基于虚拟机的内核完整性保护技术

张 磊[1]，陈兴蜀[1]，刘 亮[2]，李 辉[1]

(1. 四川大学计算机学院 成都 610065；2. 四川大学电子信息学院 成都 610065)

【摘要】针对云计算中客户虚拟机内核完整性面临的威胁，该文提出了一种保护虚拟机内核完整性的技术—CTVM。该技术在KVM虚拟机环境中实现了虚拟化可信执行环境的创建，使多个客户虚拟机同时拥有可信计算功能，能对客户虚拟机提供启动完整性度量；在此基础上利用硬件辅助虚拟化技术，通过为客户虚拟机构造隔离的地址空间，使客户虚拟机中不可信模块与内核运行在逻辑隔离的地址空间。从这两个方面实现对客户虚拟机的启动和运行时的完整性保护。最后，以某国产服务器为实验平台实现了CTVM原型系统，系统测试与分析验证了技术的可用性，系统性能损耗在可接受的范围内。

关 键 词 完整性； 内核； KVM； 可信计算； 虚拟机

中图分类号 TP309 文献标志码 A doi:10.3969/j.issn.1001-0548.2015.01.020

A Kernel Integrity Protection Technology Based on Virtual Machine

ZHANG Lei[1], CHEN Xing-shu[1], LIU Liang[2], and LI Hui[1]

(1. School of Computer Science, Sichuan University Chengdu 610065;

2. School of Electronics and Information Engineering, Sichuan University Chengdu 610065)

Abstract For the kernel integrity threats of virtual machine in cloud computing environment, an integrity protecting technology of virtual machine kernel, cloud trusted virtual machine(CTVM), is proposed. In the CTVM, the virtual trusted execution environment in kernel-based virtual machine(KVM) is created, the multiple virtual machines are endowed with a trusted computing function at the same time, and the guest virtual machines are provided with integrity measurement ability. By utilizing hardware virtualization technology, the untrusted kernel modules are isolated from operating system kernel through constructing isolated address space in guest virtual machines, so as to protect the booting integrity and runtime integrity of guest virtual machines. Finally, with a domestic server as the experimental platform, CTVM prototype system is presented. System test and analysis show that the system performance loss is within the acceptable range.

Key words integrity; kernel; KVM; trusted computing; virtual machine

用户使用云计算平台时，主要担心自己的数据和计算资源受到安全威胁：1) 在多租户共享计算资源的模式下，用户资源可能受到来自其他恶意用户的威胁；2) 用户可能遭到不可信云平台管理者发起的内部攻击。为实现云计算平台的资源虚拟化和管理，平台提供商广泛使用VMware、KVM和XEN等虚拟机监视器(virtual machine monitor, VMM)软件。目前这些软件都使用虚拟机对用户提供隔离机制，可以限制虚拟机之间的访问，但为了便于前后端驱动传递数据，VMM同时提供了虚拟机之间的内存共享机制，这可能被恶意虚拟机利用。另外，VMM和管理工具通常处于特权域的控制中，不可信的云平台提供者或管理员可借助特权域的特权获取用户的隐私或破坏客户虚拟机(guest virtual machine, GVM)的完整性[1-2]。

目前，已经有很多研究关注GVM的安全问题。Overshadow[3]基于虚拟化技术，采用对所有内存页加密的方法来保护用户的隐私性。而CHAOS[4]采用页表隐藏的方式，对I/O页面进行加密，性能更好。Inktag[5]在不可信的操作系统中基于虚拟化技术保护关键应用。CloudVisor[6]在假设特权域不可信的情况下，使用嵌套虚拟化技术[7]，将安全防护和资源管理进行分离，对VMM与VMs之间的交互实施仲裁，从而实现对GVM的隐私性保护和完整性验证。文献[8]基于Xen设计和实现了一种可信的VMM，用户可按照完整性要求保存敏感数据，只允许信任程序访问。并在Xen中设计实现了虚拟可信计算模块(virtual trusted platform module, VTPM)，可为每个虚

收稿日期：2014－02－10；修回日期：2014－11－25

基金项目：国家自然科学基金(61272447)；国家科技支撑计划(2012BAH18B05)

作者简介：张磊(1983－)，男，博士生，主要从事信息安全、云计算和可信计算方面的研究.

拟机创建一个VTPM实例, 具有和硬件TPM[8]一致的功能。文献[9]提出了一种基于UEFI的虚拟机动态安全度量框架，对系统的基础设施进行实时、动态的安全度量, 提供传统可信技术无法达到的动态保护。目前部分研究工作基于可信计算技术实现了对宿主机和虚拟机的静态和动态度量，能提供远程可信验证机制，但在虚拟机中利用可信计算保护内核完整性的技术还相对较少，并且完整性度量可发现破坏完整性的行为，却不能对攻击行为采取隔离措施。

针对上述问题，本文提出了一种CTVM的KVM虚拟机内核完整性保护技术，在启动和运行时对云计算平台中GVM的完整性提供保护。

1 概 述

1.1 威胁模型和假设

在多租户的云计算环境中，平台的可信性和恶意攻击是用户主要关注的安全问题[9]。传统的可信计算技术可保证宿主机操作系统的可信启动，并能提供完整性度量值和远程验证功能，但物理主机可信不代表GVM的可信, 尤其是运行过程中GVM的完整性可能遭到破坏。由于虚拟机通常采用商用操作系统，内核完整性容易遭到破坏，攻击者可通过获取超级管理员权限对内核进行Rootkit注入，也可利用操作系统中设备驱动的漏洞，还可加载没有通过验证的内核模块。恶意代码有多种方法破坏内核的控制流和数据完整性，例如：直接修改内核代码、修改控制数据(系统调用表、中断描述符表和函数指针)、修改非控制数据(进程描述符和文件系统元数据)、利用恶意代码直接存储器访问请求写入内核空间、控制系统堆栈(return-oriented attacks)[10]。因此，本文主要针对的安全威胁是虚拟机的可信执行和内核完整性问题。

在CTVM的设计中，假设平台提供者为了市场声誉和占有率而积极保护用户数据安全，可认为它是可信的，并且不考虑来自平台提供者机房的物理攻击。为了构建可信的内核虚拟机计算环境，假设用于部署虚拟机的物理主机上都配置有硬件TPM，可信计算组织(trusted computing group, TCG)曾指出TPM芯片可用于验证硬件设备[8]。文献[11]介绍了如何使用TPM构建可信计算基，而利用Intel的可信执行技术(trusted execution technology, TXT)可实现静态和动态的度量[12]，本文利用TXT度量宿主机启动和关键安全模块的完整性。

1.2 设计目标

基于上述安全威胁，CTVM的设计目标包括虚拟可信执行环境和虚拟机内核完整性保护。

在内核虚拟机中创建可信计算环境，将信任链从宿主机传递到虚拟机，使内核虚拟机具有与宿主机功能一致的可信计算环境，从而保护GVM的启动完整性，并为GVM提供基于可信计算的安全功能。进一步实现对GVM执行环境的可信验证机制以及类似(trusted virtual datacenter, TVDc)[13]的多层可信域分层模型，本文暂不对这些方面进行描述。

GVM运行时可能有意或无意加载不可信内核模块。针对这类安全威胁，在宿主机和虚拟机可信启动的基础上，基于硬件辅助虚拟化技术，CTVM系统对虚拟机内核与不可信模块进行地址空间逻辑隔离，限制不可信模块对内核代码及数据的访问，实现对平台运行过程中的内核完整性保护。

1.3 总体架构

CTVM架构如图1所示。

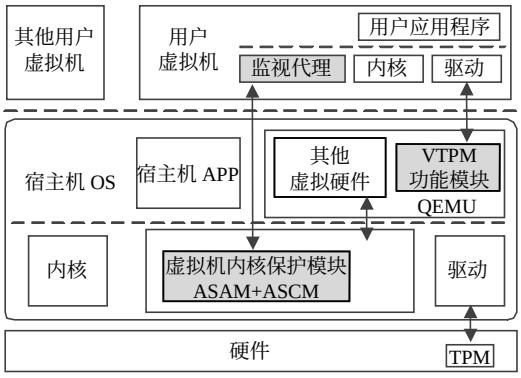

图1 CTVM架构

该系统分为监视代理、虚拟机内核保护模块和VTPM功能模块。轻量级的监视代理被部署于GVM，是一种可集成到类似VMware tools的驱动模块；虚拟机内核保护模块位于KVM代码中；VTPM位于QEMU的组件中，为GVM提供与物理TPM一致的可信计算功能，同时为系统提供启动完整性验证。

1) 监视代理模块

监视代理模块(monitor agent module, MAM)是一个轻量级的信息获取模块，其功能是在GVM中动态收集模块地址信息，为KVM中的安全模块与虚拟机提供通信功能，便于安全模块准确解析上层虚拟机内部信息，在虚拟机内核保护模块提供监视与隔离功能时，解决虚拟机与KVM之间的地址语义鸿沟问题。

2) 虚拟机内核保护模块

虚拟机内核保护模块与MAM一起工作, 对内核的不可信模块进行隔离，使不可信模块和内核运行在相互隔离的地址空间，可根据安全策略监视模块

运行过程中是否存在违反安全策略的行为。模块包括两个子模块：①模块地址空间创建子模块(address space creating module, ASCM)，用于KVM中隔离地址空间的创建；②地址空间切换与授权子模块(address space authorization module, ASAM)，其功能是完成虚拟机内核与模块之间的地址空间切换和授权操作。

3) VTPM功能模块

VTPM功能模块主要是在QEMU中实现对TPM的软件模拟，作为虚拟机TPM TIS驱动与QEMU虚拟TIS前端通信时的后端支持，并集成满足TCG-BIOS扩展规范的虚拟BIOS功能，使每台GVM拥有独立访问的VTPM，可利用IBM IMA[14]实现对虚拟机系统的启动完整性度量、文件加密等可信计算相关的功能。另外，由于监视代理自身安全性也面临威胁，可在虚拟机中利用VTPM对MAM进行启动完整性验证，而KVM中的安全模块则可由宿主机的物理TPM提供验证功能。

4) 模块运行方式

系统启动后，宿主机内核和安全模块的启动完整性由物理TPM验证，而GVM内核与监视代理的启动完整性可由QEMU分配的VTPM验证。完成可信启动后，MAM模块将在虚拟机生命周期内监视模块加载操作，加载不可信模块时，MAM将搜集该模块的虚拟内存信息，并将这些信息通过超级调用传递给KVM。KVM中的ASCM接收到模块地址信息后，为不可信模块创建一套extended page table(EPT)，使其运行在独立地址空间，并完成地址映射操作。之后的运行过程由ASAM完成地址空间切换和授权，使不可信模块始终运行在隔离的地址空间。

2　CTVM系统的实现

2.1　虚拟可信执行环境

虚拟可信执行环境的构造主要依赖于VTPM对虚拟机的支持，即在虚拟机中能访问具有可信计算功能的模块，该模块可以是软件也可以是硬件。本文在实现过程中，尝试了穿透模式和软件模拟两种方法。

1) 穿透模式

在QEMU-KVM的虚拟化环境中，以原版1.7.1的QEMU为例，若宿主机配置有硬件TPM模块，QEMU可以为虚拟机提供穿透驱动的模式访问TPM，即在GVM中可以访问到宿主机配置的物理TPM，穿透模式如图2所示。

虚拟机通过TPM TIS驱动通知QEMU模拟的TPM TIS前端，再由模拟的TPM TIS与宿主机的/dev/tpm0通信。但这种方式属于独占式访问，宿主机上的TPM不能与其他GVM共享，甚至当GVM访问TPM时，宿主机操作系统也不能访问物理TPM。

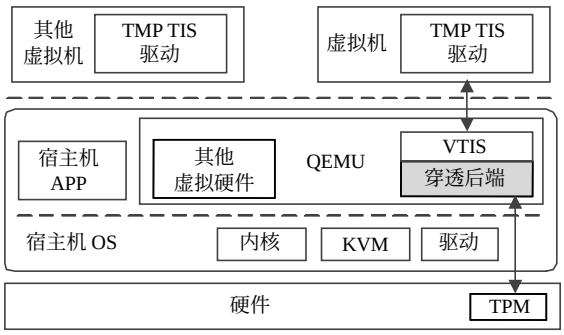

图2　穿透模式

2) 软件模拟

由于穿透模式的共享问题不适合多台虚拟机都有TPM需求的应用场景，本文的CTVM系统以软件模拟的方式构造虚拟可信执行环境，软件模拟如图3所示。

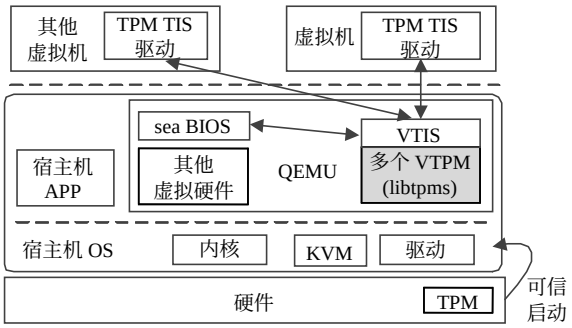

图3　软件模拟

与穿透模式不同，软件模拟可模拟出多个VTPM实例，满足多台虚拟机同时拥有VTPM的需求。图3中物理TPM可以满足宿主机的可信计算需求，保证宿主机和安全模块的启动完整性，将安全模块加入到启动度量列表后，系统还能提供宿主机和安全模块的启动完整性度量结果。多台虚拟机对VTPM的需求主要依靠软件模拟实现。利用libtpms软件库作为VTPM的后端，在QEMU的代码中加入libtpms、虚拟机NVRAM、虚拟BIOS支持后，即可为多台虚拟机同时提供VTPM。虚拟NVRAM的作用是提供一个存储TPM NVRAM值的可持续镜像文件，避免系统重启后丢失VTPM的特征信息(如endorsement key, owner password等)。seaBIOS满足TCG BIOS扩展规范，可以初始化VTPM，并为虚拟机提供可操作TPM状态的虚拟机BIOS界面。

2.2 虚拟机内核模块隔离

在CTVM系统中，对虚拟机不可信模块的隔离依靠两种技术：创建隔离地址空间和地址空间切换。

1) 创建隔离地址空间

本文借助硬件辅助虚拟化技术，客户代理传入被隔离模块的地址信息后，由KVM为不可信模块单独创建一套EPT(MEPT)，使其运行在独立的地址空间内。MEPT只保存模块自身代码和数据的映射关系，并去除虚拟机内核EPT(KEPT)中隔离模块所在内存页的可执行权限，内核和模块EPT权限设置如图4所示。

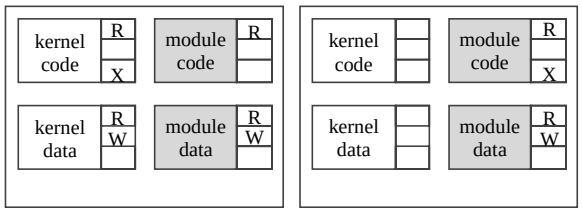

图4 内核和模块EPT权限设置

图中，kernel code表示GVM的内核代码段；module code表示GVM的模块代码段；kernel data表示GVM的内核数据段；module data表示GVM的模块数据段；R、W和X分别对应读、写和执行3种权限。

执行流程在内核和模块之间切换时会触发EPT violation并陷入到KVM中，后续工作中可在这里建立访问控制机制，监控那些引起控制流转移的GVM地址。另外，当模块需要访问内核数据时，可以内存页为单位进行授权，若模块对该内存页有访问权限，则将其映射进模块地址空间。

2) 地址空间切换

GVM在运行过程中，内核需与被隔离模块彼此交互，该过程会引起控制流在内核与模块之间转移，KVM需同步切换到对应页表，以便GVM能正常寻址内存。切换前，KVM需保证页表切换后，GVM能正常运行。所以每次页表切换时，KVM需查询对方页表，确认存在GPA(引发当前EPT violation的GVM物理地址)到HPA(宿主机物理地址)的映射关系，且权限是可执行的。

实际运行中，KEPT可能未与部分模块内存页建立映射关系，此时根据KEPT建立起的MEPT并不完整。当由此触发EPTviolation时，查询KEPT便会发生映射缺失，需根据实际情况做如下处理：

1) 若当前页表为MEPT，需先切换到KEPT，再交给KVM的EPT violation处理函数处理，最后将建立的映射关系映射到MEPT中并切换到MEPT页表；

2) 若当前页表为KEPT，则直接将执行流程交给KVM的EPT violation处理函数处理，随后将对应映射关系映射到MEPT页表。

在建立对应MEPT的映射项目时，需去除对应KEPT中页表项的可执行权限。考虑到大内存页以及混合内存页的存在，同一内存页中可能同时包含代码与数据，该内存页在KEPT中的映射关系必然包含可执行权限。当模块访问内核数据时，采用最小权限的方式将对应内存页映射进MEPT中。若被访问内存页包含可执行权限，则其对应的MEPT映射将去除可执行位。这样可保证MEPT中拥有执行权限的只有模块代码本身，模块在调用其他代码时会因为产生EPT violation而被KVM捕获。

3 系统测试与分析

为验证CTVM系统的可用性，本文对虚拟可信执行环境和内核完整性保护技术进行了测试。CTVM原型系统的配置信息如表1所示。

表1 CTVM原型系统的配置信息

名称	物理服务器	虚拟机
CPU	E5-2609 2.40 GHz	E5-2609 2.40 GHz
内存/GB	4	1
二级缓存/MB	10	10
硬盘容量/TB	1	0.1

宿主机操作版本为CentOS6.5 64位，内核版本为3.10.1，QEMU版本为1.7.1，GVM采用Linux发行版本Fedora14进行了测试。

3.1 功能测试

1) 虚拟可信执行环境测试

安装虚拟机系统前，使用qemu-img创建qcow2格式的虚拟机存储镜像和VTPM NVRAM镜像，随后使用qemu-system-x86_64创建和启动带VTPM的可信虚拟机，第一次启动虚拟机时创建VTPM，生成Endorsement Key，初始化NVRAM并添加证书。虚拟机启动后，宿主机TPM和虚拟机VTPM的使用情况如图5所示。

在内核为3.10.1的宿主机上启动了fedora虚拟机，通过vncviewer链接到虚拟机桌面，虚拟机具有/dev/tpm0设备，并且图中可看到部分启动完整性度量信息。若虚拟机启动命令行中添加了boot菜单开启选项，将在启动过程中看到seaBIOS的控制界面，进入虚拟机操作系统安装可信软件栈和tpm-tools

后，可像物理TPM一样查看和操作VTPM。

2) 模块隔离测试

对模块隔离功能主要测试了地址空间的创建和切换，当模块加载到GVM内核时，由MAM获取模块地址信息，通过超级调用传到KVM层，由虚拟机内核保护模块创建MEPT，使模块运行在隔离的地址空间。测试过程中，加载的模块大小为34 672Byte，而宿主机每个物理内存页的大小为4 KB，所以宿主机为模块分配了9个物理内存，并在MEPT中建立了映射，创建和映射模块地址空间如图6所示。

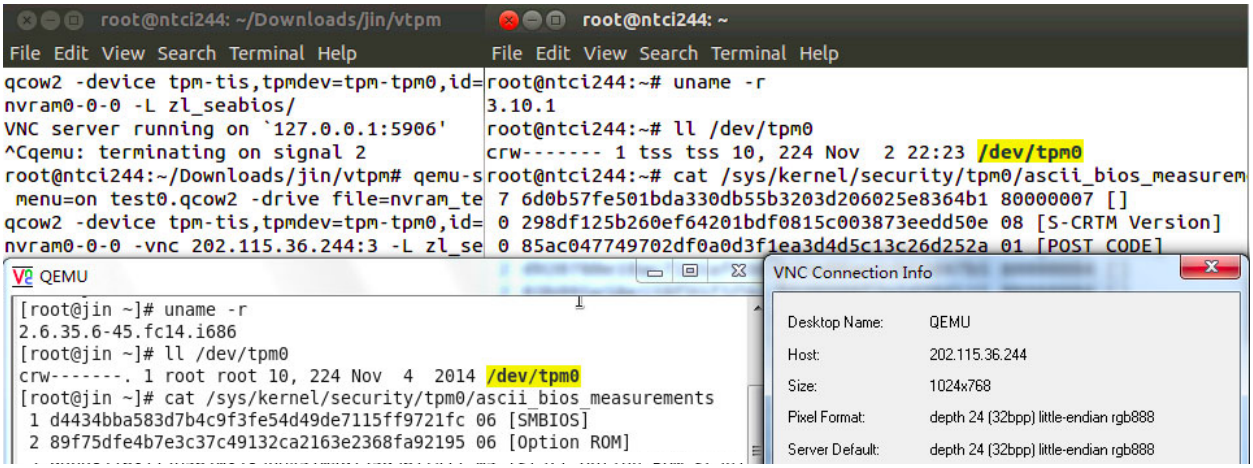

图5　软件模拟测试结果

```
19:31:04 kernel: kmon_debug: vmx_call: 0x1, 0xf7ef7000, 0x87ca
19:31:04 kernel: cons_ept_table: rebuild ept base
19:31:04 kernel: handle_ept_fault, level: 1, vaddr: 0x0, gpa: 0xb74000
19:31:04 kernel: kmon_debug: entered level 2 loop
19:31:04 kernel: handle_ept_fault, level: 1, vaddr: 0x0, gpa: 0x3605a000
19:31:04 kernel: kmon_debug:entered level one loop
19:31:04 kernel: cons_mapping: construct for ept mapping 0x3ee8f000
19:31:04 kernel: cons_mapping_x, constructed at level 1
19:31:04 kernel: kmon_debug: entered level one loop
19:31:04 kernel: cons_mapping: construct for ept mapping 0x3eeb3000
19:31:04 kernel: cons_mapping_x, constructed at level 1
19:31:04 kernel: kmon_debug: entered level one loop
19:31:04 kernel: cons_mapping: construct for ept mapping 0x3eebf000
19:31:04 kernel: cons_mapping_x, ept not present at level 1
19:31:04 kernel: kmon_debug: entered level one loop
19:31:04 kernel: cons_mapping: construct for ept mapping 0x3eeba000
19:31:04 kernel: cons_mapping_x, constructed at level 1
19:31:04 kernel: kmon_debug: entered level one loop
```

图6　创建和映射模块地址空间

```
Nov  4 19:31:04 kernel: switch to module page table
Nov  4 19:31:04 kernel: handle_ept_fault, handled at level: 1,
vaddr: 0xf3737ddc, gpa: 0x33737ddc
Nov  4 19:31:04 kernel: handle_ept_fault, handled at level: 1,
vaddr: 0xf6824fd4, gpa: 0x36824fd4
Nov  4 19:31:04 kernel: handle_ept_iso_fault, ept_vio_vaddr: 0xc0554670
gpa: 0x554670 _eip: 0xc0554670, _esp: 0xf3737d78 exit_qual: 0x184
Nov  4 19:31:04 kernel: switch to kernel page table
Nov  4 19:31:04 kernel: handle_ept_iso_fault, ept_vio_vaddr: 0xf7ef7030
gpa: 0x3ee8f030 _eip: 0xf7ef7030, _esp: 0xf3737d1c exit_qual: 0x19c
Nov  4 19:31:04 kernel: switch to module page table
```

图7　内核和模块地址空间切换

隔离的地址空间创建后，可对模块的行为进行监控，确保GVM内核函数入口的正确性，并防止内核敏感数据被修改。由于将内核和不可信模块各自的代码和数据进行了地址空间隔离，当模块调用内核函数或修改敏感数据时，由于内核模块的权限设置，虚拟机会产生缺页错误并被KVM捕获，执行地址空间切换过程，KVM中的ASAM可根据产生切换的原因进行相应的操作，内核和模块地址空间切换如图7所示。

3.2　性能测试

由于VTPM的启动时间包含在GVM系统启动时间内，与不使用VTPM的启动时间相比，几乎没有在差别，所以本文使用Unixbench 5.1.2软件，主要宿主机添加安全模块后对GVM的性能损耗进行了测试，CTVM原型系统性能测试如表2所示。

表2　CTVM原型系统性能测试

Benchmarks	未加载安全模块	加载安全模块	性能损耗率(%)
Dhrystone	2 284.0	2 278.5	0.24
Whetstone	515.5	513.4	0.41
Execl 函数调用	454.2	450.8	0.75
文件拷贝(4 KB, 8 000)	2 729.2	2 709.1	0.74
文件拷贝(256 Byte, 500)	1 173.7	1 168.4	0.45
文件拷贝(1 KB, 2 000)	1 822.0	1 813.3	0.48
管道吞吐量	974.9	961.9	1.33
基于管道的上下文切换	154.5	149.4	3.30
进程创建	468.0	464.6	0.73
脚本(8)	3 622.3	3 580.6	1.15
系统调用开销	1 200.0	1 198.2	0.20

表中，第2、3列的数值越大性能越好，性能损耗率=(未加载安全模块–加载安全模块)/未加载安全模块。从整体性能看，加载CTVM安全模块后GVM的性能损耗在10%以内。其中，管道上下文切换的性能损耗比较明显，主要原因是隔离区域与非隔离区域转换产生的上下文切换以及异常处理造成的性能损耗，但不会对GVM的性能产生明显影响。

4 总 结

本文提出了一种保护虚拟机内核完整性的技术CTVM。在KVM虚拟化环境中利用libtpms为GVM实现了VTPM，并为GVM创建了可独立使用的可信计算环境。在虚拟机内核可信启动的基础上，针对运行时的内核完整性提出了两套EPT机制，为不可信模块创建隔离的运行空间，可动态监控模块的行为，防止GVM内核完整性遭到破坏。实验结果表明，CTVM原型系统可实现虚拟可信执行环境的创建，能为不可信模块创建隔离地址空间并监视模块的执行，安全模块的添加对系统整体性能的影响控制在可接受范围内。下一步的工作目标是KVM虚拟化环境中的TVDc创建，并对运行时内核完整性保护技术进行优化，添加较为完整的安全控制策略。

参 考 文 献

[1]ROCHA F, CORREIA M. Lucy in the sky without diamonds: Stealing confidential data in the cloud[C]//2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops(DSN-W). Hong Kong, China: IEEE, 2011.

[2] DOLAN-GAVITT B, LEEK T, ZHIVICH M, et al. Virtuoso: Narrowing the semantic gap in virtual machine introspection [C]//2011 IEEE Symposium on Security and Privacy (SP). Berkeley, CA: IEEE, 2011.

[3] CHEN X, GARFINKEL T, LEWIS E C, et al. Overshadow: a virtualization-based approach to retrofitting protection in commodity operating systems[C]//Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating System. New York, USA: ACM, 2008.

[4] CHEN H, CHEN J, MAO W, et al. Daonity-grid security from two levels of virtualization[J]. Information Security Technical Report, 2007, 12(3): 123-138.

[5] HOFMANN O S, KIM S, DUNN A M, et al. Inktag: Secure applications on an untrusted operating system[J]. ACM SIGPLAN Notices, 2013, 48(4): 265-278.

[6] ZHANG F, CHEN J, CHEN H, et al. Cloudvisor: Retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization[C]//Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. New York, USA: ACM, 2011.

[7] BEN-YEHUDA M, DAY M D, DUBITZKY Z, et al. The turtles project: Design and implementation of nested virtualization[C]//9th USENIX Symposium on Operation Systems Design and Implementation(OSDI'10). Vancouver, BC: USENIX Association, 2010.

[8] TOMLINSON A. Smart cards, tokens, security and applications[M]. New York, USA: Springer, 2008.

[9] KORKL, JAGADPRAMANA P, MOWBRAY M, et al. Trustcloud: a framework for accountability and trust in cloud computing[C]//2011 IEEE World Congress on Services. Washington, USA: IEEE, 2011.

[10] SRIVASTAVA A, GIFFIN J T. Efficient monitoring of untrusted kernel-mode execution[C]//18th Annual Network & Distributed System Security Symposium. San Diego, USA: The Internet Society NDSS, 2011.

[11] GARFINKEL T, PFAFF B, CHOW J, et al. Terra: a virtual machine-based platform for trusted computing[C]// Proceedings of the nineteenth ACM symposium on Operating systems principles. New York, USA: ACM, 2003.

[12] GEBHARDT C, DALTON C I, BROWN R. Preventing hypervisor-based rootkits with trusted execution technology[J]. Network Security, 2008, 8(11): 7-12.

[13]BERGER S, CACERES R, PENDARAKIS D, et al. TVDc: Managing security in the trusted virtual datacenter[J]. ACM SIGOPS Operating Systems Review, 2008, 42(1): 40-47.

[14] SAILER R, ZHANG X, JAEGER T, et al. Design and implementation of a TCG-based integrity measurement architecture[C]//Proceedings of the 13th USENIX Security Symposium. San Diego, USA: USENIX Association, 2004.

编 辑 叶 芳