

HyperSpector: 基于 UEFI 的 VMM 动态可信监控基的设计与实现

王锴^{1,2}, 李志华^{1,2}, 黄凡^{1,2}, 严飞^{1,2}

(1. 空天信息安全与可信计算教育部重点实验室, 湖北 武汉 430072;
2. 武汉大学计算机学院, 湖北 武汉 430072)

摘要: 随着计算机技术的高速发展, 云计算这一计算模式得到迅速推广。尽管可通过 VMM 来对云服务安全状态进行监控与安全增强, 但 VMM 自身的安全性往往是安全的前提。为了增强 VMM 的安全性, 提出一种基于 UEFI 的 VMM 动态可信监控基的设计与实现方法, 通过 UEFI 的固件运行特性, 在 VMM 下增加一层安全监控层, 实现在不增加新硬件成本的前提下对 VMM 的透明监测。实验表明, 所提出的监控方案是安全有效的。

关键词: 云计算; 可信计算; 动态可信; 统一的可扩展固件接口; 虚拟机监控器

中图分类号: TP309

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2016.00133

HyperSpector: VMM dynamic trusted monitor based on UEFI

WANG Kai^{1,2}, LI Zhi-hua^{1,2}, HUANG Fan^{1,2}, YAN Fei^{1,2}

(1. Key Laboratory of Aerospace Information Security and Trusted Computing of Ministry of Education, Wuhan 430072, China
2. School of Computer, Wuhan University, Wuhan 430072, China)

Abstract: With the rapid innovation of computer technology, cloud computing model expands. VMM was introduced to enhance the cloud safety and enhance the cloud safety. The safety of VMM itself, however, is the prerequisite of the safety of the cloud service. A method of VMM dynamic trusted monitor based on UEFI was proposed on the purpose of enhancing the safety of VMM. The monitor of VMM was realized by adding a monitor layer beneath it, which needed no more newly hardware cost. Experiment results show that the solution is efficient.

Key words: cloud computing, trusted computing, dynamic trust, UEFI, SMM

1 引言

近几年, 随着互联网的飞速发展、计算机硬件能力的不断升级, 以及当今企业对大数据和复杂计算的日益需求, 虚拟化和云计算技术应运而生, 为企业在现有 IT 设施的条件下提供了一条可

行之路。虚拟化技术既能充分利用已有 IT 设施, 把硬件资源整合成高扩展、高伸缩性的现代计算集群, 满足企业日益增加的数据存储和计算能力要求, 同时也简化了 IT 管理、减少能源消耗。当前, 虚拟化技术已成为云计算的基础, 并且越来越多的虚拟机监控器 (VMM, virtual machine

收稿日期: 2016-10-20; 修回日期: 2016-12-03. 通信作者: 王锴, blankaiwang@hotmail.com

基金项目: 国家自然科学基金资助项目 (No.61003268, No.61272452); 国家重点基础研究发展计划 (“973” 计划) 基金资助项目 (No.2014CB340601)

Foundation Items: The National Natural Science Foundation of China (No.61003268, No.61272452), The Major State Basic Research Development Program of China (973 Program) (No.2014CB340601)

monitor) 作为云计算系统的组件来增强云的安全性和伸缩性。

随着 VMM 的大范围运用, 越来越多的攻击和安全问题也随之而至。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 5 000 万用户的数据安全。根据 National Vulnerability Database 的报告显示, 著名的虚拟机监控器 Xen 和 VMware ESX 分别至少有 73 和 30 个漏洞, 而且还有更多的虚拟机 (VM, virtual machine) 逃逸攻击、VM 穿透攻击和 VM rootkits 正在研究和开发中。这极大地影响了服务提供商与云客户对云计算的应用信心。Gartner 的分析报告表明, 云安全是阻碍云计算发展的关键因素。为此, 如何加强对 VMM 安全性的监测已经成为保障云安全, 促进云计算发展必须解决的问题。

VMM 作为云计算系统中最重要基础, 其安全问题倍受关注, 研究人员已经开始重视并研究增强其安全性的方法。目前提出的方法一般包括: 减少可信基 (TCB, trusted computing based) 的代码, 从而减少攻击平面的方法^[1]; 在 VMM 之下、硬件层之上, 再增加一层软件形态的虚拟化监控层^[2]。然而, 这些方案都有其一定的局限性, 随着攻击者能力的不断提升, VMM 依旧面临严峻的安全挑战, 以软件层来构造可信基的方法, 由于软件自身具有防篡改能力, 以及运行内存空间隔离问题不足, 因此很难构造防护能力更为强大的可信基础。为此, 以软件、固件与硬件相结合的技术手段成为了一种值得探讨的方向。

为此, 本文在之前所提出的名为 VirtinSpector^[3] 框架的基础上, 进一步扩展了基于统一的可扩展固件接口 (UEFI, unified extensible firmware interface) 和系统管理模式 (SMM, system management model) 的 VMM 可信监控基的方案, 针对 UEFI 层次的度量问题, 从触发协议中参数的选择、VMM 内核获取问题、寄存器获取问题、核心模块监控实现问题和可信监控基保护问题开展了相关工作, 据此方案, 还实现了一个基于 UEFI 的 VMM 动态度量原型系统, 通过攻击实验、性能实验和数据分析验证了方案的安全性与实用性。

本文的贡献包括以下 4 点。1) 提出了一种基

于硬件和固件的 VMM 度量方案, 能够对上层 VMM 进行透明的度量检测并保护自身不受攻击影响, 而且能够与现有商业中使用的 UEFI 相结合, 不用增加额外的硬件和软件成本。2) 本文方案为动态框架, 能够使用硬件与固件特性, 实时地对 VMM 状态进行监控, 能够检测 VMM 的改变, 并能防止 TOC-TOU 攻击, 提供了云环境运行时的基础设施层保护。3) 本文方案实现了一个原型系统, 对保护对象是透明的, 提高了保护手段的安全性与有效性。4) 本文方案借用 UEFI 固件被篡改较软件难度大的特点, 将其作为可信基, 提高了云环境的安全性。

2 相关背景知识

在讨论本文所提出的方案之前, 首先对本文涉及的背景知识进行简要的介绍。

2.1 SMM 和 SMRAM 简介

x86 结构处理器定义了 4 种执行模式: 实模式、虚拟 8086 模式、保护模式和系统管理模式。实模式和虚拟 8086 模式用于兼容早期的处理器和软件, 当前的操作系统和软件主要运行于 32 bit 或 64 bit 保护模式, SMM 主要用于提供透明的机制来执行电源管理、系统安全检测等系统管理功能。SMM 通常由 BIOS (basic input-output system) 实现, 拥有自己的内存空间 (SMRAM, system management RAM)^[4], 通过系统管理中断 (SMI, system management interrupt) 触发。

SMRAM 在启动过程中由 BIOS 保留并通过内存控制器的 D_LOCK 位锁定, 防止被 OS 篡改。SMI 是最高特权的中断, 高于调度异常和外部中断, 当进入 SMM 后, 所有中断 (包括后续的 SMI 和 NMI) 都会被屏蔽或推迟^[3]。通过处理器的 SMI# 引脚或 APIC 中断来触发 SMI, 处理器会将当前执行现场自动保存到 SMRAM, 并进入 SMM 模式, 执行相应的 SMI handler, 最后通过 RSM 指令返回, 当处理器在 SMM 的时候, 处理器执行的代码和数据都在 SMRAM 空间里, 并且整个执行过程对操作系统来说是透明的, 即使 VMM 也无法干扰, 所以 SMM 可以作为可信执行环境。

2.2 BIOS 和 UEFI 简介

BIOS 是计算机系统中与硬件紧密相关而又

与操作系统相对独立的一个组件,其主要作用是在计算机加电时对 CPU、内存、芯片组、IO 设备等进行初始化。BIOS 最早出现在 IBM 兼容的 PC 中,时至今日已经不能适应现代计算机系统的发展和要求,其继任者,如 UEFI^[5]、Coreboot^[6]被提出,并克服了 BIOS 的不足。UEFI 是一种详细描述全新类型接口的标准,用于操作系统自动从预启动的操作环境加载到一种操作系统上。UEFI 是用标准化、模块化、C 语言动态链接的形式构建的系统,较 BIOS 而言具有更快速、易于实现和移植等特性^[7]。

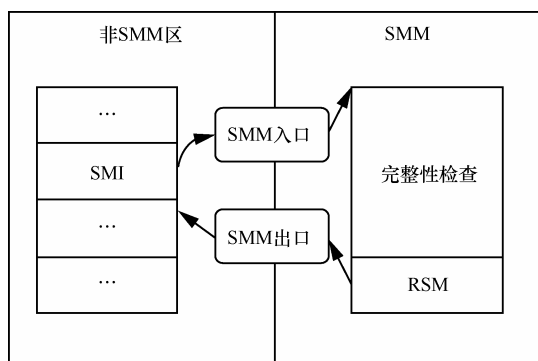


图 1 基于 SMM 的系统工作流程

作为计算系统的基础固件,UEFI 在实时运行的情况下,具有很高的安全性。首先,UEFI 程序的存储不能在运行时被篡改。其程序以固件的形式存在,通过专用工具刷入 UEFI ROM 中,且在启动过程中通过内存控制器进行硬件级的保护,相比一般系统软件被恶意程序篡改的可能性相对更小。其次,UEFI 程序运行时不能被篡改。在引导操作系统(OS, operations system)后,UEFI 调用的驱动和应用绝大部分将撤销,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,且大部分服务基于系统管理模式运行,而 SMM 中的内存 SMRAM 在系统启动时,已经被硬件锁定,在运行时难以被篡改,因此能保证运行态下的 UEFI 安全性,故 UEFI 可作为一种可信基的宿主环境。

3 威胁模型和假设

3.1 威胁模型

在虚拟化环境中,威胁是多方面的,本文的主要目标是构建虚拟化层软件 VMM 的可信监控基,

所以本文主要关注破坏完整性的攻击,而对于 DoS、ARP 等影响性能与欺骗性的攻击,不在本文的考虑范围之内;针对硬盘等存储设备上的数据修改,可以通过加密等方式进行保护,也不在本文的考虑范围之内;使用 TCG 提出的可信链启动方案已经可以很好地保护系统启动过程中的安全可信,本文作为可信启动的延伸,主要考虑 VMM 在运行阶段的威胁。因此,假设攻击者已知本文提出的 HyperSpecter 可信监护方案,攻击者可能通过使用 rootkit 来隐藏自身以试图绕过 HyperSpecter 的监测,如 IDT-rootkits 可以通过复制并修改 OS 的 IDT,再改变 IDTR 的值,使其指向新的修改后的 IDT^[8,9],从而达到隐藏自身的目的;对于 VMM^[10-13]这样复杂的软件,攻击者可以通过对其进行分析,发现软件自身的漏洞,从而获取虚拟机 domain 0 的控制权限,甚至控制虚拟机。例如,在 Xen 4.x 的最初版本中,攻击者可以通过 DMA 产生恶意 MSI^[13]来获取系统的最高权限。

最后,假设虚拟机系统装备了符合 TCG 可信启动要求的硬件(UEFI BIOS 中有 CRTM(Core Root of Trust for Measurement)和 TPM(Trusted Platform Module))和软件,并且进行了正确的配置,用户能够安全可信地启动虚拟机系统;同时,假设 UEFI 在启动 OS 后是锁定的(通常在商用 UEFI 中都会在启动完成后锁定),并且 UEFI 自身是可信的。

3.2 设计目标

基于上述的安全假设,提出了 HyperSpecter 方案,在运行时提供对 VMM 的可信度量和安全保护,该方案的成功运作需要满足以下要求:隐蔽触发、保护自身安全、获得度量对象和安全反馈。

隐蔽触发: 由于攻击者可能通过检测监测程序,通过使用 rootkit 实现逃逸或者隐藏,如 IDT-rootkits^[8,9],因此应满足对 VMM 隐蔽的情况下触发度量模式,整个触发的传递过程是安全隐蔽的,VMM 无法干预。

保护自身安全: 在 Xen 4.x 的早期版本中,攻击者已经可以通过 DMA 产生恶意的 MSI^[14]来获取 VMM 的最高权限,攻击者可以攻击度量程序本身,因此,度量程序应满足在 VMM 及 UEFI 运行过程和度量过程中所有自身模块是安全的。

获得度量对象：度量模块要能够获得度量对象（内存中的关键信息和 CPU 上下文）。

安全反馈：应满足发送反馈信息的模块是安全的，传递反馈信息的模块是安全的，接收反馈信息的模块是安全的，信息的反馈过程是安全可信的。

4 HyperSpecter 设计方法与实现

4.1 基本设计思路与原型系统结构

根据以上发现的安全问题和制定的安全要求，本文提出了一个名为 HyperSpecter 的基于 UEFI 和 SMM 的 VMM 可信监控基方案，该方案综合运用了 x86 结构 CPU 中的 SMM 模式、UEFI、VMM 在内存中的分布等技术，在不增加额外硬件和软件的情况下，实现了对 VMM 动态实时的可信监控。该方案主要由初始化模块、度量模块、验证模块、调度模块、报告模块组成，其架构如图 2 所示。

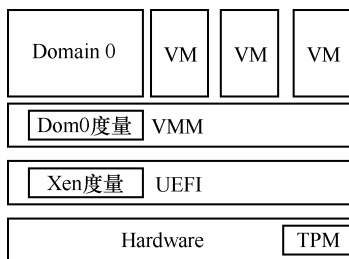


图 2 HyperSpecter 架构

1) 初始化模块

初始化模块主要用于为 HyperSpecter 的正常工作做一些准备和初始化配置，在虚拟机系统启动过程中运行，且只在启动过程中运行一次，然后就清除出系统。其工作主要分为 2 个部分：① 在 UEFI 的启动过程中初始化用于度量的 SMI handler^[15]，包括向 UEFI Manager 注册相应的定时和不时调度信息，设置定时调度的时间间隔等信息、注册触发不时 handler 的中断信息（相当于为 SMI 增加一种新的处理程序和触发端口号）；② 在 VMM 启动过程中获取 VMM 在内存中的分布信息并传递给 UEFI SMI handler。

2) 度量模块

度量模块是 HyperSpecter 的核心模块，其工作虽然比较单一，但却是 HyperSpecter 方案成败的关键所在。该模块分为 2 个部分：主体模块在

UEFI 的 SMI handler 中实现，主要功能是对 VMM 的内核模块进行度量，以及获取 CPU 执行上下文并对其关键寄存器进行度量；另一部分是在 VMM 中实现，其作用是对特权域 Domain 0 的核心模块进行度量。最后生成度量日志，并将度量值扩展写入 TPM 芯片中的 PCR（platform configuration register）中供上层应用挑战验证。

3) 验证模块

验证模块是唯一一个与 VMM 分离的模块，主要用于对报告模块提交的度量日志信息和 PCR 信息进行验证。假设虚拟机系统在配置结束第一次运行时是可信的，然后将第一次生成的度量信息作为基线值，之后收集到的信息都与基线值进行匹配，若匹配成功则表明系统安全可信，否则说明已经被攻击。

4) 调度模块

调度模块是用来定时或不定时地触发 HyperSpecter 运行的模块，用于定时触发度量模块执行，并且为用户主动调用提供接口，当收到用户发送的命令时直接触发度量模块执行，且当用户长时间不主动触发度量模块时，自身也会通过一个随机时间来进行触发，这样做主要是为了防止针对 VMM 的 TOC-TOU 攻击。

5) 报告模块

报告模块用于度量模块执行结束后，将生成的度量日志和存储在 PCR 中的度量信息安全地收集并发送给验证模块。

4.2 安全触发问题

一个好的监控基必须既能保护自身的安全，又能正确地进行度量和监控。所以其触发调用过程必须隐蔽进行，以防止恶意软件或已经被攻击的 VMM 清除自身的攻击痕迹，从而逃避监控基的检测。虽然每个 CPU 只有一个 SMI 引脚，但却有很多硬件事件可以触发 SMI，由 I/O Controller Hub (ICH) 定义，如 ICH4 有 39 种事件可以触发 SMI，包括电源管理事件、USB 事件、Total Cost of Ownership (TCO) 事件、写 0xB2 端口等，ICH5、ICH9、ICH10 等新的 ICH 芯片组也有类似的事件可以触发 SMI。根据 SMI 触发事件、UEFI 的特性、隐蔽触发以及保护自身安全的需求，HyperSpecter 使用了 2 种方式（定时触发和

手动触发)来触发 SMI。

定时触发使用了 UEFI 中对 SMI handler 基于时钟调度的支持,利用 EFI_SMM_PERIODIC_TIMER_DISPATCH2_PROTOCOL 协议注册定时度量 SMI handler。特别需要指出的是,根据特定 CPU 和芯片组的不同,SMI 触发时间间隔和特定 SMI handler 的调度时间是不同的,如 Intel Xeon 平台下,SMI 触发时间间隔必须是 1.5 ms、16 ms、32 ms、8 s、16 s、32 s,而 SMI handler 的调度时间必须是选用的 SMI 触发时间间隔的倍数,恰当选取 SMI handler 的调度时间和 SMI 触发时间间隔以减少对系统性能的影响。定时触发是基于硬件时钟并且在 UEFI 中直接进行调度,因此定时触发是对上层 VMM 透明的,即使攻击通过 VMM 的漏洞获取其控制权限,攻击者也无法发现定时触发过程进而对其进行干扰。

手动触发使用了 UEFI 中对 SMI handler 的基于 0xB2 端口的触发支持,利用 EFI_SMM_SW_DISPATCH2_PROTOCOL 协议注册 SMI handler 的手动触发。手动触发的实现主要是方便用户或管理验证端随时调度度量 handler 进行度量,而且用户和管理验证端的验证调度是随机发生的,可以进一步防止 TOC-TOU 攻击(根据 SMI 定时执行的时间而躲避其检测)。

4.3 VMM 内核获取问题

度量模块必须考虑到 VMM 运行时使用的线性地址同 SMRAM 运行的程序使用的物理地址之间的语义差别;同时,由于内核在内存中的分布及其内容因 VMM 而异,为了增强度量模块对不同 VMM 内核度量的适应性,本文的度量模块主要依赖于 VMM 内核的线性映射、稳定性和持久性 3 个特征^[16]。

线性映射:在 x86 结构中,VMM 或 OS 在内存中通常是线性映射的,Xen 内核的虚拟内存地址与物理地址是线性映射关系,且 Domain 0 也是线性映射的,上述 4 部分内容都在内核空间中且连续存放,有利于直接对其进行地址转换和度量(SMM 中直接使用物理地址进行寻址)。

稳定性:稳定性意味着 VMM 在内存中的内容是静态的、不变的。如果度量内存中的内容是可变的,那么必将有一个时间能度量到,度量结

果将难以判断,所以 HyperSpecter 不度量内核的动态可变部分。

持久性:持久性是指 VMM 内核使用的内存是一直占用的,不会被交换到硬盘中,在系统运行过程中一直存在于内存中。

本文 HyperSpecter 的实现依赖于这 3 个特征,否则将失效。通过跟踪 Xen 初始化代码的执行过程发现 Xen 内核代码在初始化过程中存在位移现象(从开始的一个地方整体转移到了另一个地方),但存在一个 xen_phys_start 变量最终保存 Xen 内核的线性地址,同时存在 __pa() __va 宏可以实现线性地址与物理地址之间的转换,这样就可以将 Xen 内核的 VMA 地址转换为物理地址。在研究 Xen 内核在内存中的分布时发现一个比较有趣的现象,通过 VMA 转换出的物理内存地址的起始地址与 Xen 真正开始存放其代码的地址之间有 1 MB 的空间进行了隔离,称之为“隔离内存”,这一部分内存为后面的操作提供了很多方便。

出于效率的考虑,使用了 CMOS 中的寄存器进行传值,但 CMOS 寄存器可以由任何具有特权的软件访问,为了保证地址的正确传递,在传值时计算地址的散列值,并将散列值同样写入前面提到的“隔离内存”中。UEFI SMI handler 读取 CMOS 中的内存物理地址进行散列,并与“隔离内存”中的散列值进行对比,当两者匹配时才进行下一步操作。由于 Xen 初始化过程中生成的一些变量或数据在初始化结束后释放,所以在 Xen 初始化过程中计算的地址散列值不会被保存,且不可重现,这就保证了在 Xen 运行过程中散列值在内存中的安全性,恶意软件或已经被攻击的 Xen 也无法发现,通过上述步骤就可以安全地将 Xen 内核的物理地址传给 SMI handler。

4.4 VMM 寄存器获取问题

根据 Intel 开发者手册^[3]中的说明,IA-32 CPU 进入 SMM 时,会保存自己的状态到 SMRAM 的相应区域([SMBASE + 8000H + 7FFFH]~[SMBASE + 8000H + 7E00H])。存在这些地方的寄存器有些是只读的,并不能修改(修改这些寄存器将导致不可预知的行为,如 CR4),SMI 处理程序不应依赖任何存储在这个区域的数据的值。图 3 是 SMRAM 的大致描述(不同 CPU 可能不一样)。

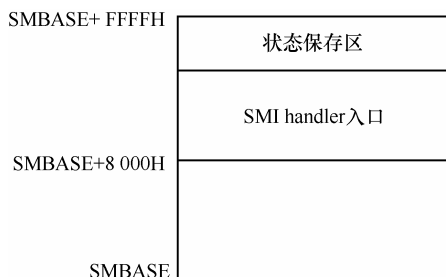


图3 SMRAM 内存分布

默认 SMRAM 大小为 64 kB, 并且它的开始会有一个物理基地址 (又叫作 SMBASE)。当然这一切都是在物理内存上的。SMBASE 的默认值是 30000H (硬件复位就是这个值。)。处理器会到 [SMBASE+8000H] 处去找 SMI 处理程序的第一条指令。[SMBASE + FE00H]~[SMBASE + FFFFH] 存储处理器的状态, 且根据各寄存器的分布可以进行访问。

主要关注 2 个寄存器: IDTR 和 CR3, IDTR 在系统初始化后应该是永久不变的, 对于 CR3, SMM 需要其进行内存地址转换 (只要是进行线性映射, CR3 就应该一样), 所以对 SMM 来说也是不变的。

4.5 VMM 核心模块监控实现问题

使用 Xen 作为 VMM 平台, 而 Xen 属于虚拟架构 3 种模式 (独立监视器模式、主机模式和混合模式) 中的混合模式, Xen 直接运行在裸机上, 具有最高运行权限, 其上运行虚拟机 (又称为域), Xen 和特权域 Domain 0 一起组成了 VMM。采用这种模式可以充分利用原操作系统中的设备驱动等功能, Xen 就可以只提供如 CPU 和内存虚拟化等功能, 但利用原操作系统, 如 Linux 的设备驱动等功能相当于复用了 Linux 内核相关模块。

可加载内核模块 (LKM, loadable kernel module) 是 Linux 内核重要特征之一, Linux 内核本身为单内核结构, 通过 LKM 可以增强其扩展性, 也方便加入内核中新的功能或驱动而不必重新编译内核, 但内核模块与内核其他部分具有相同的权限和系统资源, 在商业环境中, 内核 rootkit 通常就是通过内核来加载的, 由于在 Xen 4.x 的早期版本中, 攻击者可以通过一系列的操作获取 domain 0 的权限, 或通过 rootkit 来隐藏自身进程, 因此为了保证在 Xen 平台上运行的 VMM 可信

性, 首先要保证 VMM 主内核模块的安全可信。

本文采用了 2 种方法来保护主内核模块的安全可信: 首先通过实时检测特权域 Domain 0 中内核模块的加载, 在加载模块之前先对加载的模块进行度量, 确保要加载的模块是可信的; 然后在运行过程中通过定时或随机地对内核中运行的模块进行度量来确保其运行过程也是安全可信的。通过这 2 种方法可以确保恶意的 LKM 即使隐藏进程、端口、文件也无法逃脱 HyperSpecter 的检测。

4.6 可信监控基保护问题

HyperSpecter 以构建 VMM 的动态可信监控基为目标, 对处于运行态的 VMM 进行全方位的安全监控, 确保其安全可信, 且 HyperSpecter 作为运行在比 VMM 更底层的可信监控层, 其自身的安全可信尤为重要。HyperSpecter 主要由基于 UEFI 的度量 SMI handler、处于 Xen 中的内核模块检测代理组成以及触发机制。

根据 Intel 开发者手册^[4], 在 Intel 芯片组中定义了一组 SMRAMC (system management RAM control register) 来控制其他处理模式对 SMM 空间的可视性和可访问性, 其中有 D_OPEN 和 D_LCK 这 2 个关键的寄存器位。D_OPEN 位控制其他处理模式对 SMRAM 的可访问性, 如果 D_OPEN 清空, 其他处理模式将无法看到 SMRAM, 只有处于 SMM 模式中的代码才能在 SMM 中执行, 非 SMM 处理模式对 SMRAM 的访问请求都会被转换到对 VGA 缓存的访问。而 D_LCK 位控制 SMRAMC 的可访问性, 若 D_LCK 置位, SMRAMC 将变为只读, 无法更改^[17]。在加入 SMI handler 后, 同时要设置 UEFI 在初始化后要置 D_LCK 位并清空 D_OPEN 位, 以保护 UEFI 和 SMRAM 不受特权软件访问或篡改, 从而可以确保 UEFI 和 SMI handler 的安全可信。

VMM 内核模块检测代理是集成在 Xen 的内核模块中, 而在运行时 Xen 内核是受 SMI handler 度量监控的, 所以信任可以由 SMI handler 传递给 VMM 内核模块检测代理, 由检测代理来对 VMM 内核模块进行度量监控。

触发机制主要由定时和随机 2 种方式组成。SMI handler 的定时触发由 UEFI 中的定时调度机制实现, 而 VMM 内核模块检测代理由 Xen 中的

时钟来定时触发, 虽然两者可能不同步, 但并不影响检测效果。随机方式是 HyperSpector 为了便于用户主动请求度量 and 检测 SMM 逃逸攻击而实现的, 可以由用户通过命令接口调度运行度量执行, 也可以由调度模块通过随机时钟进行调度执行。

5 测试评估

在 UEFI 中注入了大约 400 行代码, 其中, 用于定时与非定时触发的 SMI handler 各 200 行; 在 Xen 中的内核模块检测代理, 利用 Xen 中已有的函数和脚本大大提高了代码简洁性, 仅 100 余行代码即可完成目标要求。

5.1 VMM 内存测试

VMM 内存检测是为了确认 VMM 在内存中的分布, 精确地定位 Xen 内核在内存中的地址并测试其是否是静态、稳定的。UEFI 中的 SMI handler 利用 Xen 在初始化时传递的地址寻址到 Xen 内核所在内存的地址, 同时利用 objdump 工具读取所编译的 Xen 内核进行匹配。如图 4 和图 5 所示, 可以看出两者是一致的, 说明正确找到了 Xen 的代码段地址。

```
-----, text start:BE500000
149066E9
1F0F00
1BADB002
3
E4524FFB
EB15010F
B900143F
0
C18ED98E
E98EE18E
```

图 4 SMI handler 输出

```
File Edit View Terminal Help
xen-syms: file format elf64-x86-64

Contents of section .text:
ffff82c480100000 e9669014 000f1f00 02b0ad1b 03000000
ffff82c480100010 fb4f52e4 0f0115eb 3f1400b9 00000000
ffff82c480100020 8ed98ec1 8ee18ee9 8ed1488b 0dcfa113
ffff82c480100030 000f22e1 488b25e5 3f140048 81cc187f
ffff82c480100040 00006a00 9d6808e0 0000488d 05030000
ffff82c480100050 005048cb 0f011dbb 3f140085 db0f853d
lines 1-10
```

图 5 objdump 读取的 Xen 内核

另外, 也进行了代码和数据的边际和稳定性测试, 找到了 Xen 在内存的布局, 有趣的是, 还发现了 Xen 内核在内存中分布时把最开始的 1 MB 全部初始化数字 5, 同时利用这 1 MB 内存中的部分字节进行 SMI handler 与 Xen 的数据交换。

5.2 VMM 攻击测试

如前文所述, 目前有许多针对 Xen 的攻击方法, 通过修改并重新编译 Xen 内核的方法来模拟针对 Xen 内核的攻击测试, 同时也通过执行 `rmmod vfat` 指令对内核模块进行删除模拟内核变动测试, 在重新编译 Xen 内核后, 日志记录的 SHA-1 值发生变化, PCR 记录的结果也会发生改变; 而当内核模块被删除后, 系统的动态度量值也会发生改变。测试表明, 针对 Xen 内核代码及数据的篡改以及内核模块的变动是可以检测到的。SMI handler 能够对 Xen 在内存中的代码区 and 数据区进行动态度量监控, 同时内核模块监控代理也能够成功地对内核中模块的变动进行精确的动态度量监控。因此, HyperSpector 能够对 VMM 的代码和关键模块进行动态监控。

5.3 性能测试

性能测试主要是针对 VMM 度量模块 (SMI handler) 和内核模块检测代理的执行时间进行测度, 通过多次执行来判断度量操作是否会显著影响 VMM 的服务和用户体验。性能测试本应该包括 CPU 占用率测试, 但根据 SMI 的特点, 当 SMI handler 执行时会暂停所有 CPU 的活动转而选择其中一个 CPU 来单线程执行 SMI handler, 其 CPU 占用率测试没有意义。

在实际测试时, 分 2 种情况进行度量时间消耗测试 (写 PCR 和不写 PCR), 并重复进行 10 组实验, 测试结果如图 6 所示。

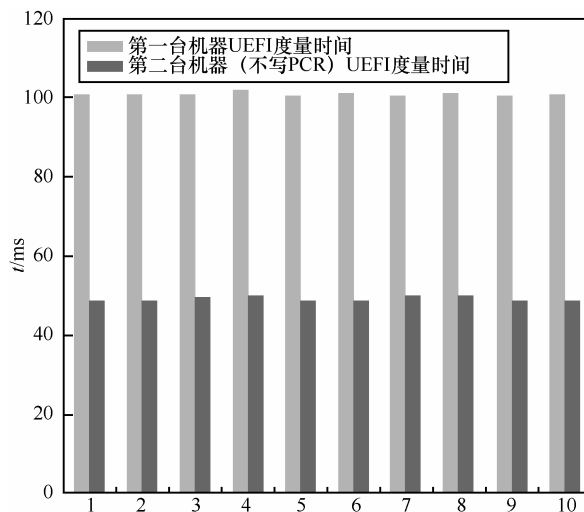


图 6 UEFI 动态度量时间消耗

上述测试结果表明,写 PCR 时会占用大约 50 ms,这可能是由于 TPM 芯片接在南桥的慢速端口上,所以读写速度比较慢;而内核模块检测代理的执行时间几乎可以忽略不计。而且整个 host 度量时间不受 VM 运行数据影响,不运行 VM 和运行 24 台 VM 的度量执行时间是一样的。总体来说,100 ms 级别的时间消耗对整个 VMM 影响很轻微,用户不会感觉到任何卡顿。

5.4 测试总结

本次测试主要侧重于 Host 动态监控模块中的功能确认测试和性能测试,从上面的测试结果可以看出:只需要加入约 500 行代码就可以实现整个 HyperSpecter,HyperSpecter 能够找到 Xen 的代码段,能够实现 CMOS、UEFI、Xen 的交互通信,能够识别模块和代码的变动,整个度量过程功能完整实现,且在写 PCR 的情况下也能够 在 100 ms 级的时间内完成 UEFI 度量功能,性能影响较小,且能支持上层 Xen 同时运行 24 台 VM。所以,HyperSpecter 实现了一个与现有商业系统兼容的具有实际可行性的动态可信监控基。

6 相关工作

通过硬件辅助技术来保护软件的完整性不是什么新技术,研究者们已经尝试了许多不同的方案。在先前的尝试中,Flicker 使用 TPM 构建一个简化的 TCB 来检测篡改内核事件^[18],它使用了受广泛支持的 DRTM 和延迟启动技术。TrustVisor^[19]设计了一个小型的带有加密功能性的 VMM,但要求所有原有应用程序必须重新编译,同时也要求 DRTM 的支持。SecureSwitch^[15]是一个基于 BIOS 辅助的用于安全初始化和可信执行环境的系统,但它的设计是专用的,不能运用到现有的云计算环境中。这些尝试虽然有一定进步,但却受限于不兼容和性能太差等原因无法运用到现有成熟的商业云环境中。

最近的研究者们开始聚焦于利用其他(如 PCI、SMM 等)硬件辅助,它们具有与现有商业系统兼容的特性,同时也具有更高的安全性和灵活性。Copilot^[20]使用一个特殊的 PCI 设备循环扫描目标机中的内存来进行分析,以确保运行系统的安全性。虽然其与现有云计算系统是兼容的且

能隐藏自身,但却无法跨越运行的系统和 PCI 设备之间的语义鸿沟,而且也无法获得 CPU 中寄存器的状态。Zhang 等^[21]提出了一种借助 SMM 对恶意软件实施监控的框架 MALT,实现在不应用虚拟化或仿真技术的前提下对 rootkit 的监测,从而避免 rootkit 针对监测进程的攻击。但是,其多核处理器核间通信信号没有得到充分处理。HyperSentry^[22]利用了基于 SMM 和 IPMI (intelligent platform management interface) 技术,构建了一个侧通道来完成触发和传值,需要 3 次进出 SMM,效率损耗较多,而且其实际度量模块是基于 Xen 运行的,然后度量 Xen,能否不被 Xen 欺骗和其自身安全性都有待商榷。

本文的 HyperSpecter 则充分利用了 SMM 与 UEFI 的特性,其不是要减少 TCB 的大小,而是扩展 TCG 可信链和 TCB 的概念,要构建一种新的动态的可信监控基,本文虽然也利用了 SMM 和 TPM 等硬件技术,但本文提出来的 HyperSpecter 不需要任务额外硬件设备的支持,也不需要反复进出 SMM,就能完成对 VMM 内核、CPU 状态、特权域中内核模块等全方位的检测和监控,并且提供了定时和随机 2 种调用方式,有效地解决了 SMM rootkits 一般性的 TOC-TOU 攻击。但是,由于多核处理体系结构中需要进行处理器核间信号处理,本文方案在处理核间内部信号时还有不足,有待进一步完善。

7 结束语

本文提出了一种基于 UEFI 和 SMM 的 VMM 动态可信监控基——HyperSpecter。该动态可信监控基以 SMM 的隐蔽性和 UEFI 固件锁定特性为基础,利用 VMM 内核的线性映射、稳定性、持久性,以及 SMI 自动保存 CPU 状态的特性,构建了针对 VMM 内核、CPU 状态和特权域关键内核模块的动态可信监控基,为 VMM 提供传统可信技术和安全措施从未有的基于硬件的动态安全保护。而且实现了一个 HyperSpecter 原型,并通过测试证明其能够有效地对 VMM 进行监控,能够发现任何 VMM 内核、状态和特权域的关键模块的变动。经测试,本文的 HyperSpecter 不需要额外的硬件或软件层,只需要 500 行代码,且对现

有 VMM 的运行影响甚微。

当然, 本文的 HyperSpector 也存在一些不足的地方, 如通过 CMOS 传值不是一种 OS 与 UEFI 之间标准的传值方式, 也不够安全, 后期将尝试使用基于 NVRAM 的传值方式进行安全数据交换; 通过软中断进行随机触发的方式也可能被 VMM 延迟或欺骗。另外, 后期需要寻找一种更安全的触发方式; 基于度量的判定方式有一定的局限性, 不能判断 VMM 可信或不可信, 并非最优化的设计方法。这些问题和不足将在后期的工作中继续研究和改进。

致谢: 本文所采用的实验方法, 保护手段来源于李志华硕士论文《云动态可信监控基的设计研究》, 在此感谢李志华在实验和讨论中对本文工作给予的帮助。

参考文献:

- [1] 周振柳, 栾好利, 张楠, 等. 基于动态安全属性保护的可信固件[J]. 计算机工程, 2011, 37(4): 137-139.
ZHOU Z L, LUAN H L, ZHANG N, et al. Trusted firmware based on dynamic security attribute protection[J]. Computer Engineering, 2011, 37(4): 137-139.
- [2] 李博, 李建欣, 胡春明, 等. 基于 VMM 层系统调用分析的软件完整性验证[J]. 计算机研究与发展, 2011, 48(8): 1438-1446.
LI B, LI J X, HU C M, et al. Software integrity verification based on VMM-Level system call analysis technique[J]. Journal of Computer Research and Development, 2011, 48(8): 1438-1446.
- [3] 严飞, 石翔, 李志华, 等. VirtinSpector: 一种基于 UEFI 的虚拟机动态安全度量框架设计与实现[J]. 四川大学学报, 2014, 46(1): 22-28.
YAN F, SHI X, LI Z H, et al. VirtinSpector: a UEFI based dynamic secure measurement framework for virtual machine[J]. Journal of Sichuan University, 2014, 46(1): 22-28.
- [4] Intel. Intel® 64 and IA-32 architectures software developer manual[M]. Intel Press, 2014.
- [5] Unified extensible firmware interface specification[EB/OL]. <http://www.uefi.org>.
- [6] Coreboot[EB/OL]. <http://www.coreboot.org>.
- [7] VINCENT Z, MICHAEL R, SURESH M. Beyond BIOS developing with the uenified extensible firmware interface[M]. Intel Press, 2010.
- [8] Handling interrupt descriptor table for fun and profit[EB/OL]. <http://www.phrack.org/issues/59/4.html>.
- [9] Hijacking the linux kernel[EB/OL]. <http://www.drops.dagstuhl.de/opus/volltexte/2011/3063/pdf/7.pdf>.
- [10] FU Y, LIN Z. Space traveling across VM: automatically bridging the semantic gap in virtual machine introspection via online kernel data redirection[C]//IEEE Symposium on Security and Privacy. 2012.
- [11] BRENDAN D G, TIM L, MICHAEL Z, et al. Virtuoso: narrowing the semantic gap in virtual machine introspection[C]//IEEE Symposium on Security and Privacy. 2011.
- [12] LIONEL L, HORACIO A L C, DAVID L. Hypervisor support for identifying covertly executing binaries[C]//Usenix Security Symposium. 2008.
- [13] BRYAN D P, MARTIN D P D A C, WENKE L. Secure and flexible monitoring of virtual machines[C]//The Annual Computer Security Applications Conference. 2007.
- [14] Following the white rabbit: Software Attacks against Intel VT-d[EB/OL]. <http://www.invisiblethingslab.com/itl/Resources.html>.
- [15] KUN S, WANG J, ZHANG F W, et al. SecureSwitch: BIOS-assisted isolation and switch between trusted and untrusted commodity OSes[C]//Network & Distributed System Security Symposium. 2012.
- [16] KOSTYA K. Cloudburst: a vMware guest to host escape[C]//Blackhat Talk. 2009.
- [17] SHAWN E, SHERRI S, CLIFF Z. SMM rootkits: a new breed of OS independent malware[C]//Security and privacy in communication networks. 2008.
- [18] JONATHAN M M, BRYAN J P, ADRIAN P, et al. Flicker: an execution infrastructure for TCB minimization[C]//ACM European Conference on Computer Systems. 2008.
- [19] JONATHAN M M, LI Y L, QU N, et al. TrustVisor: efficient TCB reduction and attestation[J]. IEEE Symposium on Security & Privacy, 2010, 41(3): 143-158.
- [20] NICK L P J, TIMOTHY F, JESUS M, et al. Copilot — a coprocessor-based kernel runtime integrity monitor[J]. Proceedings of Usenix Security Symposium, 2010, (13): 179-194.
- [21] ZHANG F W, KEVIN L, ANGELOS S, et al. Using hardware features for increased debugging transparency[C]//IEEE Symposium on Security and Privacy. 2015.
- [22] AHMED M A, NING P, WANG Z, et al. HyperSentry: enabling stealthy incontext measurement of hypervisor integrity[J]. ACM Conference on Computer and Communications Security, 2010: 38-49.

作者简介:



王锴(1993-), 男, 河北石家庄人, 武汉大学硕士生, 主要研究方向为系统安全、可信计算。



李志华(1988-), 男, 山西晋城人, 硕士, 主要研究方向为系统安全、可信计算。

黄凡(1992-), 女, 湖南常德人, 武汉大学硕士生, 主要研究方向为系统安全、可信计算。

严飞(1980-), 男, 湖北武汉人, 博士, 武汉大学副教授, 主要研究方向为系统安全、可信计算。