



(12) 发明专利申请

(10) 申请公布号 CN 105184158 A

(43) 申请公布日 2015. 12. 23

(21) 申请号 201510509372. 1

(22) 申请日 2015. 08. 18

(71) 申请人 北京汉柏科技有限公司

地址 100085 北京市海淀区上地十街 1 号院
5 号楼 5 层 511 室

(72) 发明人 王智民

(74) 专利代理机构 北京中政联科专利代理事务
所（普通合伙） 11489

代理人 柴智敏

(51) Int. Cl.

G06F 21/55(2013. 01)

G06F 21/53(2013. 01)

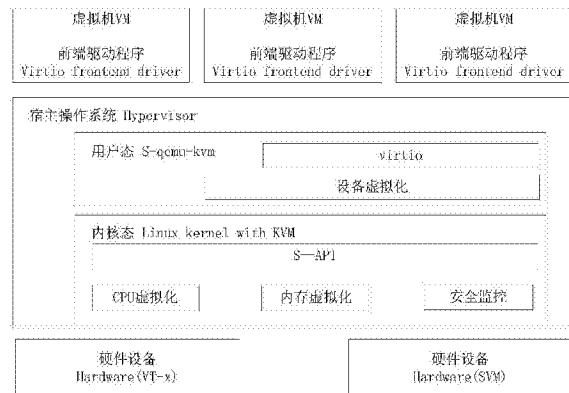
权利要求书2页 说明书8页 附图2页

(54) 发明名称

一种提升云计算操作系统安全性的方法

(57) 摘要

一种提升云计算操作系统安全性的方法，包括以下步骤：步骤 S1，屏蔽该操作系统中全部原始外部访问接口；步骤 S2，对操作系统中预先设定必要的外部访问接口进行重新封装，形成有效外部访问接口。本发明，可以简化对大型的云计算操作系统的安全性检查过程，节省人力、财力、物力，并且使得云操作系统具有更高的安全性。



1. 一种提升云计算操作系统安全性的方法,其特征在于,包括:
 - 步骤 S1,屏蔽该操作系统中全部原始外部访问接口;
 - 步骤 S2,对该操作系统中预先设定必要的外部访问接口进行重新封装,形成有效外部访问接口。
2. 根据权利要求 1 所述的方法,其特征在于,所述对该操作系统中预先设定必要的外部访问接口进行重新封装的步骤包括:
 - 封装操作系统中设置和控制全局性参数的系统调用指令;
 - 封装虚拟机中设置内存和创建虚拟处理器的虚机操作指令;
 - 封装虚拟机中控制虚拟处理器中寄存器的读写、中断、事件管理和内存管理的虚拟 CPU 操作指令;

所述虚拟 CPU 操作指令包括控制寄存器的指令、控制中断和事件管理的指令和控制内存管理的指令。
3. 根据权利要求 2 所述的方法,其特征在于,所述系统调用指令包括:
 - 创建虚拟机;查询当前虚拟机的外部接口版本;获得索引列表;检查扩展支持情况;运行虚拟机和用户态空间所共享内存区域容量。
4. 根据权利要求 2 所述的方法,其特征在于,所述虚机操作指令包括:
 - 为虚拟机创建虚拟处理器;根据结构体信息,运行虚拟机;创建虚拟可编程中断控制器,并将随后创建的虚拟处理器都关联到此可编程中断控制器;对虚拟可编程中断控制器发送中断信号;读取可编程中断控制器的中断标志信息;写入可编程中断控制器的中断标志信息;返回脏内存页的位图。
5. 根据权利要求 2 所述的方法,其特征在于,虚拟 CPU 操作指令中控制寄存器的指令包括:
 - 获取通用寄存器信息;设置通用寄存器信息;获取特殊寄存器信息;设置特殊寄存器信息;获取 MSR 寄存器信息;设置 MSR 寄存器信息;获取浮点寄存器信息;设置浮点寄存器信息;获取虚拟处理器的 xsave 寄存器信息;设置虚拟处理器的 xsave 寄存器信息;获取虚拟处理器的 xcr 寄存器信息;设置虚拟处理器的 xcr 寄存器信息。
6. 根据权利要求 2 所述的方法,其特征在于,虚拟 CPU 操作指令中控制中断和事件管理的指令包括:
 - 在虚拟处理器上产生中断;设置某个虚拟处理器的中断信号屏蔽掩码;获取虚拟处理器中被挂起待延时处理的事件;设置虚拟处理器的事件。
7. 根据权利要求 2 所述的方法,其特征在于,虚拟 CPU 操作指令中控制内存管理的指令包括:
 - 将虚拟处理器的物理地址翻译成 HPA;修改虚拟处理器的内存区域;初始化 TSS 内存区域;创建 EPT 页表。
8. 根据权利要求 1-7 任一项所述的方法,其特征在于,在步骤 S2 之后,还包括步骤 S3:对操作系统与外部程序的信息交互通道进行监控,若发现除了有效外部访问接口之外还有其他信息交互通道,则阻断该信息交互通道,并发出告警提示。
9. 根据权利要求 8 所述的方法,其特征在于,所述对操作系统与外部程序的信息交互通道进行监控的步骤,包括:

对操作系统与外部程序的交互信息进行筛选和甄别，根据预设的信息交互情况将交互信息判定为正常通信、可疑通信和危险通信；

若判定结果为正常通信，则按照正常通信状态处理；

若判定结果为可疑通信，则发出可疑信息告警提示；

若判定结果为危险通信，则阻断该信息的信息交互通道，并发出危险信息告警提示。

10. 根据权利要求 8 所述的方法，其特征在于，

所述操作系统包括宿主操作系统和客户操作系统；

所述对操作系统与外部程序的信息交互通道进行监控的步骤中，包括对宿主操作系统与外部程序的信息交互通道进行监控，以及对客户操作系统与外部程序的信息交互通道进行监控。

一种提升云计算操作系统安全性的方法

技术领域

[0001] 本发明涉及云计算技术领域，尤其涉及一种提升云计算操作系统安全性的方法。

背景技术

[0002] 云计算操作系统，又称云 OS、云计算操作系统、云计算中心操作系统，是以云计算、云存储技术作为支撑的操作系统，是云计算后台数据中心的整体管理运营系统，它是指构架于服务器、存储、网络等基础硬件资源和单机操作系统、中间件、数据库等基础软件之上的、管理海量的基础硬件、软件资源的云平台综合管理系统。

[0003] 操作系统按照是否公开源代码方式分类可分为开源操作系统和闭源操作系统。开源操作系统 (open source operating system)，即公开源代码的操作系统软件，遵循开源协议 (GNU) 进行使用、编译和再发布。在遵守 GNU 协议的前提下，任何人都可以免费使用，随意控制软件的运行方式。

[0004] 闭源操作系统 (Closed source operating system)，即不公开源代码的操作系统软件，它意味着将仅获得它们许可的计算机程序的一个二进制版本而没有这个程序的源代码，软件的翻译修改从技术方面几乎是不可能的。这个发展模型的源代码被看作这个公司的商业秘密，因此可能获得源代码接入的团体，例如学校，必须提前签订不泄漏协议。

[0005] 现有技术中，开源操作系统代表 :Linux、Symbian、Android、unix 派生系统等，闭源操作系统代表 :Windows、MacOS、iOS、WP 等。基于开源操作系统软件主要有 XEN 和 KVM，商业软件主要有 vmware 的 vSphere、Microsoft 的 Hyper-V。

[0006] 在需要高度信息安全性机关、事业单位以及企业中，每当引进一个操作系统，需要检查其源代码，以确保该操作系统的信息安全性，由于开源操作系统的源代码具有可见性、可修改性、漏洞查找便宜性等，因此，在需要高度信息安全性机关、单位、事业部门以及企业一般选择开源操作系统。

[0007] 现有的大型的云计算操作系统均由国外厂商开发，当引进的操作系统为云计算操作系统时，更需要严格检查其源代码的安全性，以防止国外别有居心的厂商利用云计算操作系统盗取我国国家机关、事业单位以及企业的机密数据，从而给我国国家机关、事业单位以及企业带来严重安全威胁。

[0008] 但由于大型的云计算操作系统的源代码数量相当庞大，按照传统安全性处理的方式对其源代码进行检查，不仅需要耗费大量的时间、人力、财力，而且在与硬件结合的情况下，即使检查时未发现安全隐患，在使用中，仍然会产生一些未知的安全隐患。因此，亟需一种更为安全的、效率更高的对云计算操作的安全性进行加固的方法。

发明内容

[0009] 本发明的目的是提供一种提升云计算操作系统安全性的方法，包括以下步骤：

[0010] 步骤 S1，屏蔽该操作系统中全部原始外部访问接口；

[0011] 步骤 S2，对该操作系统中预先设定必要的外部访问接口进行重新封装，形成有效

外部访问接口。

[0012] 其中,所述对该操作系统中预先设定必要的外部访问接口进行重新封装的步骤包括:

[0013] 封装操作系统中设置和控制全局性参数的系统调用指令;

[0014] 封装虚拟机中设置内存和创建虚拟处理器的虚机操作指令;

[0015] 封装虚拟机中控制虚拟处理器中寄存器的读写、中断、事件管理和内存管理的虚拟 CPU 操作指令;

[0016] 所述虚拟 CPU 操作指令包括控制寄存器的指令、控制中断和事件管理的指令和控制内存管理的指令。

[0017] 其中,所述系统调用指令包括:

[0018] 创建虚拟机;查询当前虚拟机的外部接口版本;获得索引列表;检查扩展支持情况;运行虚拟机和用户态空间所共享内存区域容量。

[0019] 其中,所述虚机操作指令包括:

[0020] 为虚拟机创建虚拟处理器;根据结构体信息,运行虚拟机;创建虚拟可编程中断控制器,并将随后创建的虚拟处理器都关联到此可编程中断控制器;对虚拟可编程中断控制器发送中断信号;读取可编程中断控制器的中断标志信息;写入可编程中断控制器的中断标志信息;返回脏内存页的位图。

[0021] 其中,虚拟 CPU 操作指令中控制寄存器的指令包括:

[0022] 获取通用寄存器信息;设置通用寄存器信息;获取特殊寄存器信息;设置特殊寄存器信息;获取 MSR 寄存器信息;设置 MSR 寄存器信息;获取浮点寄存器信息;设置浮点寄存器信息;获取虚拟处理器的 xsave 寄存器信息;设置虚拟处理器的 xsave 寄存器信息;获取虚拟处理器的 xcr 寄存器信息;设置虚拟处理器的 xcr 寄存器信息。

[0023] 其中,虚拟 CPU 操作指令中控制中断和事件管理的指令包括:

[0024] 在虚拟处理器上产生中断;设置某个虚拟处理器的中断信号屏蔽掩码;获取虚拟处理器中被挂起待延时处理的事件;设置虚拟处理器的事件。

[0025] 其中,虚拟 CPU 操作指令中控制内存管理的指令包括:

[0026] 将虚拟处理器的物理地址翻译成 HPA;修改虚拟处理器的内存区域;初始化 TSS 内存区域;创建 EPT 页表。

[0027] 其中,在步骤 S2 之后,还包括步骤 S3:对操作系统与外部程序的信息交互通道进行监控,若发现除了有效外部访问接口之外还有其他信息交互通道,则阻断该信息交互通道,并发出告警提示。

[0028] 其中,所述对操作系统与外部程序的信息交互通道进行监控的步骤,包括:

[0029] 对操作系统与外部程序的交互信息进行筛选和甄别,根据预设的信息交互情况将交互信息判定为正常通信、可疑通信和危险通信;

[0030] 若判定结果为正常通信,则按照正常通信状态处理;

[0031] 若判定结果为可疑通信,则发出可疑信息告警提示;

[0032] 若判定结果为危险通信,则阻断该信息的信息交互通道,并发出危险信息告警提示。

[0033] 其中,所述操作系统包括宿主操作系统和客户操作系统;

[0034] 所述对操作系统与外部程序的信息交互通道进行监控的步骤中,包括对宿主操作系统与外部程序的信息交互通道进行监控,以及对客户操作系统与外部程序的信息交互通道进行监控。

[0035] 本发明的有益效果为:简化对大型的云计算操作系统的安全性检查工作,节省人力、财力、物力,并且使得云操作系统具有更高的安全性。

附图说明

[0036] 图 1 为现有技术中 KVM 逻辑框架;

[0037] 图 2 为本发明中 KVM 逻辑框架;

[0038] 图 3 为本发明提升云计算操作系统安全性的方法的流程图;

[0039] 图 4 为本发明提升云计算操作系统安全性的方法中对信息交互通道进行监控的步骤流程图。

具体实施方式

[0040] 为使本发明的目的、技术方案和优点更加清楚明了,下面结合具体实施方式并参照附图,对本发明进一步详细说明。应该理解,这些描述只是示例性的,而并非要限制本发明的范围。此外,在以下说明中,省略了对公知结构和技术的描述,以避免不必要的混淆本发明的概念。

[0041] 术语解释

[0042] 虚拟化技术是云计算操作系统中的一项核心技术,所有的虚拟化都是两部分组成:虚拟机(VM)和宿主(HOST),虚拟机(VM)内运行客户操作系统(Guest OS),HOST 中运行宿主操作系统(Host OS)。

[0043] HOST——宿主,指物理上存在的计算机。

[0044] Host OS——宿主操作系统,指 HOST 上运行的操作系统。

[0045] VM(Virtual Machine)——虚拟机,逻辑上的计算机,指由Vmware模拟出来的一台虚拟的计算机。

[0046] Guest OS——客户操作系统,指运行在 VM(Virtual Machine)上的操作系统。

[0047] 例如:在一台安装了 Windows NT 的计算机上安装了Vmware,那么,HOST 指的是安装 Windows NT 的这台计算机,其 Host OS 为 Windows NT。VM 上运行的是 Linux,那么 Linux 即为 Guest OS。

[0048] Full-Virtualization——完全虚拟化。客户操作系统(Guest OS)运行在位于物理机器上的 hypervisor 之上,客户操作系统并不知道它已被虚拟化,并且不需要任何更改就可以工作。

[0049] Para-Virtualization——半虚拟化。客户操作系统(Guest OS)不仅知道它运行在 hypervisor 之上,还包括让客户操作系统更高效地过度到 hypervisor 的代码。

[0050] Hypervisor——一种运行在宿主(HOST)和虚拟机(VM)之间的中间软件,可允许多个操作系统和应用共享一套基础物理硬件,因此也可以看作是虚拟环境中的“元”操作系统,它可以协调访问服务器上的所有物理设备和虚拟机,也叫虚拟机监视器(Virtual Machine Monitor)。当服务器启动并执行 Hypervisor 时,它会给每一台虚拟机分配适量

的内存、CPU、网络和磁盘，并加载所有虚拟机的客户操作系统。hypervisor 可以捕获 CPU 指令，为指令访问硬件控制器和外设充当中介。在完全虚拟化的环境下，hypervisor 运行在裸硬件上，充当宿主操作系统，而由 hypervisor 管理的虚拟服务器运行客户端操作系统（guest OS）。

[0051] KVM(Kernel-based Virtual Machine)——基于 Linux 内核的虚拟机。KVM 的功能组件包括两部分：内核态（linux kernel with KVM）和用户态（qemu-kvm）。

[0052] 内核态（linux kernel with KVM）——负责模拟虚拟机的 CPU 运行、内存管理、设备管理等。

[0053] 用户态（qemu-kvm）——用于模拟虚拟机的 IO 设备接口以及用户态控制接口 API。

[0054] Virtio——KVM 虚拟环境下针对 I/O 虚拟化的最主要的通用框架。virtio 提供了一套有效、易维护、易开发、易扩展的中间层接口 API。Virtio 是 hypervisor 中位于设备之上的抽象层，是对 hypervisor 中的一组通用模拟设备的抽象。Virtio 的设置允许 hypervisor 导出一组通用的模拟设备，并通过一个通用的应用编程接口（API）让导出的模拟设备变得可用。

[0055] Device driver——设备驱动程序。

[0056] Frontend driver——前端驱动程序，指客户操作系统中的驱动程序。

[0057] Backend driver——后端驱动程序，指宿主操作系统中的驱动程序。

[0058] I/O trap——I/O 命令捕捉器，用于捕捉 I/O 命令。

[0059] Device emulation——仿真器

[0060] 本发明所述的提升云计算操作系统安全性的方法，主要适用于开源模式的云计算操作系统，但不限制于此。

[0061] 图 1 为现有技术中 KVM 逻辑框架。

[0062] 如图 1 所示，现有技术中的 KVM 模型中，每一个虚拟机都是一个由 Linux 调度程序管理的标准进程，在用户空间启动客户机操作系统。一个普通的 Linux 进程有两种运行模式：内核模式和用户模式。KVM 增加了第三种模式：客户模式（有自己的内核模式和用户模式）。

[0063] KVM 对 qemu-kvm 暴露一个字符文件 /dev/kvm，qemu-kvm 通过操作文件和 ioctl 方式与 KVM 交互，KVM 不提供针对 /dev/kvm 的 read 和 write 接口，只提供文件 open 和 close 的接口，其他的都是通过 ioctl 接口交互。

[0064] 图 2 为本发明中 KVM 逻辑框架。

[0065] 如图 2 所示，本发明中用 S-API 替换 /dev/kvm 以及 KVM 处理有关 I/O 请求处理接口；用 S-qemu-kvm 替换现有技术中的标准的 qemu-kvm；在 linux kernel with KVM 中设置了用于安全监控的模块。

[0066] 如图 3 所示，提升云计算操作系统安全性的方法包括以下步骤：

[0067] 步骤 S1，屏蔽该操作系统中全部原始外部访问接口。

[0068] 屏蔽具体含义为：关闭接口的外部访问功能，使其无法与外部连接或访问。原始外部访问接口具体含义为：引进的操作系统初始自带（本身固有）的外部访问接口 API，而不是后面生成的外部访问接口。操作系统的外部访问接口 API 是操作系统与外部进行信息交互的主要通道，当需要高度信息安全性的团体（如国家机关、事业单位以及企业单位）引进

的操作系统为大型的云计算操作系统时,按照以往的方式去检查引进的云计算操作系统的源代码耗费太多的时间和人力,因此,本发明不关心原有操作系统中是否有泄露信息的源代码,采用屏蔽该操作系统中原始外部访问接口 API 这一关闭原有通信通道的方式,从而从根源上避免了信息的泄露,且关闭原有信息通道的方式要远远比检查源代码的方式操作简单,因此能够有效的节约了时间和人力。

[0069] 步骤 S2,对该操作系统中预先设定必要的外部访问接口进行重新封装,形成有效外部访问接口。

[0070] 重新封装的含义为:解除原始外部访问接口的屏蔽状态,使其能够与外部连接或访问,并在原始外部访问接口的程序代码的基础上进行程序改进。

[0071] 没有重新封装的原始外部访问接口仍处于屏蔽状态。

[0072] 其中,预先设定必要的外部访问接口为:根据操作系统的所需的必要功能,屏蔽不必要的 API,只留下维持基本功能所必须的接口 API,从而实现精简该操作系统中的外部访问接口 API 的数量,即最小化 API 的集。

[0073] 预先设定必要的外部访问接口 API,例如包括:“计算虚拟化 API”、用户操作 API、网络流量访问 API 等。

[0074] 对该操作系统中预先设定必要的外部访问接口 API 进行重新封装,形成有效外部访问接口 S—API。具体为:

[0075] 封装操作系统中设置和控制全局性参数的系统调用指令(system 指令)。系统调用指令(system 指令)包括:创建虚拟机;查询当前虚拟机的外部接口版本;获得索引列表;检查扩展支持情况;运行虚拟机和用户态空间所共享内存区域容量。

[0076] 封装虚拟机中设置内存和创建虚拟处理器的虚机操作指令(VM 指令)。虚机操作指令(VM 指令)包括:为虚拟机创建虚拟处理器;根据结构体信息,运行虚拟机;创建虚拟可编程中断控制器,并将随后创建的虚拟处理器都关联到此可编程中断控制器;对虚拟可编程中断控制器发送中断信号;读取可编程中断控制器的中断标志信息;写入可编程中断控制器的中断标志信息;返回脏内存页的位图。

[0077] 封装虚拟机中控制虚拟处理器中寄存器的读写、中断、事件管理和内存管理的虚拟 CPU 操作指令(VCPU 指令)。虚拟 CPU 操作指令(VCPU 指令)包括控制寄存器的指令、控制中断和事件管理的指令和控制内存管理的指令。

[0078] 控制寄存器的指令包括:获取通用寄存器信息;设置通用寄存器信息;获取特殊寄存器信息;设置特殊寄存器信息;获取 MSR 寄存器信息;设置 MSR 寄存器信息;获取浮点寄存器信息;设置浮点寄存器信息;获取虚拟处理器的 xsave 寄存器信息;设置虚拟处理器的 xsave 寄存器信息;获取虚拟处理器的 xcr 寄存器信息;设置虚拟处理器的 xcr 寄存器信息。

[0079] 控制中断和事件管理的指令包括:在虚拟处理器上产生中断;设置某个虚拟处理器的中断信号屏蔽掩码;获取虚拟处理器中被挂起待延时处理的事件;设置虚拟处理器的事件。

[0080] 控制内存管理的指令包括:将虚拟处理器的物理地址翻译成 HPA;修改虚拟处理器的内存区域;初始化 TSS 内存区域;创建 EPT 页表。

[0081] 在本发明的一个实施例中,当开源软件为 KVM 时,system 指令具体为:

[0082]

KVM_CREATE_VM	创建 KVM 虚拟机
KVM_GET_API_VERSION	查询当前 KVM API 版本

[0083]

KVM_GET_MSR_INDEX_LIST	获得 MSR 索引列表
KVM_CHECK_EXTENSION	检查扩展支持情况
KVM_GET_VCPU_MMAP_SIZE	运行虚拟机和用户态共享内存区域的大小；

[0084] 在本发明的另一个实施例中，当开源软件为 KVM 时，VM 指令具体为：

[0085]

KVM_CREATE_VCPU	为虚拟机创建 VCPU
KVM_RUN	根据 kvm_run 结构体信息，运行 VM 虚拟机
KVM_CREATE_IRQCHIP	创建虚拟 APIC，且随后创建的 VCPU 都关联到此 APIC
KVM_IRQ_LINE	对某虚拟 APIC 发出中断信号
KVM_GET_IRQCHIP	读取 APIC 的中断标志信息
KVM_SET_IRQCHIP	写入 APIC 的中断标志信息
KVM_GET_DIRTY_LOG	返回脏内存页的位图

[0086] 在本发明的另一个实施例中，当开源软件为 KVM 时，VCPU 指令中控制寄存器的指令具体为：

[0087]

KVM_GET_REGS	获取通用寄存器信息
KVM_SET_REGS	设置通用寄存器信息
KVM_GET_SREGS	获取特殊寄存器信息
KVM_SET_SREGS	设置特殊寄存器信息
KVM_GET_MSRS	获取 MSR 寄存器信息
KVM_SET_MSRS	设置 MSR 寄存器信息
KVM_GET_FPU	获取浮点寄存器信息
KVM_SET_FPU	设置浮点寄存器信息
KVM_GET_XSAVE	获取 VCPU 的 xsave 寄存器信息
KVM_SET_XSAVE	设置 VCPU 的 xsave 寄存器信息
KVM_GET_XCRS	获取 VCPU 的 xcr 寄存器信息
KVM_SET_XCRS	设置 VCPU 的 xcr 寄存器信息

[0088] 在本发明的另一个实施例中，当开源软件为 KVM 时，VCPU 指令中控制中断和事件管理的指令具体为：

[0089]

KVM_INTERRUPT 在 VCPU	上产生中断(当 APIC 无效时)
KVM_SET_SIGNAL_MASK	设置某个 VCPU 的中断信号屏蔽掩码
KVM_GET_CPU_EVENTS	获取 VCPU 中被挂起待延时处理的事件，如中断、NMI 或异常
KVM_SET_CPU_EVENTS	设置 VCPU 的事件，如中断、NMI 或异常

[0090] 在本发明的一个实施例中，当开源软件为 KVM 时，VCPU 指令中控制内存管理的指令具体为：

[0091]

KVM_TRANSLATE	将 VCPU 的物理地址翻译成 HPA
KVM_SET_USER_MEMORY_REGION	修改 VCPU 的内存区域
KVM_SET_TSS_ADDR	初始化 TSS 内存区域(Intel 架构专用)
KVM_SET_IDENTITY_MAP_ADDR	创建 EPT 页表(Intel 架构专用)

[0092] VCPU 指令中控制内存管理的指令还包括 CPUID 的设置、调试接口等指令。在本发明的另一个实施例中，为了进一步提高云计算操作系统的安全性，在上述步骤 S3 之后，还包括以下步骤：对操作系统与外部程序的信息交互通道进行监控，若发现除了有效外部访

问接口之外还有其他信息交互通道，则阻断该信息交互通道，并发出告警提示。

[0093] S-API 是开源软件（例如 KVM）被动对外提供的外部访问接口 API，为了防范开源软件（例如 KVM）主动向外提供“接口”，比如利用某些不可知的硬件组件或特性主动向外发送信息，主动或被动触发向外发送加密信息等，需要对操作系统与外部程序的信息交互通道进行监控，一旦发现除了 S-API 交互通道之外的信息传输，则立即阻断并报警。

[0094] 操作系统包括宿主操作系统和客户操作系统，因此对操作系统与外部程序的信息交互通道进行监控的步骤中，既包括对宿主操作系统与外部程序的信息交互通道进行监控，又包括对客户操作系统与外部程序的信息交互通道进行监控。由于 Linux 宿主操作系统也存在较大的不安全性，因此，对操作系统与外部程序的信息交互通道进行监控的方式中，不仅仅是针对 KVM 的监控，也会针对 Linux 内核的其他模块进行外发通信的监控。当然对信息交互通道进行监控，会影响系统的性能，可以考虑在紧急时刻进行监控。

[0095] 图 4 为本发明提升云计算操作系统安全性的方法中对信息交互通道进行监控的步骤流程图。

[0096] 如图 4 所示，对操作系统与外部程序的信息交互通道进行监控的步骤，具体包括以下步骤：

[0097] 对操作系统与外部程序的交互信息进行筛选和甄别，根据预设的信息交互情况将交互信息判定为正常通信、可疑通信和危险通信；

[0098] 若判定结果为正常通信，则按照正常通信状态处理；

[0099] 若判定结果为可疑通信，则发出可疑信息告警提示；

[0100] 若判定结果为危险通信，则阻断该信息的信息交互通道，并发出危险信息告警提示。

[0101] 应当理解的是，本发明的上述具体实施方式仅仅用于示例性说明或解释本发明的原理，而不构成对本发明的限制。因此，在不偏离本发明的精神和范围的情况下所做的任何修改、等同替换、改进等，均应包含在本发明的保护范围之内。此外，本发明所附权利要求旨在涵盖落入所附权利要求范围和边界、或者这种范围和边界的等同形式内的全部变化和修改例。

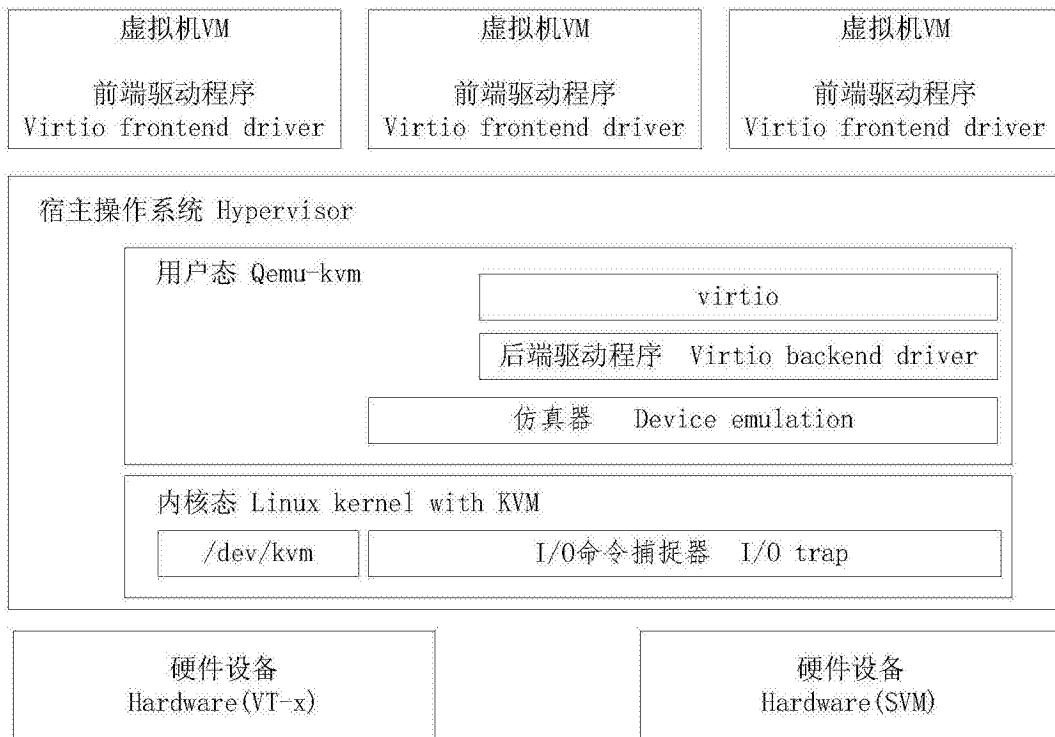


图 1

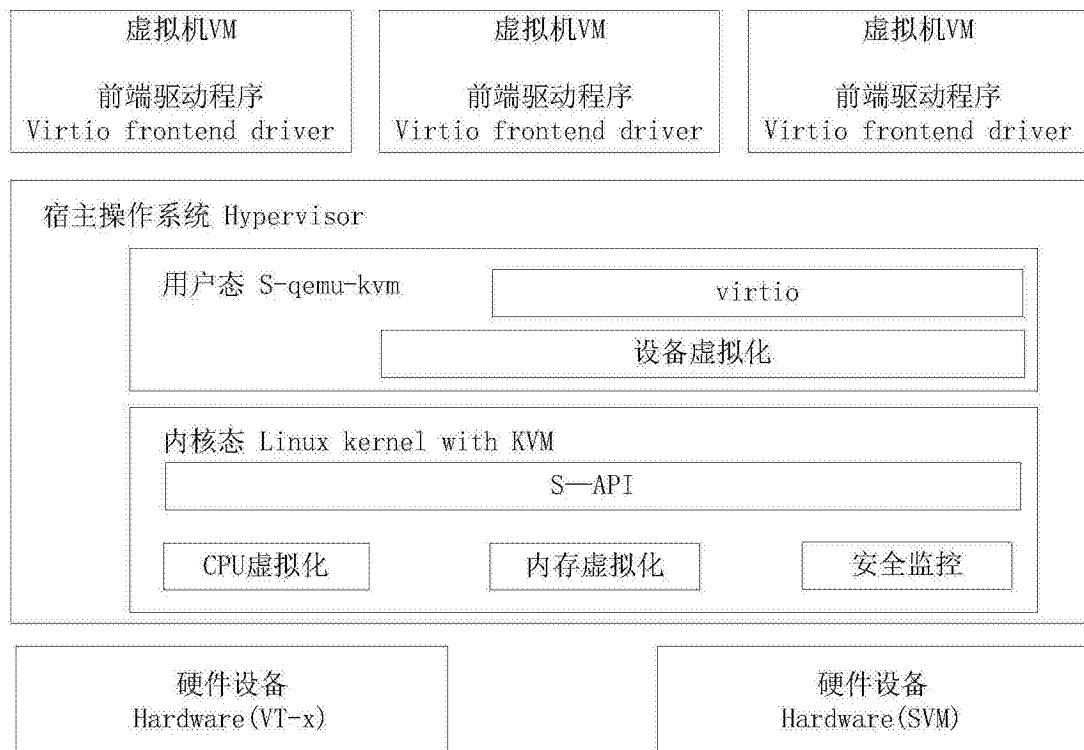


图 2

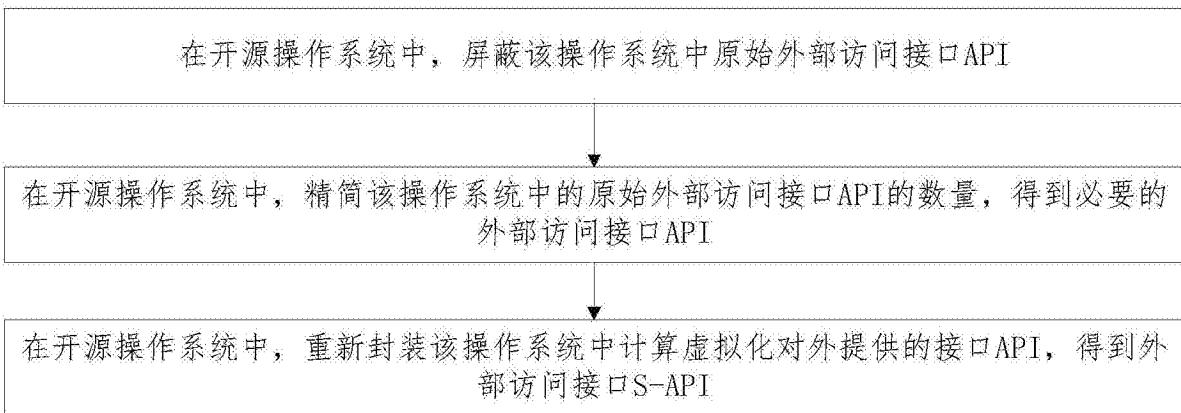


图 3

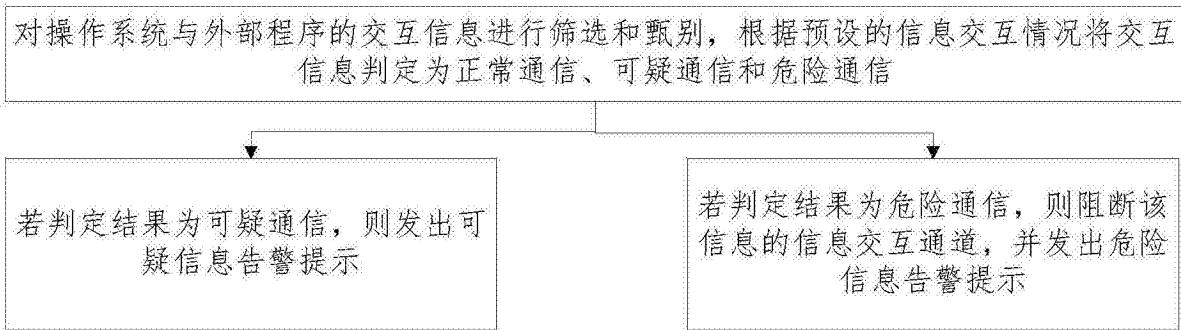


图 4