(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 106970823 A
(43)申请公布日 2017.07.21

(21)申请号 201710104010.3

(22)申请日 2017.02.24

(71)申请人 上海交通大学
地址 200240 上海市闵行区东川路800号

(72)发明人 陈海波 刘宇涛 臧斌宇

(74)专利代理机构 上海汉声知识产权代理有限公司 31236
代理人 郭国中

(51)Int.Cl.
G06F 9/455(2006.01)
G06F 21/60(2013.01)

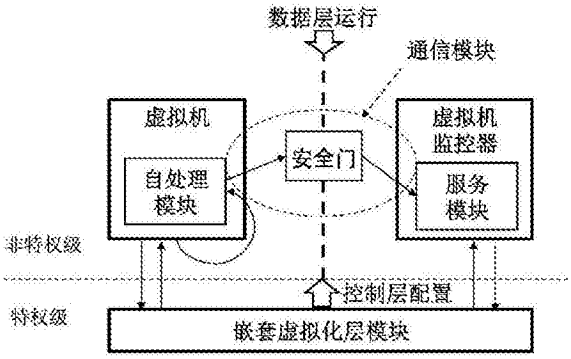
权利要求书2页 说明书10页 附图3页

(54)发明名称

高效的基于嵌套虚拟化的虚拟机安全保护方法及系统

(57)摘要

本发明提供了一种高效的基于嵌套虚拟化的虚拟机安全保护方法及系统，包括：步骤1：将虚拟机与虚拟化层进行隔离；步骤2：实现虚拟化层软件与虚拟机的直接交互。具体地，利用嵌套虚拟化技术防止恶意的虚拟化层软件窃取和篡改虚拟机内存和磁盘的数据；对虚拟化层的操作进行检查和监控，以及对嵌套虚拟化层的软件进行安全性验证，防止恶意的虚拟化层软件篡改虚拟机的控制流。本发明相比现有的基于嵌套虚拟化的虚拟机安全解决方案，能够大量避免由于隔离机制所造成的虚拟机下陷，从而在不损失安全性的前提下，大幅度提升整个系统的性能，能够满足当下对于包括安全性、功能性和高效性在内的各方面的需求。



CN 106970823 A

1.一种高效的基于嵌套虚拟化的虚拟机安全保护方法，其特征在于，包括如下步骤：

步骤1：将虚拟机与虚拟化层进行隔离；

步骤2：实现虚拟化层软件与虚拟机的直接交互。

2.根据权利要求1所述的高效的基于嵌套虚拟化的虚拟机安全保护方法，其特征在于，所述步骤1包括：利用嵌套虚拟化技术防止恶意的虚拟化层软件窃取和篡改虚拟机内存和磁盘的数据；对虚拟化层的操作进行检查和监控，以及对嵌套虚拟化层的软件进行安全性验证，防止恶意的虚拟化层软件篡改虚拟机的控制流。

3.根据权利要求1所述的高效的基于嵌套虚拟化的虚拟机安全保护方法，其特征在于，所述步骤2包括：在虚拟机和虚拟化层之间设置安全通道，所述安全通道用于实现虚拟机与虚拟化层软件的直接交互。

4.根据权利要求1至3中任一项所述的高效的基于嵌套虚拟化的虚拟机安全保护方法，其特征在于，所述步骤1包括：

步骤1.1：将虚拟机和虚拟机监控器运行在非特权级的环境中；

步骤1.2：对非特权级环境中的实体进行内存隔离配置；所述实体包括：虚拟机、虚拟机监控器以及安全门，当虚拟机监控器恶意访问虚拟机内存时，会触发下陷，由嵌套虚拟化层进行检查并禁止；

步骤1.3：在嵌套虚拟化层为虚拟机和虚拟机监控器分别配置相关的虚拟机控制结构中特定的域，所述域用于控制虚拟机和虚拟机监控器运行时的行为。

5.根据权利要求2所述的高效的基于嵌套虚拟化的虚拟机安全保护方法，其特征在于，所述步骤2包括：

步骤2.1：在虚拟机和虚拟机监控器之间建立非特权级环境中的代理，所述代理即为虚拟机和虚拟机监控器之间的安全门，并设置相应的跳板函数，对安全门和跳板函数进行安全性的验证；

步骤2.2：虚拟机在运行时如果产生了需要虚拟机监控器处理的事件，则通过安全门中的代码直接向虚拟机监控器发出请求，由虚拟机监控器提供的服务进行相应的处理，即包括处理内存分配，异常处理，中断处理，I/O处理等，并在处理结束之后通过安全门返回，并进行隐私性和完整性的检查。

6.一种高效的基于嵌套虚拟化的虚拟机安全保护系统，其特征在于，包括：嵌套虚拟化层模块、虚拟机自处理事件模块、虚拟机监控器服务模块以及通信模块；

所述嵌套虚拟化层模块用于对控制层的非特权级实体内存进行配置，使得各个非特权级实体之间相互隔离，并配置通信模块；所述非特权级实体包括：虚拟机、虚拟机监控器、虚拟机与虚拟机监控器之间的安全门；

所述虚拟机自处理事件模块用于在发生特定事件时对相关事件进行自行处理，并触发上下文切换；

所述虚拟机监控器服务模块用于提供相应的服务，该服务包括：分配页表、进行中断处理、部分异常处理以及I/O处理；

所述通信模块用于实现虚拟机和虚拟机监控器之间的交互。

7.根据权利要求6所述的高效的基于嵌套虚拟化的虚拟机安全保护系统，其特征在于，所述通信模块在虚拟机和虚拟机监控器之间插入一个嵌套虚拟化在非特权级中的代理，并

为虚拟机、虚拟机监控器以及代理提供相应的跳板函数，所述代理即为安全门，该安全门是一段代码逻辑。

高效的基于嵌套虚拟化的虚拟机安全保护方法及系统

技术领域

[0001] 本发明涉及云计算平台系统安全技术领域，具体地，涉及高效的基于嵌套虚拟化的虚拟机安全保护方法及系统。

背景技术

[0002] 在云计算平台中，虚拟化技术通常被用来高效整合服务器，提高包括处理器、内存在内的硬件的利用率。同时，虚拟化技术也通常被用于做虚拟机之间的隔离，防止恶意虚拟机访问其他虚拟机中的内存和磁盘存储的隐私数据。但是，随着多租户云计算平台中虚拟化层软件栈的日益庞大和复杂，整个系统的可信基也逐渐增大，租户在虚拟机中存储的隐私数据面临着来自外部和内部共同的威胁。一方面，随着虚拟化层的功能性日益繁多，虚拟化层软件栈的代码量也日益扩大，其中漏洞数目也越来越多，如根据CVE的统计，截止到2015年，当前主流的3大虚拟化平台VMWare、Xen和KVM分别被披露了234个、135个和89个已知的漏洞，外部攻击可以通过利用这些漏洞控制虚拟化层软件栈，从而威胁整个系统的安全性。另一方面，在云平台内部也存在着不安全因素，比如随着安全事故的发生(如2010年Google公司的雇员窃取用户的隐私数据)，云平台中恶意的管理人员也成为了一个潜在的威胁。为了增强虚拟化平台的安全性，避免过大的攻击面，之前的研究提出将虚拟化层的软件栈(虚拟机监控器)移除出可信计算基。其中最具代表性的工作是CloudVisor。

[0003] 由于在传统的虚拟化框架中，虚拟化监控器处于整个系统的最高权限层，一旦其被控制，是没有办法通过软件的方法进行阻止的，因此，CloudVisor提出加入一层新的抽象层，通过嵌套虚拟化技术，将虚拟机监控器移出最高权限层，从而防止恶意的虚拟机监控器对整个系统造成破坏。研究指出，在虚拟化环境中，虚拟机隐私和完整性的保护和其他功能性的逻辑是相互独立的，因此CloudVisor系统通过将安全保护机制的逻辑从整个虚拟化层剥离出来，在嵌套虚拟化层对虚拟机监控器进行监控，从而保护虚拟机，防止虚拟机中的隐私数据被虚拟化层窃取和篡改。具体来说，如图1所示，在CloudVisor的架构中，虚拟机监控器将被置于非特权指令集中，并使用扩展页表(EPT)机制保护用户虚拟机不受到未授权的访问。虚拟机监控器对用户虚拟机进行的所有敏感操作都将下陷到CloudVisor中接受安全检查。这种机制很好地将安全性与功能性解耦合，在保证功能完备的同时尽量减小了可信计算基的大小，有利于对可信基进行进一步的正确性验证。因此在CloudVisor中可信基仅仅有5.5K行代码，相比于庞大的虚拟机监控器具有更加简单的逻辑及更高的可信性。

[0004] 然而，CloudVisor这种基于嵌套虚拟化的方案存在一个严重的问题：由于嵌套虚拟化层的引入，使得虚拟机到虚拟化层的路径必须经过嵌套虚拟化层，这就对整个系统造成了巨大的性能损失。如图2所示，在传统的虚拟化框架中，虚拟机的特权指令或者特定的事件直接被虚拟化层拦截，通过一次虚拟机下陷(VMExit)和一次虚拟机进入(VMEntry)即可完成整个路径。然而在CloudVisor框架中，所有的操作首先进入嵌套虚拟化层，再由嵌套虚拟化层的软件将控制权交还给虚拟化层，在虚拟化层处理完之后，再进入嵌套虚拟化层，最后返回虚拟机。另外，在虚拟化层处理过程中，由于虚拟化层属于非特权级的环境中，某

些特定的特权操作还会下陷到嵌套虚拟化层进行模拟，因此，原来一次VMExit就能完成的操作，需要至少两次(一般情况下会远远大于两次)的VMExit才能完成，从而对整个系统造成巨大的性能损失。

[0005] 造成这些性能损失的根本原因在于两点：第一，由于之前硬件提供的功能和特性，很多操作和事件必须要下陷到特权级模式(即嵌套虚拟化层中)进行处理，没有一套可行的方法能够让处于非特权级中的相互隔离的两个实体(虚拟机和虚拟机监控器)直接进行交互；第二，出于对安全的考量，嵌套虚拟化层必须对两个实体(虚拟机和虚拟机监控器)的交互进行拦截，对虚拟机监控器的操作进行检查，从而防止恶意的虚拟机监控器窃取和篡改虚拟机的内存磁盘数据。

[0006] 然而，随着硬件的发展，第一个根本原因(即在特定操作和事件中必然发生VMExit)已经不再是必须的了。比如，Intel对硬件虚拟化提供了更为高级的支持，许多硬件特性都使得越来越多的操作和事件都可以被配置为不引起VMExit，以下列举6个相应的硬件特性实例来进行详细的说明：

[0007] 1)VMFunc(EPTP switching)：在虚拟化环境中，内存虚拟化是通过两级页表映射实现的，如图1所示，内存虚拟化的两级页表机制客户虚拟机中每个客户虚拟地址(GVA)首先会被映射到一个客户物理地址(GPA)，之后在VMM里面，每个客户物理地址又会被映射为一个宿主机物理地址(HPA)。也就是说在VMM里面会维护一个页表(Intel中被称为EPT)，该页表会对所有客户虚拟机中的物理地址进行映射，只有存在于该页表中的映射的客户机地址才会被反映在真实的物理内存中。该页表通过一个硬件寄存器进行访问，该寄存器被称为EPT Pointer(EPTP)。理论上，完全可以通过操作EPT(即改变GPA到HPA之间的映射)来操作客户机中的内存分配，但是在正常模式下，修改EPT中地址的映射关系都是由VMM进行操作的，在虚拟机中切换不同的EPTP会引起虚拟机下陷，造成比较大的性能开销。而VMFunc是Intel提出的一套硬件机制，EPTP Switching是其包含的一项功能，即允许在非特权级的模式下能够直接切换相应的EPTP，而不引发VMExit。当然，前提是特权级软件预先配置好可选的EPTP，而非特权级只允许在有限的几个配置好的EPTP中进行切换。

[0008] 2)Virtualization Exception(VE)：在传统的虚拟化环境中，在虚拟机运行时如果发现当前访问的物理页在EPT中没有被映射，则会发生一个事件，即EPT Violation，该事件会造成VMExit，从而由特权级的软件进行处理(比如，填充相应的EPT项)。而VE则是Intel提出的一个硬件特性，在进行了相应的配置之后，可以使得EPT Violation不会引发VMExit，而是作为一个普通的异常(Exception)，直接触发非特权级(如虚拟机)中的异常处理函数进行处理。

[0009] 3)VMCS shadowing：在Intel硬件虚拟化中的CPU虚拟化中，所有和CPU相关的信息都会被保存在一个数据结构中，该数据结构被称为VMCS。这些数据结构保存了虚拟机(VM)和主机(host)的状态，以及一些控制信息。VMCS会被加载在特定的内存中，当发生VMEntry的时候，处理器会加载VMCS中和虚拟机相关的状态，在发生VMExit的时候，处理器会加载VMCS中和主机相关的状态。在传统的虚拟化环境中，对VMCS的访问只能由特权级软件进行，非特权级中产生的访问会引发VMExit，因此如果将虚拟机监控器放置在非特权级中，由于其会频繁访问VMCS，因此会造成大量的VMExit。而VMCS shadowing机制使得在经过相应的映射之后，非特权级模式下的软件也能够直接修改内存中的VMCS。

[0010] 4)APIC virtualization:在Intel多核处理器中,系统软件如果处理中断相关事务,需要与APIC(Advanced Programmable Interrupt Controller)交互。APIC的功能包括两点:1)从内部或外部接收中断,并传送给相应的处理器;2)在多核系统中为某处理器向其他处理器发送处理器间中断(IPI)。然而,在虚拟化环境中,虚拟机无法直接接触APIC,需要由虚拟机监控器为其模拟APIC操作。具体而言,客户虚拟机任何访问APIC的操作都会造成一次下陷(VMExit),之后虚拟机监控器检查相应的下陷信息,从而为虚拟机完成对应的模拟。为了消除APIC模拟带来的下陷开销,Intel推出了APIC虚拟化技术,由硬件为每个虚拟机提供APIC的模拟。此时,客户虚拟机的APIC操作将不再下陷。

[0011] 5)Posted interrupt:APIC虚拟化技术支持Posted Interrupt Processing,允许VMM向一个正在运行的虚拟CPU直接发送中断,这就避免了一次由中断带来的下陷。然而,发送Posted Interrupt依然要求另一个处理器上运行着虚拟机监控器。

[0012] 6)IOMMU:为了使虚拟机监控器不参与整个中断的转发,将来自物理设备的中断直接发送给某虚拟CPU。可以使用最新的IOMMU技术。借助IOMMU,VMM将某物理设备直接绑定给一个虚拟机,由其独占此设备。来自独占设备中断的转发需要借助IOMMU中的中断重映射表(Interrupt Remapping Table)。在中断重映射表中,虚拟机监控器设置某中断的转发类型,决定硬件如何处理这个中断。若此中断的类型是Posting模式,硬件将借助APIC虚拟化中的Posted Interrupt Processing功能,直接将其插入正在运行的虚拟CPU中。

[0013] 上述这些新的硬件特性提供了一个机遇,即在发生某些特定事件和操作的时候不引发VMExit,而是由非特权级模式中的实体直接进行处理,从而避免了大量VMExit造成的性能损失。同时,由于VMFunc的存在,为非特权级模式下不同实体之间直接交互带来了可能性,可以通过改变EPTP达到不同实体间的上下文切换的效果。因此,在以上所提到的硬件支撑的基础上,如何在不产生VMExit的情况下保证虚拟机和虚拟机监控器的隔离性,将是需要解决的关键问题。

发明内容

[0014] 针对现有技术中的缺陷,本发明的目的是提供一种高效的基于嵌套虚拟化的虚拟机安全保护方法及系统。

[0015] 根据本发明提供的高效的基于嵌套虚拟化的虚拟机安全保护方法,包括如下步骤:

[0016] 步骤1:将虚拟机与虚拟化层进行隔离;

[0017] 步骤2:实现虚拟化层软件与虚拟机的直接交互。

[0018] 优选地,所述步骤1包括:利用嵌套虚拟化技术防止恶意的虚拟化层软件窃取和篡改虚拟机内存和磁盘的数据;对虚拟化层的操作进行检查和监控,以及对嵌套虚拟化层的软件进行安全性验证,防止恶意的虚拟化层软件篡改虚拟机的控制流。

[0019] 优选地,所述步骤2包括:在虚拟机和虚拟化层之间设置安全通道,所述安全通道用于实现虚拟机与虚拟化层软件的直接交互。

[0020] 优选地,所述步骤1包括:

[0021] 步骤1.1:将虚拟机和虚拟机监控器运行在非特权级的环境中;

[0022] 步骤1.2:对非特权级环境中的实体进行内存隔离配置;所述实体包括:虚拟机、虚

拟机监控器以及安全门，当虚拟机监控器恶意访问虚拟机内存时，会触发下陷，由嵌套虚拟化层进行检查并禁止；

[0023] 步骤1.3：在嵌套虚拟化层为虚拟机和虚拟机监控器分别配置相关的虚拟机控制结构中特定的域，所述域用于控制虚拟机和虚拟机监控器运行时的行为。

[0024] 优选地，所述步骤2包括：

[0025] 步骤2.1：在虚拟机和虚拟机监控器之间建立非特权级环境中的代理，所述代理即为虚拟机和虚拟机监控器之间的安全门，并设置相应的跳板函数，对安全门和跳板函数进行安全性的验证；

[0026] 步骤2.2：虚拟机在运行时如果产生了需要虚拟机监控器处理的事件，则通过安全门中的代码直接向虚拟机监控器发出请求，由虚拟机监控器提供的服务进行相应的处理，即包括处理内存分配，异常处理，中断处理，I/O处理等，并在处理结束之后通过安全门返回，并进行隐私性和完整性的检查。

[0027] 根据本发明提供的高效的基于嵌套虚拟化的虚拟机安全保护系统，包括：嵌套虚拟化层模块、虚拟机自处理事件模块、虚拟机监控器服务模块以及通信模块；

[0028] 所述嵌套虚拟化层模块用于对控制层的非特权级实体内存进行配置，使得各个非特权级实体之间相互隔离，并配置通信模块；所述非特权级实体包括：虚拟机、虚拟机监控器、虚拟机与虚拟机监控器之间的安全门；

[0029] 所述虚拟机自处理事件模块用于在发生特定事件时对相关事件进行自行处理，并触发上下文切换；

[0030] 所述虚拟机监控器服务模块用于提供相应的服务，该服务包括：分配页表、进行中断处理、部分异常处理以及I/O处理；

[0031] 所述通信模块用于实现虚拟机和虚拟机监控器之间的交互。

[0032] 优选地，所述通信模块在虚拟机和虚拟机监控器之间插入一个嵌套虚拟化在非特权级中的代理，并为虚拟机、虚拟机监控器以及代理提供相应的跳板函数，所述代理即为安全门，该安全门是一段代码逻辑。

[0033] 与现有技术相比，本发明具有如下的有益效果：

[0034] 本发明提供的高效的基于嵌套虚拟化的虚拟机安全保护方法以及系统，能够将虚拟机和虚拟化层进行隔离，防止恶意的虚拟机监控器窃取和篡改虚拟机的数据，并且在运行时尽可能地减少由隔离机制所带来的虚拟机下陷，使得虚拟机和虚拟机监控器能够进行安全且高效的切换。

附图说明

[0035] 通过阅读参照以下附图对非限制性实施例所作的详细描述，本发明的其它特征、目的和优点将会变得更明显：

[0036] 图1为CloudVisor的架构示意图；

[0037] 图2为本发明中的系统架构示意图；

[0038] 图3为本发明中多个实体之间的内存隔离机制示意图；

[0039] 图4为本发明中虚拟机和虚拟机监控器之间通信模块的示意图；

[0040] 图5为本发明中处理虚拟机EPT violation的流程示意图；

[0041] 图6为本发明中处理虚拟机中基于IOMMU设备的I/O操作的流程示意图；

[0042] 图7为本发明中处理虚拟机中基于半虚拟化技术的I/O操作的流程示意图。

具体实施方式

[0043] 下面结合具体实施例对本发明进行详细说明。以下实施例将有助于本领域的技术人员进一步理解本发明，但不以任何形式限制本发明。应当指出的是，对本领域的普通技术人员来说，在不脱离本发明构思的前提下，还可以做出若干变化和改进。这些都属于本发明的保护范围。

[0044] 根据本发明提供的基于嵌套虚拟化的虚拟机安全保护方法，包括如下步骤：

[0045] 步骤1：将虚拟机与虚拟化层进行隔离；

[0046] 步骤2：实现虚拟化层软件与虚拟机的直接交互。

[0047] 所述步骤1包括：利用嵌套虚拟化技术防止恶意的虚拟化层软件窃取和篡改虚拟机内存和磁盘的数据；对虚拟化层的操作进行检查和监控，以及对嵌套虚拟化层的软件进行安全性验证，防止恶意的虚拟化层软件篡改虚拟机的控制流。

[0048] 所述步骤2包括：在虚拟机和虚拟化层之间设置安全通道，所述安全通道用于实现虚拟机与虚拟化层软件的直接交互。这种方式避免大量的虚拟机下陷(到嵌套虚拟化层)所产生的性能损失，同时保证虚拟机和虚拟化层之间的隔离。

[0049] 所述步骤1包括：

[0050] 步骤1.1：将虚拟机和虚拟机监控器运行在非特权级的环境中；

[0051] 步骤1.2：在虚拟机和虚拟机监控器之间建立非特权级环境中的代理，所述代理即虚拟机和虚拟机监控器之间的安全门，并设置相应的跳板函数，对安全门和跳板函数进行安全性的验证；

[0052] 步骤1.3：对非特权级环境中的实体进行内存隔离配置；所述实体包括：虚拟机、虚拟机监控器以及安全门；

[0053] 步骤1.4：为虚拟机和虚拟机监控器分别配置相关的虚拟机控制结构中特定的域；

[0054] 步骤1.5：虚拟机在运行时如果产生了需要虚拟机监控器处理的事件，则通过安全门直接向虚拟机监控器发出请求，由虚拟机监控器进行相应的处理，即包括处理内存分配，异常处理，中断处理，I/O处理等，并在处理结束之后进行隐私性和完整性的检查。

[0055] 根据本发明提供的高效的基于嵌套虚拟化的虚拟机安全保护系统，包括：嵌套虚拟化层模块、虚拟机自处理事件模块、虚拟机监控器服务模块以及通信模块；

[0056] 所述嵌套虚拟化层模块用于对控制层的非特权级实体内存进行配置，使得各个非特权级实体之间相互隔离，并配置通信模块；所述非特权级实体包括：虚拟机、虚拟机监控器、虚拟机与虚拟机监控器之间的安全门；

[0057] 所述虚拟机自处理事件模块用于在发生特定事件时对相关事件进行自行处理，并触发上下文切换；

[0058] 所述虚拟机监控器服务模块用于提供相应的服务，该服务包括：分配页表、进行中断处理、部分异常处理以及I/O处理；

[0059] 所述通信模块用于实现虚拟机和虚拟机监控器之间的交互。

[0060] 所述通信模块在虚拟机和虚拟机监控器之间插入一个嵌套虚拟化在非特权级中

的代理，并为虚拟机、虚拟机监控器以及代理提供相应的跳板函数，所述代理即为安全门，该安全门是一段代码逻辑。

[0061] 下面结合附图对本发明中的技术方案做更加详细的说明。

[0062] 图2展示了本发明的整体架构图。总体来说，本发明遵循着将控制层(control plane)和数据层(data plane)分离的思想：首先，在特权级环境中，嵌套虚拟化层模块需要对整个系统进行控制层的配置，包括对非特权级三个实体(虚拟机，虚拟机监控器和它们之间的安全门)的内存进行配置，使得它们相互隔离，以及建立通信模块，并保证其中安全门和跳板函数代码的安全性，使得在运行时的数据层交互直接通过通信模块，而不需要下陷到嵌套虚拟化层；其次，虚拟机自处理事件模块在发生特定事件时能够对相关事件自行处理，并且触发上下文切换，通过嵌套虚拟化层之前配置好的通信模块和虚拟机监控器进行交互；而虚拟机监控器服务模块提供相应的服务，当控制流从通信模块中进入虚拟机监控器时，对特定的请求进行处理；最后，虚拟机和虚拟机监控器两个实体之间的通信模块保证通信的高效性和安全性，特别是在虚拟机监控器返回之后需要对其提供的服务结果进行检查，防止虚拟机监控器窃取和篡改虚拟机的数据。

[0063] 嵌套虚拟化层模块需要配置相应的控制层，具体来说，需要做五件事：(1)将虚拟机和虚拟机监控器运行在非特权级的环境中；(2)建立其在非特权级环境中的代理，即虚拟机和虚拟机监控器之间的安全门，以及相应的跳板函数，并对安全门和跳板函数进行安全性的验证；(3)对非特权级环境中的三个实体(虚拟机，虚拟机监控器和安全门)进行内存隔离的配置；(4)需要为虚拟机和虚拟机监控器分别配置相关的虚拟机控制结构(Virtual Machine Control Structure，VMCS)中特定的域，从而决定在运行时哪些事件会造成VMExit，哪些事件可以直接由虚拟机或者虚拟机监控器直接处理；(5)对于那些需要下陷的事件进行相应的处理。

[0064] 对于第(3)点，即对三个实体进行内存配置，如图3所示，具体来说，就是通过Intel内存硬件虚拟化提供的扩展页表(EPT)支持，来控制三个实体所能够访问的内存区域的范围，即为三个实体配置三个EPT页表，其中虚拟机和虚拟机监控器的内存区域是相互隔离的，同时它们都无法访问安全门所在的内存区域，从而保证它们无法篡改通信模块的控制流，而通信模块中的安全门可以访问虚拟机和虚拟机监控器的内存，并且可以在这两个内存区域间进行切换，切换的过程在通信模块中详述。对于第(4)点，

[0065] 即配置VMCS以决定需要下陷的事件，表1展示了所有可能造成虚拟机下陷(VMExit)的60个事件，以及通过嵌套虚拟化层对VMCS进行配置从而决定相关事件是否最终引起VMExit的结果。

[0066] 表1

[0067]

编号	描述	VM下陷	VMM下陷	编号	描述	VM下陷	VMM下陷	编号	描述	VM下陷	VMM下陷
0	NMI	否	是	20	VMLAUNCH	是	是	43	TPR below threshold	否	是
1	External interrupt	部分	部分	21	VMPTRLD	是	是	44	APIC access	否	是
2	Triple fault	是	是	22	VMPTRST	是	是	45	Virtualized EOI	否	是
3	INIT signal	是	是	23	VMREAD	是	部分	46	Access to G(I)DTR	否	否
4	SIPI	是	是	24	VMRESUME	是	是	47	Access to LDTR/TR	否	否
5	SMI	是	是	25	VMWRITE	是	部分	48	EPT violation	否	是
6	Other SMI	是	是	26	VMXOFF	是	是	49	EPT mis-configuration	是	是

[0068]

7	Interrupt window	是	是	27	VMXON	是	是	50	INVEPT	是	是
8	NMI window	是	是	28	Control-register access	否	否	51	RDTSCP	否	否
9	Task switch	是	是	20	MOV DR	否	否	52	VMX-preemption	是	是
10	CPUID	是	是	30	I/O instruction	部分	是	53	INVVPID	是	是
11	GETSEC	是	是	31	RDMSR	部分	部分	54	WBINVD	否	否
12	HLT	否	是	32	WRMSR	部分	部分	55	XSETBV	是	是
13	INVD	是	是	33	VM-entry (invalid g state)	是	是	56	APIC write	否	否
14	INVLPG	否	否	34	VM-entry fail (MSR load)	是	是	57	RDRAND	否	否
15	RDPMC	否	否	36	MWAIT	否	否	58	INVPCID	否	否
16	RDTSC	否	否	37	Monitor trap flag	否	否	59	VMFUNC	否	否
17	RSM	是	是	39	MONITOR	否	否	61	RDSEED	否	否
18	VMCALL	是	是	40	PAUSE	否	否	63	XSAVES	否	否
19	VMCLEAR	是	是	41	VM-entry fail (machine check)	是	是	64	XRSTORS	否	否

[0069] 其中，由于硬件特性决定，有25个事件会无条件地引发VMExit，而出于安全和功能考虑，将另外5个事件设置为在虚拟机和虚拟机监控器中都会触发VMExit，其它的事件则有些在虚拟机环境中不会触发VMExit，有些在虚拟机监控器环境下不会触发VMExit，还有一些在两种环境下都不会触发VMExit。另外，有些事件可以配置成部分触发VMExit，部分不触发，比如exception，RDMSR，WRMSR等，采用的方法是通过硬件提供的bitmap来配置相应的域，这样可以更加细粒度地配置触发VMExit的条件。这里需要注意的是，那些会无条件触发VMExit的情况都是属于出现的很少，甚至是在一些特殊的配置下才会产生，一般不会使用的特性，因此这些VMExit所造成的性能开销很小，而对于其它的配置，所遵循的原则如下：

[0070] 第一、如果某个实体本来不应该进行某些操作，则将其配置为会引起VMExit，比如在虚拟机中不应该读写VMCS，所以在虚拟机中VMREAD和VMWRITE都配置成VMExit，还有比如在虚拟机监控器中不应该有I/O的操作，所以在虚拟机监控器中I/O instruction被配置成VMExit；

[0071] 第二、对于一些必须要引发VMExit才会被通知的事件，则都配置成VMExit，比如interrupt/NMI windows，这些事件的发生如果不产生VMExit，则相关的软件无法被通知，

也不会有其它的机制来处理相应的事件，所以都被配置成VMExit；

[0072] 第三、对于一些功能性的操作，比如VMX-preemption timer，就是通过VMExit的产生来通知特权级软件时间片被用完了，所以也都被配置成VMExit；第四，其它的事件就被配置成不产生VMExit。

[0073] 关于虚拟机自处理事件模块，对于所有不产生下陷的事件，都由虚拟机自处理模块自行处理，在处理的过程中分为两个步骤，如果该事件可以在虚拟机直接解决，比如读写控制寄存器(control-register access)等，则在虚拟机自处理模块中自己进行，否则，对于那些需要经由虚拟机监控器完成的操作，比如修改EPT(EPT violation)，或者进行I/O操作等，则需要通过通信模块和虚拟机监控机进行通信，并调用相应的服务。

[0074] 关于虚拟机监控器服务模块，其为虚拟机提供必须的服务，包括为其分配页表，进行中断处理，部分异常处理，I/O处理等。虚拟机通过通信模块将请求发送给虚拟机监控器，而虚拟机监控器在处理完之后，再通过通信模块返回相应结果。

[0075] 图4展示了虚拟机和虚拟机监控器之间的通信模块。该通信模块可以进行虚拟机和虚拟机监控器上下文的高效切换，采用的方法是Intel处理器提供的一个虚拟化硬件扩展机制：EPTP Switching(扩展页表指针替换)，这是硬件提供的一个可以在虚拟机中运行的函数，该函数的功能是在不下陷到虚拟机监控器的情况下改变EPT Pointer的值。由于在嵌套虚拟化层模块的第三步中为这三个实体建立了三套EPT，并且将它们对应的EPTP指针组成了一个EPTP数组，并且将该数组的首地址填到数据结构VMCS中的一个特定的域EPTP_LIST_ADDR中。

[0076] 上下文切换的过程如下：虚拟机自处理模块通过跳板函数调用VMFUNC指令，并将eax寄存器设置为0，将ecx寄存器设置成安全门的EPTP对应的索引，则可以达到在不产生VMExit的情况下降当前的上下文切换到通安全门对应的内存中，类似的，安全门也可以通过设置寄存器，并调用VMFUNC指令，将上下文切换到虚拟机监控器的内存上下文中。反之，从虚拟机监控器返回虚拟机的途径也是用VMFUNC机制。需要注意的是，虽然虚拟机监控器为虚拟机提供服务，但是需要保证其不能窃取和篡改虚拟机的数据。为了保证这一点，需要安全门在虚拟机监控器返回时对结果进行检查和处理。比如当虚拟机监控器处理虚拟机的EPT violation的时候，它为虚拟机分配了一段空的内存，并填写好相应的EPT的条目，安全门在检查的过程中就需要将该内存页从虚拟机监控器的映射中去除，从而禁止其访问该页的内容，在保证了安全性的前提下大量减少了VMExit的数量，从而提高了系统的整体性能。

[0077] 本发明提出的一种高效的基于嵌套虚拟化的虚拟机安全保护框架，它通过嵌套虚拟化技术将虚拟机和虚拟化层进行有效的隔离，从而防止恶意的虚拟机监控器窃取和篡改虚拟机的数据，同时在实际运行过程中尽可能地避免由隔离机制所带来的虚拟机下陷，极大地减小了性能损失，使得虚拟机和虚拟机监控器能够进行安全且高效的切换。该架构能够被部署到现有的云计算平台中，满足用户对安全性、功能性和高效性的各方面需求。

[0078] 实施例

[0079] 一种高效的基于嵌套虚拟化的虚拟机安全保护框架的具体部署流程包括：

[0080] 步骤S1：嵌套虚拟化层模块对于控制层进行的配置；

[0081] 步骤S2：运行虚拟机自处理模块；

[0082] 步骤S3：运行虚拟机监控器服务模块；

[0083] 步骤S4:运行通信机制;

[0084] 步骤S5:对EPT violation进行处理;

[0085] 步骤S6:基于IOMMU设备对I/O进行处理;

[0086] 步骤S7:基于半虚拟化技术对I/O进行处理。

[0087] 其中步骤S5至步骤S7通过对三个虚拟化环境中最关键的场景的处理来具体阐述运行时各模块之间的行为。以下将通过具体实施示例来详细描述本发明。

[0088] 本发明的示例具体步骤如下:

[0089] 步骤S1包括:嵌套虚拟化层模块对控制层进行配置,配置的内容如下:第一,执行相关硬件指令,将虚拟机和虚拟机监控器运行在非特权级环境中;第二,创建嵌套虚拟化层在非特权级的代理,即通信模块,并且对其代码进行安全检查;第三,为非特权级环境中的三个实体(虚拟机,虚拟机监控器和通信模块)配置它们对应的EPT,保证虚拟机和虚拟机监控器的内存是相互隔离的,而且它们都无法访问通信模块的内存,而通信模块可以访问另外两个实体的内存;第四,按照表1的内容,为虚拟机和虚拟机监控器分别配置相关的VMCS中特定的域;第五,配置好嵌套虚拟化层中的下陷处理函数,对于那些在非特权级环境中会产生VMExit的事件进行相应的处理。

[0090] 步骤S2包括:在虚拟机的运行环境中,对于所有不产生VMExit的事件,都由进入虚拟机自处理模块的处理函数,由虚拟机对该事件自行处理。在处理的过程进行判断,如果该事件可以由虚拟机解决,则在虚拟机自处理模块中自行处理,否则,对于那些需要经由虚拟机监控器完成的操作,则通过步骤S4中的通信机制,和虚拟机监控机进行通信,并调用相应的服务。

[0091] 步骤S3包括:在虚拟机监控器的运行环境中,虚拟机监控器的服务模块为虚拟机提供必须的服务,包括为其分配页表,进行中断处理,部分异常处理,I/O处理等。虚拟机通过通行模块将请求发送给虚拟机监控器,而虚拟机监控器在处理完之后,再通过步骤S4中的通信机制,向虚拟机返回相应结果。

[0092] 步骤S4包括:在虚拟机自处理模块中,如果需要向虚拟机监控器请求服务,则进入通信模块在虚拟机端提供的跳板函数,该跳板函数通过VMFUNC指令,传入通信模块中的安全门对应的EPT指针索引,快速切换到安全门相对应的内存空间,并由安全门的跳板函数切换到虚拟机监控器服务模块的内存空间;虚拟机监控器服务模块处理完请求之后,进入通信模块在虚拟机监控器端提供的跳板函数,该跳板函数通过VMFUNC指令,传入安全门相对应的EPT指针索引,快速切换到相应的内存空间,安全门对返回的结果进行检查,防止虚拟机监控器对虚拟机数据的窃取和篡改,如果检查成功,则通过其跳板函数返回虚拟机的内存空间,完成整个通信流程。

[0093] 步骤S5包括:在虚拟机运行时如果发生EPT violation事件,如图5所示,由于嵌套虚拟化层在VMCS中配置了该事件在虚拟机中不发生VMExit,因此控制流会进入虚拟机自处理模块,自处理模块判断该事件需要请求虚拟机监控器进行处理,因此通过步骤S4中的通信机制将请求发送给虚拟机监控器,在经过安全门的时候,安全门会将相应的虚拟机的EPT页表页设置为虚拟机监控器可直接读写。虚拟机监控器根据请求中提供的信息,为该虚拟机分配相应的内存页,在虚拟机的EPT中填入相应的条目,并且通过步骤S4中的通信机制返回虚拟机,在经过安全门的时候,安全门会对虚拟机的页表项进行检查,并且将虚拟机的

EPT页表页设置为虚拟机监控器不可访问，同时，将新分配的页也设置成虚拟机监控器不可访问，从而防止虚拟机监控器恶意窃取和篡改虚拟机的内存数据。这里需要注意的是，安全门在设置虚拟机EPT页表页权限的时候需要先将其保护起来，防止虚拟机监控器在检查之后，设置权限之前对EPT页表页进行篡改，即所谓的TOCTOU(time-too-check to time-of-use)攻击，采用的方法包括硬件事务内存等机制。

[0094] 步骤S6包括：在虚拟机需要进行基于IOMMU设备的I/O事件的处理的时候，如图6所示，由于IOMMU设备可以将某个I/O设备直接分配给虚拟机。因此虚拟机可以直接对该I/O设备进行读写操作，不需要向虚拟机监控器请求服务。另外，I/O设备会产生大量的中断，由于我们在嵌套虚拟化中配置了在虚拟机运行时产生的中断不会引发VMExit，因此中断会直接进入虚拟机自处理模块，而虚拟机自处理模块判断当前中断是由直接分配给自己的I/O设备产生，可以由自己直接处理，因此也不需要向虚拟机监控器请求服务。最后，为了保证虚拟机监控器不能访问虚拟机磁盘上的数据，需要防止虚拟机监控器的I/O操作，即在嵌套虚拟化层配置虚拟机监控器的I/O操作会产生VMExit，从而对其进行拦截。

[0095] 步骤S7包括：在虚拟机需要进行基于半虚拟化技术的I/O事件的处理的时候，如图7所示，需要加入另外一个实体，即专门负责I/O设备驱动的虚拟机(driver VM)，并且在嵌套虚拟化层模块中对其EPT进行配置。初始化时，虚拟机和驱动虚拟机之间首先会建立一块共享内存，当发生I/O写操作时，虚拟机将数据写入共享内存，并且通过步骤S4中的通信机制(虚拟机->安全门->驱动虚拟机)通知驱动虚拟机，执行真正的I/O写操作。在发生I/O读操作的时候，驱动虚拟机先从I/O设备中读取数据，并将它们写入共享内存，然后通过步骤S4中的通信机制(驱动虚拟机->安全门->虚拟机)通知虚拟机，由虚拟机直接从共享内存中读取数据。这里需要注意的是，由于驱动虚拟机并不是可信的，所以I/O数据需要由虚拟机自己负责进行加解密和对完整性的保护，从而防止驱动虚拟机窃取和篡改I/O的数据。

[0096] 以上对本发明的具体实施例进行了描述。需要理解的是，本发明并不局限于上述特定实施方式，本领域技术人员可以在权利要求的范围内做出各种变化或修改，这并不影响本发明的实质内容。在不冲突的情况下，本申请的实施例和实施例中的特征可以任意相互组合。

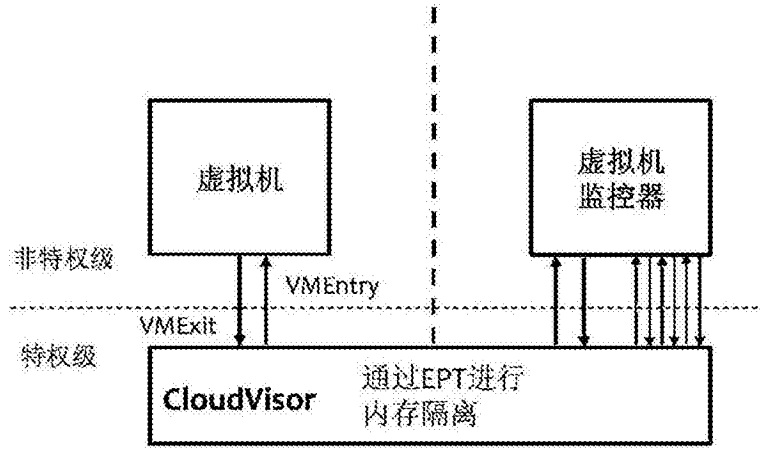

虚拟机
虚拟机
监控器
非特权级
VMEntry
VMExit
特权级
CloudVisor
通过EPT进行
内存隔离

图1

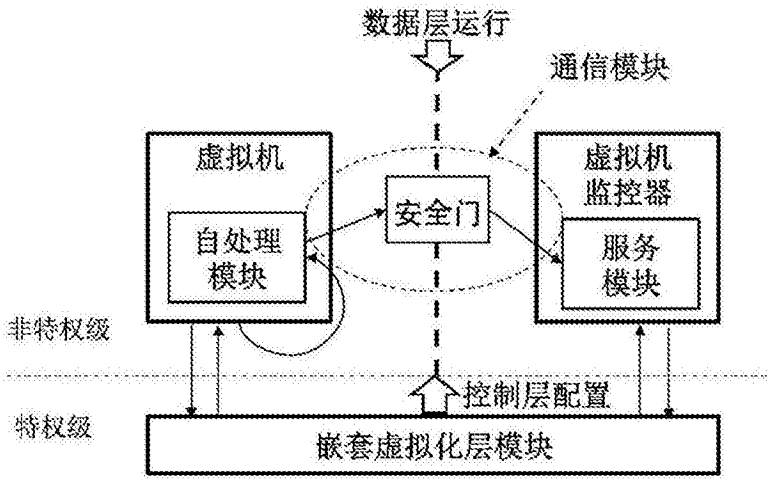

数据层运行
通信模块
虚拟机
自处理
模块
安全门
虚拟机
监控器
服务
模块
非特权级
控制层配置
特权级
嵌套虚拟化层模块

图2

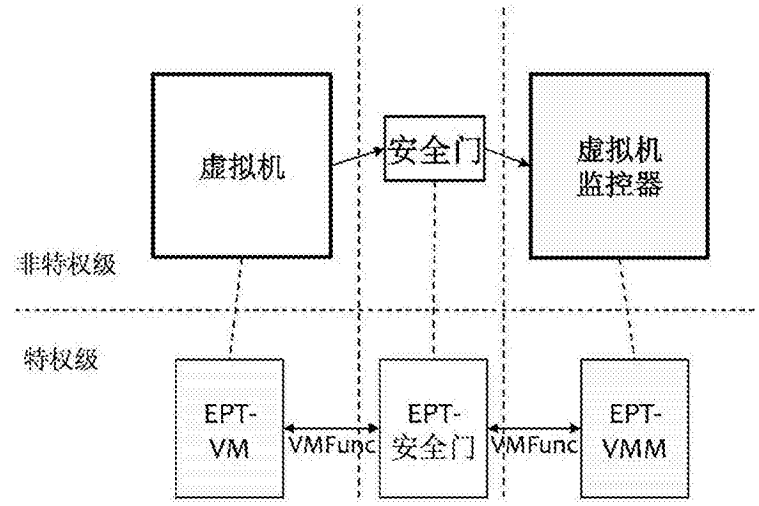

虚拟机
安全门
虚拟机
监控器
非特权级
特权级
EPT-
VM
VMFunc
EPT-
安全门
VMFunc
EPT-
VMM

图3

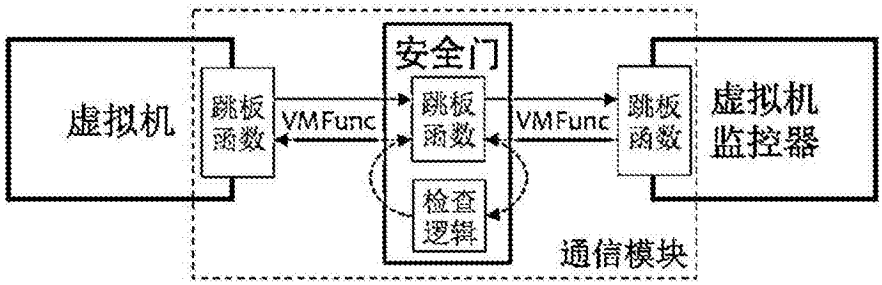

虚拟机
跳板函数
VMFunc
安全门
跳板函数
检查逻辑
VMFunc
跳板函数
虚拟机监控器
通信模块

图4

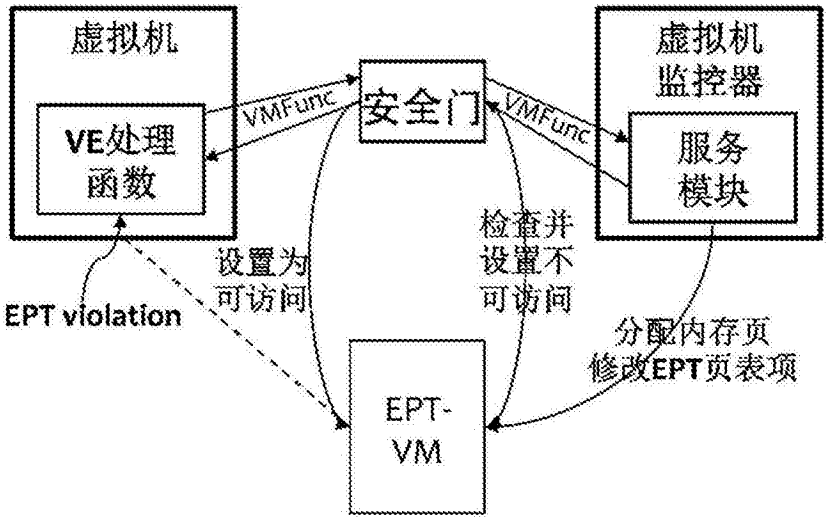

虚拟机
VE处理函数
VMFunc
安全门
VMFunc
虚拟机监控器
服务模块
设置为可访问
检查并设置不可访问
EPT violation
分配内存页修改EPT页表项
EPT-VM

图5

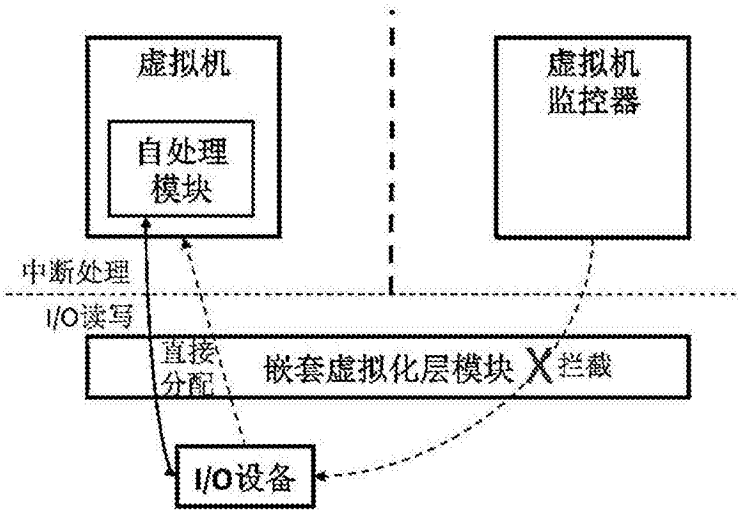

虚拟机
自处理模块
虚拟机监控器
中断处理
I/O读写
直接分配
嵌套虚拟化层模块
拦截
I/O设备

图6

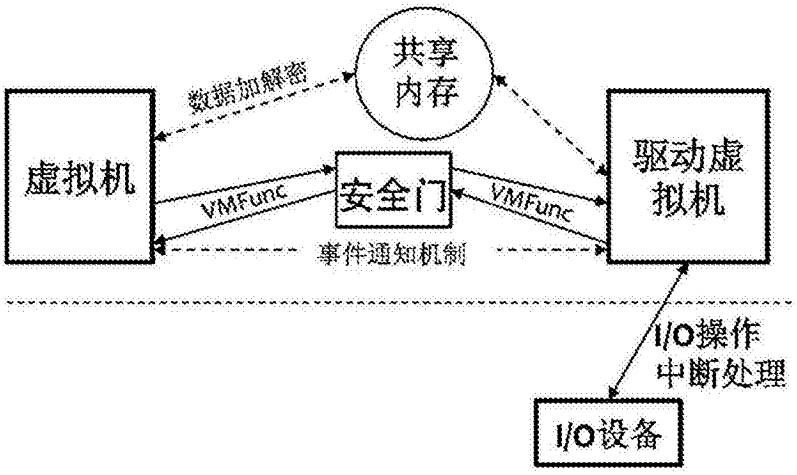

共享
内存
数据加解密
虚拟机
VMFunc
安全门
VMFunc
驱动虚
拟机
事件通知机制
I/O操作
中断处理
I/O设备

图7