文章编号:1009-3087(2014)01-0022-07

VirtinSpector:一种基于 UEFI 的虚拟机动态安全度量框架设计与实现

严 飞,石 翔,李志华,王 鹃,张焕国
(武汉大学 计算机学院,湖北 武汉 430072)

摘 要:通过可信硬件能够弥补单纯软件安全的不足,从整体上提高云系统的安全性。但是,面对云环境运行时的安全,传统可信硬件技术无法提供足够的保障。为此,提出了一种基于 UEFI 的虚拟机动态安全框架——VirtinSpector。该框架能够将 UEFI 固件作为可信基础,对云系统的基础设施层进行实时、动态的安全度量,提供传统可信技术无法达到的动态保护。在此框架基础上,以某国产服务器为实验平台,构建云环境,实现了一个面向 Xen 环境的 UEFI 虚拟机动态安全度量原型系统。实验与分析表明,该框架能够有效检测针对虚拟域、管理域和虚拟化软件的攻击,为云系统提供来自基础设施层的安全支撑。并且对原有系统的性能损耗在允许范围之内,不影响用户的正常使用。

关键词:云安全;可信计算;动态度量;虚拟化
中图分类号:TP309 **文献标志码**:A

VirtinSpector: A UEFI Based Dynamic Secure Measurement Framework for Virtual Machine

YAN Fei, SHI Xiang, LI Zhi-hua, WANG Juan, ZHANG Huan-guo
(School of Computer, Wuhan Univ., Wuhan 430072, China)

Abstract: Trusted computing technology has been introduced to build a secure cloud infrastructure, which can improve the dilemma of software security. However, traditional trusted hardware technology does not provide sufficient protection for runtime security for cloud. To solve this problem, a UEFI based dynamic security framework for virtual machine, named VirtinSpector, was proposed. The VirtinSpector treated UEFI firmware as a trusted computing base to acquire a run-time and dynamic security measurement for cloud, providing a dynamic protection out of traditional solution. And a prototype of VirtinSpector for the Xen hypervisor was implemented. The experiments showed that the framework can measure and explore some mainstream attacks of cloud, and its performance payload is restricted whining an acceptable range, without affecting the user's daily use.

Key words: cloud security; trusted computing; dynamic measurement; virtualization

云计算已成为信息技术领域的发展热点。国际各大 IT 企业都在大力开发和推进云计算。但是,近年来针对云环境的安全事件层出不穷:2011 年,谷歌邮箱数据泄露,影响了约 15 万的 Gmail 用户;2013 年,Evernote 遭到入侵,威胁了约 5 千万用户的数据安全。这极大影响了服务提供商与云用户对云计算的应用信心。据 Gartner 的分析报告称,云安全是阻碍云计算发展的关键因素。

为了从技术上解决云安全问题,工业界和学术界都投入了极大兴趣,诸多安全手段被引入该领域,如 Pioneer[1]、NFORCE 系统[2]和语义完整性监控[3]等。但是,这类软件为了降低部署成本,提高语义理解能力,往往运行于虚拟机中,其安全性依赖于宿主本身。对于一些利用虚拟化软件对虚拟机发起的攻击(如:硬件虚拟化 Rootkit[4]),此类手段往往效果不佳。也有研究提出通过虚拟化软件对虚拟机进行保护,如 HIMA[5]、REDAS[6]、Lares[7]和 Patagonix[8]。此类手段能够检测到一些从虚拟机外部对虚拟机进行的攻击。但此类保护软件不能确保其自身的运行是安全的。为此,一些研究,如:Dyad[9]、BITS[10]、AEGIS[11]、HyperSentry[12]、SecureSwitch[13]

收稿日期:2013-06-20
基金项目:国家自然科学基金资助项目(61003268;61272452;91118003;61173138;61003185)
作者简介:严 飞(1980—),男,副教授,博士. 研究方向:信息系统安全;可信计算. E-mail:yanfei@whu.edu.cn

等,将可信计算等硬件安全手段引入云计算领域,利用安全硬件的抗篡改特性,对云系统进行安全保护。然而,由于传统硬件安全的局限性,现有的安全解决方案,更侧重于通过硬件来确保系统启动环境的静态安全性,即允许被度量过的代码被加载,或者允许系统加载到一个受到启动度量的环境中。而在云环境中,系统更需要运行时的安全保障。由于连续业务提供的要求,云基础设施中的服务器、网络存储等设备均不会频繁的启动,因此系统的启动加载度量模块在此环境中发挥作用的场景较少,这样就极大约束了安全硬件发挥其抗篡改能力的特性。

UEFI(统一的可扩展固件接口,unified extensible firmware interface),是一种详细描述全新类型接口的标准,利用加载 EFI 驱动的形式,识别并操作硬件,突破了传统 BIOS 的空间限制并改善了可移植性[14]。UEFI 较传统 BIOS 而言更易于实现,为安全地访问系统中的安全硬件、实施动态的安全保护提供可能。作为计算机系统的基础固件,UEFI 在实时运行的情况下,从结构和技术上保证了其程序是不被篡改的。首先,UEFI 程序的存储不能在运行时被轻易篡改。其程序以固件的形式存在,通过专用工具刷入 UEFI ROM 中。该 ROM 是被保护的,不会被恶意程序轻易篡改。其次,UEFI 程序运行时不被篡改。在引导 OS 的时,UEFI 驱动和应用会依次被释放到内存中进行加载和运行,但当要引导操作系统时,绝大部分功能将消失,只留下很少的一部分作为操作系统必须的 runtime services 依然在运行,且大部分服务基于 SMM(系统管理模式,system management mode)运行,而 SMM 中的内存 SMRAM 的修改权限在系统启动时,已经被硬件锁定,不能被篡改,因此能保证运行态下 UEFI 的安全性[15]。

为此提出了一种基于 UEFI 的虚拟机动态安全框架 VirtinSpector,该框架基于 UEFI 固件,在系统运行时提供对虚拟机环境实时的安全性保护,从而防御在系统运行时的多类攻击。根据此框架,还实现了一个基于 UEFI 的虚拟机动态度量原型系统,通过攻击测试和性能测试验证了框架的安全性与实用性。

1　前提条件与面临的威胁

1.1　前提条件

所提出的框架,基于以下假设前提:

1)本框架为动态保护框架,保护时间点为系统运行时,防御系统运行时的攻击。而系统启动时的攻击,通过可信硬件手段能够相对容易克服,如 AEGIS[11]、Tboot,因此该内容不属于本文重点考虑范畴。

2)本框架涉及硬件部件的自身安全性是可以保障的,包括可信平台模块 TPM、平台固件 UEFI BIOS 所存储的物理介质。在框架实施中,系统的初始化过程处于一个安全可信的环境中,在未获得授权时,不可对可信硬件以及 UEFI BIOS 等固件执行刷新操作。

3)所提出的框架各安全模块自身所涉及的代码不考虑存在安全漏洞。由于代码自身的安全缺陷是一个不可彻底克服的问题,当前针对嵌入式系统、固件的安全性成为研究热点。因此对此问题暂不探讨。同时,在设计和实现阶段,框架中各模块没有植入恶意代码和后门软件。

1.2　面临的威胁

在虚拟化环境中,威胁是多方面的,主要关注破坏完整性的攻击,而对于 DOS 攻击、ARP 攻击等影响性能与欺骗性的攻击,不在本文考虑范围之内。

1.2.1　虚拟机威胁

对虚拟机的威胁主要包括传统的攻击和以虚拟化软件为基础的攻击。传统的攻击直接对虚拟化操作系统本身发动,利用系统漏洞修改操作系统的关键数据段或程序段,进行 rootkit,添加后门。以虚拟化软件为基础的攻击,主要利用硬件虚拟化技术。硬件虚拟化的 CPU 运行于 non-root 状态和 root 状态。non-root 状态为虚拟机运行状态,root 状态为虚拟化软件运行状态。攻击者通过攻击虚拟化软件,修改其在 root 状态下的运行程序,实现对虚拟机的攻击[5]。

1.2.2　虚拟化软件威胁

目前,对虚拟化软件的威胁已经存在。此类攻击直接针对虚拟化软件,其危害极大,一旦攻击者获得虚拟化软件的控制权,将有能力控制整个被攻破的虚拟化软件所虚拟的运行环境。Rutkowska 展示了一种利用 DMA 攻击虚拟化软件 Xen 的例子[16]。该方法通过特殊的手段,能让 DMA 控制物理内存,利用 DMA 对内存进行修改,在 Xen 的程序中添加恶意后门模块,以此获得对获得 Xen 的控制权,并启动一个隐形的管理域,对整个虚拟化软件实施恶意控制。

1.2.3　UEFI 威胁

UEFI 大大提高了硬件编程的灵活性,使得普通程序开发者经过简单培训也能进行相关开发。但其

灵活性与开放性使其面临着巨大的安全威胁。2008年的黑帽大会上,Heasman 等总结了对 EFI 的攻击手段。通过篡改 bootloader、NVRAM、重新刷新硬件、攻击驱动的实现流等静态手段发动攻击[17]。此类攻击的准备阶段是在系统运行时(对 bootloader、NVRAM 等部分进行篡改),而发起实际攻击是在系统启动时(运行不可信的 UEFI BIOS)。由于可信计算的信任链技术特别适合于系统启动时的安全,因此通过与可信计算技术进行有效融合,能够有效的遏制该类攻击的发生。此外,所提出的框架,也能够有效遏制在运行时对诸如 bootloader、NVRAM 等部分的篡改攻击。

2 安全架构

针对以上安全要求,提出一个名为 VirtinSpector 的基于 UEFI 的虚拟机动态安全框架(图1),该框架分为管理模块,调度模块,Hypervisor 安全模块和 UEFI 固件安全模块。其中,管理模块在管理域中,调度模块和 Hypervisor 安全模块在虚拟化软件(Hypervisor)中,UEFI 固件安全模块在硬件与上层 Hypervisor 接口的 UEFI 接口层。通过图中的调用关系进行相互调用。而对于虚拟域来说,整个框架是透明的。

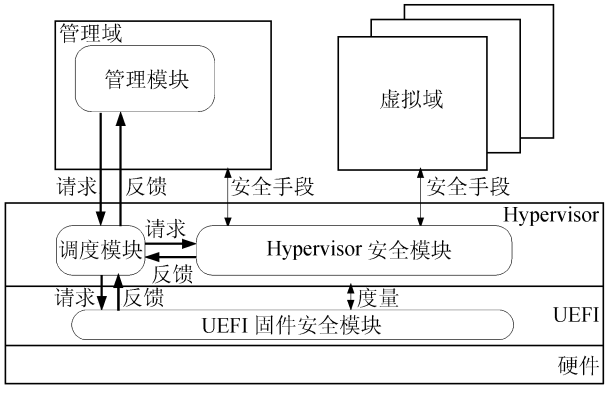

图1 VirtinSpector 框架

Fig. 1 Framework of VirtinSpector

2.1 功能模块的描述

2.1.1 UEFI 固件安全模块

此模块在 UEFI 层中,由于 UEFI 的设计和实现安全要求,其被篡改的可能性较低,可视为本框架的可信根。在本框架中,可通过一定的安全机制,并与相关安全硬件配合(如:TPM),实现对上层 Hypervisor 的保护。对于没有 TPM 的环境,类似 TPM 的安全机制也可通过在 UEFI 中植入软件模拟获得。此模块具有调用安全硬件接口,自动度量其它模块安全状态的功能。

2.1.2 Hypervisor 安全模块

此模块在虚拟化软件中,由可信根 UEFI 固件安全模块来确保其安全性。此模块根据一定的安全机制,设置安全手段对管理域和虚拟域实施安全保护。此模块的实现策略可与 UEFI 固件安全模块不同,具有自动度量与验证管理域和虚拟域运行态安全性的功能。

2.1.3 调度模块

此模块在虚拟化软件中,被可信根 UEFI 固件安全模块证明其安全性。此模块能够提供安全的接口供管理模块向 UEFI 固件安全模块和 Hypervisor 安全模块发送命令请求,并且能够提供安全的接口供 UEFI 固件安全模块和 Hypervisor 安全模块对管理模块进行反馈。

2.1.4 管理模块

此模块在管理域中,被可信的 Hypervisor 安全模块证明其安全性。根据一定的安全策略,该模块能实现管理功能。此模块根据底层模块对各个虚拟域的安全信息的反馈,对反馈信息进行分析管理,方便管理员了解各个虚拟域的运行情况。并根据相应情况,通过命令请求的方式,对底层模块进行操作。

2.2 模块运行方式

在开机过程后,信任根 UEFI 固件安全模块立即通对 Hypervisor 进行安全检查,判断 Hypervisor 安全模块的安全性和调度模块的安全性。在验证2个模块的安全性后,Hypervisor 安全模块立即对管理域进行安全检查,判断管理模块的安全性,并据此使 UEFI 固件安全模块能够判断管理模块的安全性。在系统运行过程中,需要实时地进行上述过程,动态地验证整个框架,确保实时的安全性。并且当某特定动作发生前,需要进行安全验证。首先,当管理模块需要触发 UEFI 固件安全模块或 Hypervisor 安全模块前,管理模块通过命令请求的方式,向下方模块发起命令请求,在调度模块和管理模块得到安全性验证后,准许此命令的执行。其次,当 UEFI 固件安全模块或 Hypervisor 安全模块需要向管理模块反馈信息前,需要对调度模块和管理模块进行安全性验证。在验证安全性后,准许反馈行为的发生。

3 VirtinSpector 实现

在提出的基于 UEFI 固件的动态安全框架的基础上,实现了一个基于动态度量的原型系统,具体实现结构如图2所示。

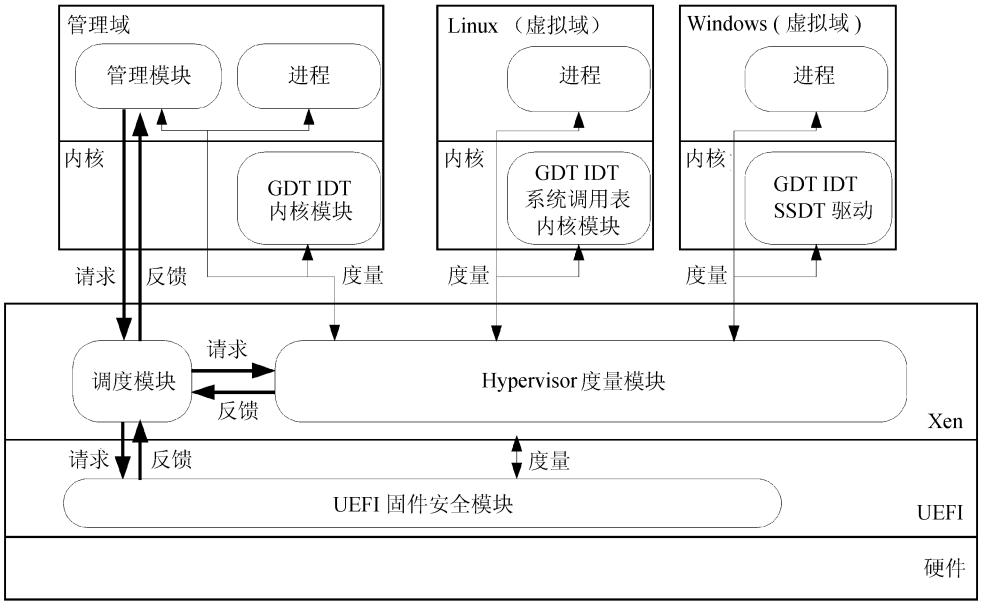

图 2　VirtinSpector 框架实现的原型系统结构

Fig. 2　Architecture of VirtinSpector based prototype system

该原型系统以国内某知名厂商某型服务器为实验开发平台搭建云环境,在云基础设施层中以 Xen 为开发对象,在 UEFI 固件安全模块和 Hypervisor 安全模块中实现动态度量功能。

度量行为的触发包括定时触发和手动触发。UEFI 固件度量模块在系统启动后立即进行定时度量,在管理域运行时,能够通过管理模块发送手动度量命令请求,手动触发 UEFI 固件的度量行为。UEFI 的定时触发,使用 EFI 的 SMM Periodic Timer Dispatch Protocol 注册时间片度量 handler,通过 Period 参数控制该 handler 被触发的时间间隔,当 EFI 启动后,每隔指定的时间间隔就自动运行 SMI handler。手动触发,使用 EFI 的 SMM Software Dispatch Protocol 注册手动度量 handler,注册时会指定相应的索引值,当向 0xB2 端口写该索引值,就触发手动度量 SMI handler 运行。Hypervisor 度量模块被在 UEFI 固件度量模块验证安全性后,立即进行时间片度量,管理域能够向其发送手动度量命令请求,手动触发度量行为。Hypervisor 度量模块的定时触发通过 Xen 的 timer 实现,为每一个虚拟域设置一个 timer,并绑定度量函数,定时通过 timer 触发度量函数。手动触发由 hypercall 实现,管理域通过 hypercall 向 Hypervisor 度量模块发送度量请求。

3.1 安全功能测试与分析

3.1.1　虚拟机威胁

1)传统攻击

本测试选取多种破坏虚拟机系统完整性的恶意程序样本,分别对 Windows、Linux 系统进行攻击。其中,针对 windows 的攻击样本有:Ntrootkit、Agony Ring0 Rootkit。其中,Ntrootkit、Agony Ring0 Rootkit 通过修改 SSDT 表项对系统内核进行挂钩。针对 linux 的攻击样本有:Enyelkm、Adore-ng。均通过添加恶意内核模块对系统进行攻击,结果如表 1 所示。

下面以攻击 Windows 的 Ntrootkit 样本为例进行分析。此样本通过修改 SSDT,对 SSDT 进行挂钩,修改了 NtCreateFile、NtOpenFile、NtCreateKey 等 10 余处函数入口地址,将其中的相应表项指向自身恶意程序进行攻击。Hypervisor 度量模块能够从虚拟域底层,实时地对 SSDT 的每一表项进行迭代哈希,当病毒修改 SSDT 后,Hypervisor 度量模块通过判断 SSDT 计算的哈希结果,发现 Ntrootkit 对 SSDT 的攻击。

上述实验表明,表 1 中列出的恶意程序病毒的破坏行为均被 VirtinSpector 检测到。结合 VirtinSpector 的实现框架分析,对于 windows,VirtinSpector 能够对系统进程、GDT、IDT、SSDT 和驱动进行完整性度量,发现针对破坏这些部分的恶意程序、木马、病毒。对于 Linux,VirtinSpector 能够对系统的进程、GDT、IDT、系统调用表、内核模块进行度量。发现恶意程序对系统调用表的 hook,添加恶意模块等恶意行为。

2)通过虚拟化软件攻击

本测试模拟利用硬件虚拟化对虚拟域进行攻击。在 Intel 的 VT 技术的硬件虚拟化中,CPU 运行

VM-Exit 程序从 non-root 状态切换到 root 状态,运行 VM-Entry 程序,从 root 状态切换回 non-root 状态。前文中提到的攻击,通过篡改 VM-Exit 或 VM-Entry 程序从虚拟机底层对虚拟域进行攻击。

本测试通过手动修改硬件虚拟化代码模拟此类攻击,结果如表2所示。

表1 传统攻击测试

Tab.1 Penetrate test in virtual machine through traditional attack

恶意程序样本	攻击对象	攻击前度量值	攻击后度量值	测试结果
Ntrootkit	Windows	d1dcf4ec00081e8fd6e79612b5f39e9575b0870e	b73cdefa4c6ebc11b18df6897aefe71f8f009970	SSDT 表项被篡改
Agony Ring0 Rookit	Windows	d1dcf4ec00081e8fd6e79612b5f39e9575b0870e	0c7104a41a38465359fd82688902d707e3b2204f	SSDT 表项被篡改
Enyelkm	Linux	fca1be40b3ca016bae6ae4243105fc1ebbda85f7	75a57223e550f6db029e37caa9a81044b17408ee	加载恶意内核模块
Adore-ng	Linux	fca1be40b3ca016bae6ae4243105fc1ebbda85f7	83f2d8f9fa44f3644d5cfe849456cafe227c7d4b	加载恶意内核模块

表2 利用虚拟化软件攻击虚拟域的测试

Tab.2 Penetrate test from hypervisor to virtual machine

攻击对象	攻击前度量值	攻击后度量值	测试结果
vmx_asm_vmexit_handler	8b9cb7672c23f4d31df34fb6155027b4a860fb27	e0634b09b4c20d29b649764630e93f78caa011cc	Xen 代码区完整性破坏
vmx_asm_do_vmentry	8b9cb7672c23f4d31df34fb6155027b4a860fb27	a5a8dd4b3b38e1746e7c8c29a663c628337f1e83	Xen 代码区完整性破坏

上述实验表明,对 VM-Exit 和 VM-Entry 程序的篡改,VirtinSpector 都能检测到。可信根 UEFI 固件度量模块对 Xen 进行度量,能够发现 Xen 的 HVM 代码区被篡改。因此,VirtinSpector 能够发现一些利用虚拟化软件对虚拟域进行的攻击。

3.1.2 虚拟化软件威胁

本测试模拟前文介绍的利用 DMA 对 Xen 进行的攻击。攻击针对 Xen 中关键的程序区和数据区,syscall_enter 为 Xen 处理系统调用的程序,通过篡改此代码可实现对 syscall 的挂钩。hypercall_table 为 Xen 的调用表,通过修改其中表项可实现对 hypercall 的挂钩。

本测试通过手动篡改 Xen 的程序或代码模拟此攻击,结果如表3所示。

表3 攻击虚拟化软件测试

Tab.3 Penetrate test of hypervisor

攻击对象	对象类型	攻击前度量值	攻击后度量值	测试结果
syscall_enter	程序	8b9cb7672c23f4d31df34fb6155027b4a860fb27	4154b283158dca727e941737679010fbbd7a719c	Xen 程序区破坏
int80_direct_trap	程序	8b9cb7672c23f4d31df34fb6155027b4a860fb27	c48c45d3e558c88474b60701947db85520536f3e	Xen 程序区破坏
hypercall_table	数据	8b9cb7672c23f4d31df34fb6155027b4a860fb27	167f6384feb326323977d6ea83a2417f6b82b375	Xen 数据区破坏
exception_table	数据	8b9cb7672c23f4d31df34fb6155027b4a860fb27	9d8fd0753cc8fcb1e7deb7a27c8d5ae87c44a0bc	Xen 数据区破坏

上述实验表明,对于测试中的攻击对象,其完整性的破坏都能被检测到。UEFI 固件度量模块对 Xen 的程序区和数据区进行动态度量,上述攻击对 Xen 的程序或数据进行了篡改,UEFI 固件度量模块通过判断 Xen 的哈希结果不匹配发现其攻击。因此,VirtinSpector 能够发现一些对拟化软件进行的攻击。

3.1.3　UEFI 威胁

如前文介绍的 UEFI 的威胁,目前已知的此类攻击多为在系统运行时对 bootloader、NVRAM 等部分进行篡改,之后在系统启动阶段运行不可信的 UEFI BIOS 对 UEFI 进行攻击。防御此攻击重心在系统启动阶段,能够通过系统启动时的安全策略很好的解决。VirtinSpector 通过与系统启动时的安全策略相融合,将重心放在系统运行阶段,对 bootloader、NVRAM 等与系统启动相关的对象进行实时的度量,与启动安全策略构成一个静态保护与动态保护相结合的保护手段。本测试通过模拟修改 bootloader、NVRAM 的攻击手段进行。通过手动的方式替换 bootloader,模拟对 UEFI 的攻击,结果如表 4 所示。结果表明,VirtinSpector 能够发现 bootloader 被替换。因此,VirtinSpector 能够通过对文件的动态度量,与启动安全策略相结合,对 UEFI 实现一种动静结合的保护手段。

表 4　bootloader 篡改测试

Tab. 4　Tamper test of bootloader

bootloader 度量时间	度量值	备注
2013 - 04 - 22T21:33:33	0187acb0839c18990e56943d677877e3bca0c2bd	攻击实施前
2013 - 04 - 22T21:37:40	b21319aa061ad1b3512f7619f8cad8bb866f1606	攻击实施后

3.2　性能测试

3.2.1　度量时间测试

本测试通过让 Hypervisor 度量模块同时度量多个虚拟机和 UEFI 固件度量模块度量 Xen,记录度量操作的耗时,判断度量操作是否显著影响用户使用云服务。虚拟域操作系统统一为 Windows 7。测试结果如图 3 所示,横坐标为虚拟机(VM)的数量,纵坐标为时间,虚拟机数量为 0 ~ 16 台,度量耗时随虚拟机数量接近正比例增加。计算得到近似公式为(细虚线):

$$time = 0.27 \times VM + 100.16。$$

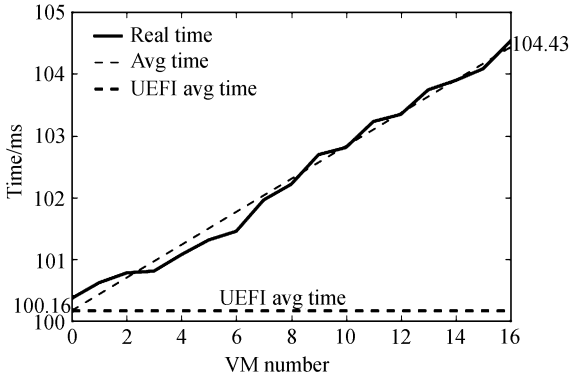

图 3　同时度量多个虚拟机耗时

Fig. 3　Measurement time of multiple VMs

斜率约为 0.27 ms/VM,说明对每个虚拟域的度量是相互独立的,表示度量系统中每增加 1 VM,会增加 0.27 ms 的度量开销。UEFI 固件度量模块度量 Xen 的时间不随虚拟机的数量而改变,估算线与纵坐标焦点的纵坐标 100.16 ms,为 UEFI 固件度量模块度量 Xen 的时间。上述分析可得,UEFI 固件度量模块度量 Xen 的时间比 Hypervisor 度量模块度量 VM 时间长很多,但数量级为毫秒级,不显著影响用户正常使用云服务。

3.2.2　CPU 占用率测试

本测试通过依次测试运行 1、8、16 台虚拟机(VM)在度量开启和关闭情况下的的 CPU 占用率,判断度量操作是否显著的占用 CPU 资源,对虚拟机框架造成影响。测试时间为 300 s,测试结果见图 4,其中,实线表示开启度量操作的 CPU 占用率的变化情况,虚线表示关闭度量操作的 CPU 占用率的变化情况。

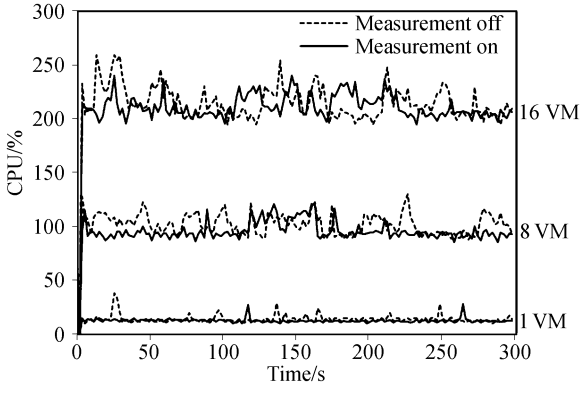

图 4　CPU 占用率测试结果

Fig. 4　CPU payload test

测试结果如图 4 所示,CPU 占用率随着 VM 数量的增加而增大,当 VM 数量较少时,CPU 占用率相

对稳定,VM 数量较大时,CPU 占用率波动较大。CPU 占用率在 1 VM、8 VM 和 16 VM 的情况下分别分布在 0 ~ 50%、75% ~ 125% 和 200% ~ 265% 区间内。值得注意的是,在相同 VM 数量下,度量操作的打开和关闭的 CPU 占用率的分布区间基本一致。说明度量操作对虚拟化框架基本无影响。

4 结束语

提出一种基于 UEFI 的虚拟机动态安全度量框架:VirtinSpector,该框架能够将 UEFI 固件作为可信硬件,对云系统的基础设施层进行实时的、动态的安全检测,提供传统可信技术未有的,通过硬件对云系统的基础设施层进行的动态保护。但某些方面仍然制约着 VirtinSpector 的安全性。首先,本框架从外部监控保护对象,监控模块解析保护对象能力有限,存在语义鸿沟问题,往往对监控对象的监控不够全面。其次,本框架由于效率的因素,并不能做到对任意时刻的监控,攻击者能够利用时间间隙进行攻击。再次,本框架采用度量判定手段,存在着局限性,只能判断度量对象可信和不可信 2 种情况,并非最优化设计方法。这些工作将在后续研究中继续开展。

参考文献:

[1] Seshadri A, Luk M, Shi E, et al. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy system[C]//Proceedings of the 20th ACM Symposium on Operating Systems Principles. New York: ACM, 2005:1 - 16.

[2] Heine D, Kouskoulas Y. N-force Daemon prototype technical description[R]. Technical Report VS-03-021, Laurel: The Johns Hopkins University Applied Physics Laboratory, 2003.

[3] Petroni N L, Fraser Jr T, Walters A, et al. An architecture for specification-based detection of semantic integrity violations in kernel dynamic data[C]//Proceedings of the 15th USENIX Security Symposium. Berkeley: USENIX, 2006:289 - 304.

[4] Dino A, Dai Zovi. Hardware virtualization rootkits[C]. Black Hat Conference 2006, Caesars Palace, CA, USA, 2006.

[5] Azab A M, Ning Peng, Sezer E C, et al. HIMA: A hypervisor-based integrity measurement agent[C]//Proceedings of the 25th Annual Computer Security Applications Conference(ACSAC'09). Los Alamitos: IEEE, 2009:461 - 470.

[6] Kil C, Sezer E C, Azab A M, et al. Remote attestation to dynamic system properties: Towards providing complete system integrity evidence[C]//Proceedings of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009). Los Alamitos: IEEE, 2009:115 - 124.

[7] Payne B D, Carbone M, Sharif M, et al. Lares: An architecture for secure active monitoring using virtualization [C]//Proceedings of the 29th IEEE Symposium on Security and Privacy. Los Alamitos: IEEE, 2008:233 - 247.

[8] Litty L, Lagar-Cavilla H A, Lie D. Hypervisor support for identifying covertly executing binaries[C]//Proceedings of the 17th USENIX Conference on Security Symposium. Berkeley: USENIX, 2008:243 - 258.

[9] Tygar J D, Yee B. Dyad: A system for using physical secure coprocessors[C]//Proceedings of the Joint Harvard-MIT Workshop on Technological Strategies for the Protection of Intellectual Property in the Network Multimedia Environment. Annapolis: Interactive Multimedia Association, 1991.

[10] Clark P C, Hoffman L J. BITS: A smartcard protected operating system[J]. Communications of the ACM, 1994, 37(11):66 - 70.

[11] Arbaugh W A, Farber D J, Smith J M. A secure and reliable bootstrap architecture[C]//Proceedings of the 1997 IEEE Symposium on Security and Privacy. Los Alamitos: IEEE, 1997:65 - 71.

[12] Azab A M, Ning Peng, Wang Zhi, et al. HyperSentry: Enabling stealthy in-context measurement of hypervisor intergrity[C]//Proceedings of the 17th ACM Conference on Computer and Communications Security(CCS 2010). New York: ACM, 2010:38 - 49.

[13] Sun Kun, Wang Jiang, Zhang Fengwei, et al. SecureSwitch: BIOS-assisted isolation and switch between trusted and untrusted commodity OSes[C]//Network and Distributed System Security Symposium(NDSS). Reston: Internet Society, 2012.

[14] Zimmer V, Rothman M, Hale R. Beyond BIOS: Implementing UEFI—The unified extensible firmware interface [M]. Hillsboro: Intel Press, 2006.

[15] Unified EFI, Inc. Unified extensible firmware interface specification[EB/OL]. [2013 - 3 - 17]. www. uefi. org/specs/download/UEFI_Spec_2_3_Errata_C. pdf.

[16] Wojtczuk R, Rutkowska J. Xen owning trilogy[C]. Black Hat Conference, USA, 2008.

[17] Heasman J. Hacking the extensible firmware interface [C]. Black Hat Conference, USA, 2010.

(编辑 杨 蓓)