Jiang X, Xu D, Wang Y. Collapsar: A VM-based honeyfarm and reverse honeyfarm architecture for network attack capture and detention. Journal of Parallel and Distribtued Computing, 2006,66(9):1165−1180. [doi: 10.1016/j.jpdc.2006.04.012]

[57] Jiang X, Wang X. "Out-of-the-Box" monitoring of VM-based high-interaction honeypots. In: Proc. of the 10th Int'l Symp. on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2007. 198−218.

[58] Nazario J. PhoneyC: A virtual client honeypot. In: Proc. of the 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats. Berkeley: USENIX Association, 2009. 31−36.

[59] Gobel J. Amun: A python honeypot. Technical Report, TR-2009-008, Laboratory for Dependable Distributed Systems, University of Mannheim, 2009. 1−14.

[60] Li P, Salour M, Su X. A survey of Internet worm detection and containment. IEEE Communications Surveys and Tutorials, 2008, 10(1):20−35. [doi: 10.1109/COMST.2008.4483668]

[61] Zhu Z, Lu G, Chen Y. Botnet research survey. In: Proc. of the 32nd Annual IEEE Int'l Computer Software and Applications Conf. Washington: IEEE Computer Society, 2008. 967−972. [doi: 10.1109/COMPSAC.2008.205]

[62] Hong F, Cui G, Fu X. Introduction to Information Security. Wuhan: Huazhong University of Science and Technology Press, 2005, 1−21 (in Chinese).

[63] Kim GH, Spafford EH. The design and implementation of tripwire: A file system integrity checker. In: Proc. of the 2nd ACM Conf. on Computer and Communications Security. New York: ACM Press, 1994. 18−29. [doi: 10.1145/191177.191183]

[64] AIDE. http://aide.sourceforge.net/

[65] Samhain. http://www.la-samhna.de/samhain/

[66] Patil S, Kashyap A, Sivathanu G, Zadok E. I^{3}FS: An in-kernel integrity checker and intrusion detection file system. In: Proc. of the 18th USENIX Large Installation System Administration Conf. Berkeley: USENIX Association, 2004. 67−78.

[67] Quynh NA, Takefuji Y. A novel approach for a file-system integrity monitor tool of Xen virtual machine. In: Proc. of the 2nd ACM Symp. on Information, Computer and Communications Security. New York: ACM Press, 2007. 194−203. [doi: 10.1145/1229285.1229313]

[68] Quynh NA, Takefuji Y. A real-time integrity monitor for Xen virtual machine. In: Proc. of the 2006 Int'l Conf. on Networking and Services. Washington: IEEE Computer Society, 2006. 90−95. [doi: 10.1109/ICNS.2006.13]

[69] Jin H, Xiang G, Zou D, Zhao F, Li M, Yu C. A guest-transparent file integrity monitoring method in virtualization environment. The Journal of Computers and Mathematics with Applications, 2010,60(2):256−266. [doi: 10.1016/j.camwa.2010.01.007]

[70] Petroni NL, Fraser T, Molina J, William AA. Copilot—A coprocessor-based kernel runtime integrity monitor. In: Proc. of the 13th Conf. on USENIX Security Symp. Berkeley: USENIX Association, 2004. 179−194.

[71] Seshadri A, Luk M, Shi E, Perrig A, Doorn L, Khosla P. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In: Proc. of the 20th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2005. 1−16. [doi: 10.1145/1095809.1095812]

[72] Seshadri A, Luk M, Qu N, Perrig A. SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. In: Proc. of the 21st ACM Symp. on Operating Systems Principles. New York: ACM Press, 2007. 335−350. [doi: 10.1145/1294261.1294294]

[73] Carbone M, Cui W, Lu L, Lee W, Peinado M, Jiang X. Mapping kernel objects to enable systematic integrity checking. In: Proc. of the 17th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2009. 555−565. [doi: 10.1145/1653662.1653729]

[74] Xiong X, Tian D, Liu P. Practical protection of kernel integrity for commodity OS from untrusted extensions. In: Proc. of the 18th Annual Network and Distributed System Security Symp. Rosten: Internet Society, 2011. 114−130.

[75] Wang Z, Jiang X. HyperSafe: A lightweight approach to provide lifetime hypervisor control-flow integrity. In: Proc. of the 31st IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 2010. 380−395. [doi: 10.1109/SP.2010.30]

[76] Azab AM, Ning P, Wang Z, Jiang X, Zhang X, Skalsky N. HyperSentry: Enabling stealthy in-context measurement of hypervisor integrity. In: Proc. of the 17th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2010. 38−49. [doi: 10.1145/1866307.1866313]

[77] Wang J, Stavrou A, Ghosh A. HyperCheck: A hardware-assisted integrity monitor. In: Proc. of the 13th Int'l Symp. on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2010. 158−177.

[78] Jacob G, Debar H, Filiol E. Behavioral detection of malware: From a survey towards an established taxonomy. Journal in Computer Virology, 2008,4(3):251−266. [doi: 10.1007/s11416-008-0086-0]

[79] Jiang X, Wang X, Xu D. Stealthy malware detection through VMM-based "out-of-the-box" semantic view reconstruction. In: Proc. of the 14th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2007. 128−138. [doi: 10.1145/1698750.1698752]

[80] Dinaburg A, Royal P, Sharif M, Lee W. Ether: Malware analysis via hardware virtualization extensions. In: Proc. of the 15th ACM Conf. on Computer and Communications Security. New York: ACM Press, 2008. 51−62. [doi: 10.1145/1455770.1455779]

[81] Jones ST, Arpaci-Dusseau AC, Arpaci-Dusseau RH. VMM-Based hidden process detection and identification using lycosid. In: Proc. of the 4th ACM Int'l Conf. on Virtual Execution Environments. New York: ACM Press, 2008. 91−100. [doi: 10.1145/1346256.1346269]

[82] Lanzi A, Sharif M, Lee W. *K*-Tracer: A system for extracting kernel malware behavior. In: Proc. of the 16th Annual Network and Distributed System Security Symp. Rosten: Internet Society, 2009. 191−203.

[83] Xuan C, Copeland J, Beyah R. Toward revealing kernel malware behavior in virtual execution environments. In: Proc. of the 12th Int'l Symp. on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2009. 304−325. [doi: 10.1007/978-3-642-04342-0_16]

[84] Andreas M, Christopher K, Engin K. Exploring multiple execution paths for malware analysis. In: Proc. of the 28th IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society, 2007. 231−245. [doi: 10.1109/SP.2007.17]

[85] Crandall JR, Wassermann G, Oliveira DA, Su Z, Wu SF, Chong FT. Temporal search: Detecting hidden malware timebombs with virtual machines. In: Proc. of the 12th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2006. 25−36. [doi: 10.1145/1168857.1168862]

[86] Wang Y, Beck D, Jiang X, Roussev R. Automated Web patrol with strider HoneyMonkeys: Finding Web sites that exploit browser vulnerabilities. In: Proc. of the 13th Network and Distributed Systems Security Symp. Rosten: Internet Society, 2006, 1−15.

[87] Payne BD, Carbone MA, Lee W. Secure and flexible monitoring of virtual machines. In: Proc. of the 23rd Annual Computer Security Applications Conf. New York: ACM Press, 2007. 385−397.

[88] Srivastava A, Singh K, Giffin J. Secure observation of kernel behavior. Technical Report, GT-CS-08-01, Georgia Institute of Technology, 2008. 1−14.

[89] Xiang G, Jin H, Zou D, Zhang X, Wen S, Zhao F. VMDriver: A driver-based monitoring mechanism for virtualization. In: Proc. of the 29th Int'l Symp. on Reliable Distributed Systems. Washington: IEEE Computer Society, 2010. 72−81. [doi: 10.1109/SRDS.2010.38]

[90] VMware ESX security vulnerabilities. http://www.cvedetails.com/vulnerability-list.php?vendor_id=252&product_id=14180&version_id=&page=1&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=0&month=0&cweid=0&order=1&trc=23&sha=8efab6ff7074db7dae27dee5850bcd63488694c6

[91] Xen security vulnerabilities. http://www.cvedetails.com/vulnerability-list.php?vendor_id=6276&product_id=&version_id=&page=1&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=0&month=0&cweid=0&order=1&trc=6&sha=7354f1cd84d744aba90e37868d68b6095ad317f5

[92] David C, Kent Y, Ryan C, David S, Leendert VD. A Practical Guide to Trusted Computing. Boston: Pearson Education, 2008. 3−10.

[93] Arnes A, Valeur F, Vigna G, Kemmerer RA. Using hidden Markov models to evaluate the risks of intrusions. In: Proc. of the 9th Int'l Symp. on Recent Advances in Intrusion Detection. Berlin: Springer-Verlag, 2006. 145−164. [doi: 10.1007/11856214_8]

[94] Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. Above the clouds: A Berkeley view of cloud computing. Technical Report, UCB/EECS-2009-28, University of California at Berkeley, 2009. 1−23.

[95] Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In: Proc. of the 2008 Grid Computing Environments Workshop. Washington: IEEE Computer Society, 2008. 1−10. [doi: 10.1109/GCE.2008.4738445]

[96] Mather T, Kumaraswamy S, Latif S. Cloud Security and Privacy. Sebastopol: O'Reilly Press, 2009. 109−144.

附中文参考文献:

[4] 金海,等.计算系统虚拟化-原理与应用.北京:清华大学出版社,2008.1−26.

[17] 金海.浅析云计算安全的核心问题.中国计算机学会通讯,2009,5(2):47−48.

[24] 连一峰.入侵检测综述.网络安全技术与应用,2003,(1):46−48.

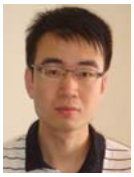
[26] 孙建华.大规模网络中的入侵检测与病毒防御模型研究[博士学位论文].武汉:华中科技大学,2005.

[41] 杨震宇,邓晓衡.操作系统日志的行为挖掘与模式识别.电脑与信息技术,2010,18(1):53−55.

[48] 吴佳民,彭新光,高丹.基于 Xen 虚拟机的系统日志安全研究.计算机应用与软件,2010,27(4):125−127.

[50] 唐勇,卢锡城,胡华平,朱培栋.Honeypot 技术及其应用研究综述.小型微型计算机系统,2007,28(8):1345−1351.

[62] 洪帆,崔国华,付小青.信息安全概论.武汉:华中科技大学出版社,2005.1−21.



项国富(1984－),男,湖北麻城人,博士,主要研究领域为虚拟化技术,基于虚拟化技术的安全监控,计算机系统结构.

邹德清(1975－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为虚拟化技术,可信计算,系统安全,安全评估,网格计算以及集群与高性能计算.

金海(1966－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机系统结构,虚拟化技术,集群计算,网格计算,并行与分布式计算,对等计算,普适计算,语义网,存储与安全.

陈学广(1947－),男,教授,博士生导师,主要研究领域为信息系统工程,决策支持系统,协商/协调理论与方法.