|  |
| --- |
| **信工所第五研究室 数据中心安全组** |
| **虚拟机安全套件系统** |
| **技术报告** |
|  |
|  |
| **当前版本： 1.0**  **更新时间： 2018-10-28**  **作 者： 刘文清** |
|  |

# 修订历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **修订者** | **参与者** | **时间** | **备注** |
| 1.0 | 刘文清 |  | 2019-01-09 | 创建文档 |
| 2.0 | 刘文清 | 张坤 | 2019-06-01 | 修改各个章节 |

目录

[修订历史 1](#_Toc2451)

[1 概述 4](#_Toc5283)

[1.1 术语与解释 4](#_Toc13517)

[1.2 研究背景 5](#_Toc23411)

[1.3 研究现状 7](#_Toc12419)

[1.3.1 国内发展现状 7](#_Toc31485)

[1.3.2 国外发展现状 7](#_Toc4220)

[1.4 研究问题（动机） 7](#_Toc15208)

[2 可行性分析 9](#_Toc22999)

[2.1 研究目标 9](#_Toc15520)

[2.2 需求分析 9](#_Toc18162)

[2.2.1运行环境  9](#_Toc2342)

[2.2.2功能需求 10](#_Toc29564)

[2.2.3性能需求  10](#_Toc25689)

[2.2.4运行需求  11](#_Toc31259)

[2.3 可行性分析 12](#_Toc22524)

[3 总体技术 13](#_Toc6227)

[3.1 威胁模型和假设 13](#_Toc25356)

[3.1.1 威胁模型 13](#_Toc21137)

[3.1.2 假设 15](#_Toc31779)

[3.2 总体设计 15](#_Toc8572)

[3.3 系统架构 16](#_Toc27633)

[3.4 系统流程 18](#_Toc15111)

[3.5 技术偏离表 20](#_Toc26777)

[4 关键技术1：基于同层隔离机制的地址空间隔离 20](#_Toc23764)

[4.1 基本原理 20](#_Toc24828)

[4.2 技术点1 21](#_Toc27367)

[4.3 技术点2 21](#_Toc7980)

[4.4 技术点3 23](#_Toc3260)

[4.5 技术评价 24](#_Toc25501)

[4.5.1 技术特色 24](#_Toc1082)

[4.5.2 创新性证明 25](#_Toc26325)

[4.5.3 成果说明 25](#_Toc5405)

[5 关键技术2：监控VM与HOS的交互关键数据 26](#_Toc23978)

[5.1 基本原理 26](#_Toc18633)

[5.2 技术点1 26](#_Toc15432)

[5.3 技术点2 27](#_Toc31145)

[5.4 技术评价 27](#_Toc5786)

[5.4.1 技术特色 27](#_Toc10881)

[5.4.2 创新性证明 28](#_Toc12489)

[5.4.3 成果说明 28](#_Toc4451)

[6 关键技术3：动态内存标记与跟踪 28](#_Toc10483)

[6.1 基本原理 28](#_Toc15132)

[6.2 技术点1 29](#_Toc11073)

[6.3 技术点2 29](#_Toc4961)

[6.4 技术评价 30](#_Toc27653)

[6.4.1 技术特色 30](#_Toc16126)

[6.4.2 创新性证明 31](#_Toc16273)

[6.4.3 成果说明 31](#_Toc24784)

[7 原型系统评估 31](#_Toc9689)

[7.1 安全性测试 31](#_Toc26461)

[7.1.1 测试环境与配置 32](#_Toc4134)

[7.1.2 测试结果分析 33](#_Toc12346)

[7.2 性能测试 34](#_Toc6501)

[7.2.1 测试环境与配置 34](#_Toc30629)

[7.2.2 测试结果分析 35](#_Toc28110)

[8 环境约束 35](#_Toc32260)

[8.1 开发环境 35](#_Toc17665)

[8.2 部署环境 36](#_Toc23067)

[9 后续说明 36](#_Toc10698)

[9.1 后续工作 36](#_Toc26171)

[9.2 其他事项 36](#_Toc25887)

[10 参考文献 37](#_Toc26560)

# 概述

概述分为背景、与自己研究内容相关的国内外研究现状、与自己系统相关的概念介绍（下面的1.3是别人书写的比较好的样板，大家可以参考多详细写写自己的部分）

## 术语与解释

* Hypervisor

又称虚拟机器监视器（英语：Virtual Machine Monitor，缩写为 VMM）。虚拟化就是通过某种方式隐藏底层物理硬件的过程，从而让多个虚拟机可以透明地使用和共享底层物理硬件。这种架构的另一个更常见的名称是平台虚拟化。

* KVM

一种基于硬件虚拟化技术的虚拟机监控器。

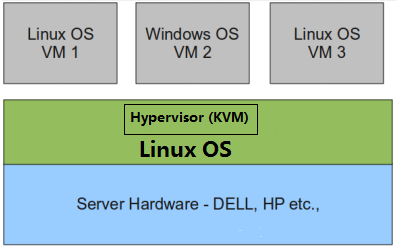


图1. KVM架构图

* 虚拟机上下文切换

虚拟机监控器为上层虚拟机提供资源分配和管理技术，每一个物理核每次只能运行虚拟机或者Hypervisor，所以在一个物理核上需要进行系统切换，在切换的过程中存在寄存器等系统的关键数据结构的内容变化，以便在下次切换时，能够正常进入系统。这些关键数据结构被称为上下文，这些数据被存放在VMCS结构体中。

虚拟机上下文切换主要实现虚拟机与宿主机的交互，该交互过程包含两个过程，即虚拟机进入（VM ENTRY）和虚拟机退出（VM EXIT）。虚拟机退出是因为虚拟机运行时，遇到一些敏感指令和特权指令无法处理时，需要退出到Hypervisor，让Hypervisor执行这些指令。那么这个过程叫做虚拟机退出，在此期间会进行一些交互，将虚拟机的寄存器信息和一些重要的状态信息写入到VMCS结构体中，同时读取VMCS中有关宿主机的寄存器等信息，转到宿主机执行状态。在这个过程中，会根据退出的不同原因处理退出事件。虚拟机进入则是相反的过程，将宿主机的寄存器信息和一些重要的状态信息写入到VMCS结构体中，同时读取VMCS中有关虚拟机的寄存器等信息，转到虚拟机执行状态。

* MMU

MMU是Memory Management Unit的缩写，中文名是内存管理单元，它是中央处理器（CPU）中用来管理虚拟存储器、物理存储器的控制线路，同时也负责虚拟地址映射为物理地址，以及提供硬件机制的内存访问授权，多用户多进程操作系统。主要有2类功能，1）将线性地址映射为物理地址，2）提供硬件机制的内存访问授权。针对第一个功能，现代的多用户[多进程操作系统](https://baike.baidu.com/item/%E5%A4%9A%E8%BF%9B%E7%A8%8B%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F" \t "https://baike.baidu.com/item/MMU/_blank)，需要MMU，才能达到每个用户进程都拥有自己独立的[地址空间](https://baike.baidu.com/item/%E5%9C%B0%E5%9D%80%E7%A9%BA%E9%97%B4" \t "https://baike.baidu.com/item/MMU/_blank)的目标。针对第二个功能，[微处理器](https://baike.baidu.com/item/%E5%BE%AE%E5%A4%84%E7%90%86%E5%99%A8" \t "https://baike.baidu.com/item/MMU/_blank)一直带有片上[存储器管理](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8%E7%AE%A1%E7%90%86" \t "https://baike.baidu.com/item/MMU/_blank)单元（MMU），MMU能使单个软件线程工作于硬件保护[地址空间](https://baike.baidu.com/item/%E5%9C%B0%E5%9D%80%E7%A9%BA%E9%97%B4" \t "https://baike.baidu.com/item/MMU/_blank)。

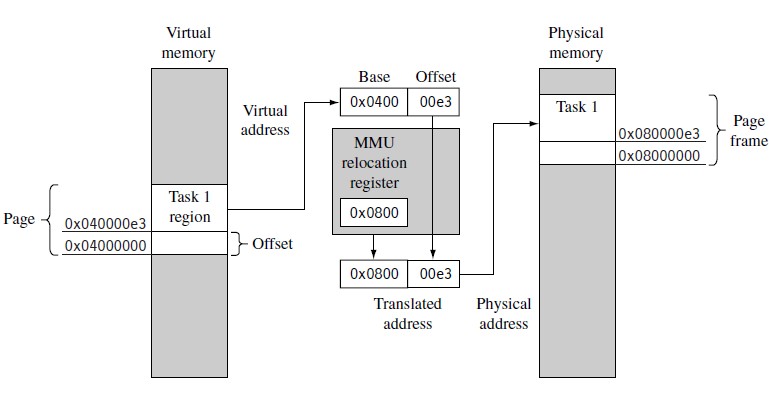


图2. MMU工作机制图

* DEP

数据执行保护技术（Data Execution Prevention，以下简称 DEP）是 winxp sp2 和 2003 sp1 新增的一项功能。DEP 简单地说可以看作操作系统底层的安全防护机制，本身不具备对病毒检测的功能。但如果有利用漏洞溢出包括本地溢出的病毒发作时，DEP 可以进行防护作用。其功能是禁止在标记为数据存储的内存区域中执行代码。此功能也称作“不执行”和“执行保护”。当尝试运行标记的数据页中的代码时，就会立即发生异常并禁止执行代码。这可以防止攻击者使用代码致使数据缓冲区溢出，然后执行该代码的攻击。

* SMEP

SMEP（supervisor mode execution prevention，防止超级用户执行）机制。SMEP机制可以阻止物理内存页被特权模式下的指令读取。当CR4寄存器中SMEP=1时，执行在特权级别下的软件不可以访问用户空间的线性地址。即内核程序不能跳转到用户态执行代码。当SMEP被设置，在用户空间的页表的虚拟地址并没有supervisor标志，当跳转到用户态时，会触发异常。

## 研究背景

把与自己研究内容相关的背景交代清楚。

2017 年数据泄露，网络攻击，随着物联网设备的激增，网络攻击目标泛化，并成指数级增加。用户的隐私数据泄露问题是特别严重的，数据安全问题已经发展成为一个全球性问题……信息泄露的方式从员工内部泄露到外部的黑客攻击。

云计算技术利用了一些虚拟化技术、资源动态均衡分配技术等，来给上层的多租户提供资源。这些租户使用相同的物理资源，即底层的硬件资源，这些资源是通过虚拟机监控器（VMM、VM machine monitor、Hypervisor）进行统一的分配和管理。其中虚拟化（Virtualization）是一种资源管理技术，是将计算机的各种实体资源，如服务器、网络、内存及存储等，予以抽象、转换后呈现出来，打破实体结构间的不可切割的障碍，使用户可以比原本的组态更好的方式来应用这些资源。这些资源的新虚拟部份是不受现有资源的架设方式，地域或物理组态所限制。

虚拟化技术已经产生了巨大的影响，并在云计算中扮演着至关重要的角色。云计算平台上的多租户共享物理资源，然而物理资源是由底层的Hypervisor 进行管理的。由于Hypervisor被授予最高权限，攻击者危害Hypervisor 可能会危及整个云计算基础设施，并危及云中的任何数据。

Hypervisor存在这样的弱点：1）当Hypervisor的代码量增大时，其存在的漏洞也越多，攻击者的攻击面也越广。2）Hypervisor和VM 之间需要频繁交互，一旦Hypervisor被攻击，则VM也会受到影响。

* 代码量增大，攻击面越广

虚拟化技术发展很快，越来越多的功能逐渐添加到Hypervisor 中，其代码量也逐渐增大。到目前为止，内核2.6.36.1 中XEN 和KVM 的代码量分别达到30 万行[1]和33.6万行[2]，当然，越来越多的代码量，也就意味着它会存在着许多有漏洞的地方，更容易被攻击的地方。

根据CVE 漏洞网站相关的数据，从2004 年到现在，有357 个和XEN 相关的漏洞，130 个和KVM 相关的漏洞[3]，比如，CVE-2018-1087[4]是个高危漏洞，攻击者可以利用它进行提权来攻击Hypervisor，从而对云平台上的租户进行攻击。因为云平台上的多租户共享物理资源，其中一种主要的资源就是物理内存，当多租户中某一租户被攻击，那么很可能发生跨域攻击。

图3. 2004-2017年KVM相关漏洞

* Hypervisor和VM 之间频繁交互

Hypervisor管理着所有VM的运行和物理资源分配。I/O设备的访问、内存资源的分配等。由于VM运行的过程中会使用到一些特权指令，但是这些指令不能被模拟，必须陷入到Hypervisor中进行处理，在这个过程中它们需要大量的交互。交互的过程很容易被攻击。

从上述的相关信息中可以得到，所以从Hypervisor 的角度对虚拟机进行保护是至关重要的。一个是控制Hypervisor的代码量，尽量不增加其代码量，另一方面，对Hypervisor和VM之间的交互进行监控保护。并且可以直接隔离保护VM的物理内存资源，从而达到最终的目标——减少来自恶意的Hypervisor 攻击，对VM进行保护。

## 研究问题

虚拟化技术发展很快，越来越多的功能逐渐添加到Hypervisor 中，其代码量也逐渐增大。到目前为止，内核2.6.36.1 中XEN 和KVM 的代码量分别达到30万行和33600行，当然，越来越多的代码量，也就意味着它会存在着许多有漏洞的地方，更容易被攻击的地方。根据CVE 漏洞网站相关的数据，从2004 年到现在，有357个和XEN 相关的漏洞，180个和KVM 相关的漏洞，比如，CVE-2009-2287在提权后，用于加载恶意的EPT页表。CVE-2018-1087是个高危漏洞，攻击者可以利用它进行提权来攻击Hypervisor，从而对云平台上的租户进行攻击。因为云平台上的多租户共享物理资源，其中一种主要的资源就是物理内存，当多租户中某一租户被攻击，那么很可能发生跨域攻击。

针对此，提出了虚拟机隔离与监控技术这一研究内容，主要是在KVM中添加虚拟机安全套件的方式完成，用于对虚拟机的运行时安全进行监控以及隔离虚拟机内存，尽可能减弱虚拟机跨域攻击造成的信息泄露。该方法是基于软件实现的，相比基于硬件方法实现的虚拟机安全防护[7][9][13][14][15]，其优点在于可移植性强，适用的平台广；相比于基于软件的大规模修改Hypervisor的方法[6][8][10],本方法对系统的修改小，适用于商业平台，作为内核模块可加载。

该系统的组成分为传统的Hypervisor和虚拟机安全套件，以及切换门。

切换门，用于两个环境进行切换，能够保障安全性，环境切换过程不可绕过，不可伪造，不可中断。

虚拟机安全套件，由安全执行环境，虚拟机映射安全监控，虚拟机安全隔离组成。用于提供对虚拟机运行时状态信息的隔离保护、对虚拟机内存高强度隔离保护，从而阻止用户隐私泄露问题。

## 研究现状

国内外研究现状一定要写与自己研究内容相关的，尽量详细清楚。以总结归纳的方式写

### 1.4.1 国内发展现状

针对在不可信的执行环境中提供对虚拟机的保护。国内各研究学者提出了各自的保护策略。复旦大学提出CloudVisor和Tinychecker，Cloudvisor利用嵌套虚拟化[10][11]将策略从Hypervisor 剥离，透明的对虚拟机进行保护。Tinycheeker 利用嵌套虚拟化技术透明地检查和恢复Hypervsior 的故障，利用嵌套虚拟化提供的隔离性可以实现安全的IDS 和蜜罐。比如说，V-Met 将基于VMI 的IDS 隔离于虚拟化系统，避免不安全虚拟化系统对IDS 的影响。上海交通大学并行与系统安全实验室提出了NeXen，主要通过将xen进行功能分割，以最小特权的方式将主要功能保留，每个VM拥有一个xen片段，它们公共的功能是由共享服务域进行处理，同时采用了嵌套虚拟化的技术布置了一个安全的监控器，用来监控内存管理和特权指令操作，针对DOS攻击进行防御。西安交通大学提出了Secpod，在x86平台上创建安全的执行环境，性能开销较嵌套虚拟化和微内核小。

### 1.4.2 国外发展现状

普林斯顿大学提出了NoHype的方案，利用设备的虚拟化特性通过资源预分配的方式实现资源的物理隔离，进行了彻底的隔离；北卡罗纳州立大学提出了对Hypervisor进行分割的技术，从逻辑上对Hypervisor的代码进行分割，保证每个虚拟机对应一份可运行的Hypervisor。韩国科技大学提出H-SVM的方法，利用CPU的硬件隔离特性，将Hypervisor中相应的功能替换为自身提供的微代码，从而限制Hypervisor对虚拟机内存资源的操作。Vmware公司提出了虚拟化平台 vSphere 构建云计算基础架构，提供最高级别的可用性和响应能力。虚拟化平台 vSphere 使用户能够自信地运行关键业务应用程序，更快地对其业务作出响应。

针对国内外学者提出的研究成果进行详细分析，创建可信执行环境和VM的防护两个部分。可信执行环境包含同层地址空间隔离技术实现隔离空间、定制硬件实现隔离空间等。对VM的防护主要是重组Hypervisor，实现不同组件和功能。

### 1.4.3 同层隔离技术

**SKEE**

为了提供对内核监控和保护，SKEE创建了与内核同级别的可信执行环境TEE，基于ARM平台，其主要目标是允许对内核进行安全的监视和保护，而不需要高级特权软件和虚拟化扩展的积极参与。

SKEE提供了一套保证隔离的新技术。它创建了一个内核无法访问的受保护地址空间，内核和独立环境共享相同的特权级别时。SKEE通过阻止内核管理自己的内存转换表来实现隔离技术。因此，内核被迫切换到SKEE来修改系统的内存布局。反过来，SKEE会验证来自内核的修改请求并且不会影响受保护地址空间的隔离。从操作系统内核到SKEE的切换只通过一个控制良好的开关门。这个开关门经过精心设计，因此它的执行顺序是原子的和确定性的。这些属性的结合保证了潜在的非可信内核不能利用交换序列来破坏隔离。如果内核试图违反这些属性，它只会导致系统失败，而不会暴露受保护的地址空间。

SKEE专门控制整个操作系统内存的访问权限。因此，它可以防止试图将未验证的代码注入内核的攻击。此外，它还可以很容易地扩展到拦截其他系统事件，以支持各种入侵检测和完整性验证工具。

该框架的优点在于基于ARM平台实现对内核的安全防护。

* 使用同层隔离思想，在同一份页表上创建不同的地址空间，基本实现如下图。
* 能够阻止原系统内核对页表和MMU的特权访问。
* 在实现地址空间切换时保证隔离空间的安全性。

### 1.4.4 硬件隔离技术

**H-SVM**

随着对云计算的需求不断增加，保护客户虚拟机（VM）免受恶意攻击者的攻击已成为提供安全服务的关键。当前的基于软件虚拟化的云安全模型依赖于软件管理程序及其具有根权限的值得信赖的管理员的无懈可击性。但是，如果将管理程序与远程攻击或根权限相冲突，则攻击者可以完全访问来宾虚拟机的内存和上下文。本文提出了一种基于硬件的方法来保护来宾虚拟机，即使在不受信任的管理程序下。通过这种机制，安全硬件提供了内存隔离，这比软件管理程序的脆弱性要小得多。该机制以较小的额外硬件成本扩展了当前基于嵌套分页的内存虚拟化硬件支持。虚拟机监控程序仍然可以灵活地将物理内存页分配给虚拟机，以实现高效的资源管理。除了安全虚拟化的系统设计外，本文还提出了一个使用系统管理模式的原型实现。虽然目前的系统管理模式不适用于安全功能，因而限制了性能和完整的保护，但原型实现证明了所提出设计的可行性。

### 1.4.5 重组系统Hypervisor

**NeXen**

该框架由上海交大并行与系统安全实验室提出。

Hypervisor容易受到攻击。不幸的是，高效强化的Hypervisor具有挑战性，因为它们缺乏特权安全监视和分解策略。在这项工作中，NeXen系统地分析了Xen Security Advisories中的191个Xen虚拟机监控程序漏洞，发现大多数（144）位于核心虚拟机监控程序中，而不是dom0。然后，使用分析将Xen（称为nexen）分解为一个安全监视器、一个共享服务域和每个VM Xen切片，这些切片由一个最低特权的沙盒框架隔离。使用嵌套内核架构实现nexen，在xen地址空间内有效地嵌套自己，并通过为任意多个保护域添加服务以及动态分配器、数据隔离和跨域控制流完整性来扩展嵌套内核设计。其效果是，nexen将基于虚拟机的管理程序限制在单个Xen虚拟机实例上，阻止已知Xen漏洞的74%（107/144），并强制Xen代码完整性（防御所有代码注入危害），同时观察可忽略的开销（平均1.2%）。总的来说，Nexen的独特定位是以最低的性能和实现成本提供管理程序强化的基本需求。

## 研究目标

最终在技术上、功能上或效果上的目标。

该虚拟机安全套件系统是一套附加在传统原有Hypervisor上的组件，该系统需要达到在非可信虚拟化层执行环境下尽可能保证上层虚拟机运行的安全性的目标，抵御一定的黑客内存越权访问攻击，保护虚拟机的运行时状态信息。在设计与实现的过程中，该软件的安全性设计要完备，同时开发实现的虚拟化安全套件需达到简洁、易于使用的目的，所以该安全套件应该符合以下的要求：

* 功能需求

根据用户的需求，设计虚拟化安全套件的功能，功能要达到相对的完善性，能覆盖用户的所有需求。达到虚拟机隔离的目的，以及能够抵御用户需求相关领域的黑客攻击。

* 性能需求

虚拟化安全套件在主要的3个功能（内存资源隔离，上下文切换，虚拟机安全映射）上的性能开销达到用户的要求（平均小于10%）。

* 出错处理需求

虚拟化安全套件的需求设计尽量趋于全面，根据用户的需求，指定合适的设计和测试方案，测试要全面，测试用例要合适和全面，发现尽可能多的软件错误和缺陷，并加以纠正，从而减少后期维护所付出的成本。

详细的需求如下小节介绍，主要是在运行环境、功能、性能、系统运行方面。

# 可行性分析

该节主要是对系统进行可行性和需求分析，写出开发这块系统的必要性。

可行性分析主要是研究该软件是否值得开发，能否解决研究问题中所提出的要求，通过对功能和性能需求上联合当前的技术手段进行分析，从而确定该方案的技术可行性。

## 需求分析

为解决上述研究问题需要在功能、性能、运行环境、部署等方面的需求。

### 2.1.1运行环境

* 硬件环境

系统的运行依赖于国产硬件基础环境，主要针对64位的x86架构平台。用于构建涉密云的计算节点，计算节点的CPU需要支持虚拟化，支持多台虚拟机运行在同一台服务器上，内存和硬盘要适量大。

* 软件环境

操作系统平台内核是linux4.4.0版本。虚拟机安全套件的运行需要完整的虚拟化系统，需要Hypervisor（KVM）、Host OS、Guest OS、libvirt、qemu。

KVM Hypervisor：KVM Hypervisor用于对物理资源进行虚拟，为VM的运行提供CPU、内存和I/O设备支持。对于x86平台，包含KVM的Linux版本要求4.4.0。

Libvirt：虚拟化API，在涉密云平台中结合云平台管理软件提供虚拟资源管理、VM调度和管理等，版本要求在3.0.0以上。

Qemu：Qemu用于和KVM结合提供用户层的接口，同时提供I/O设备的虚拟化功能，模拟VM的I/O操作，版本要求在2.6以上。

GOS：运行于虚拟化平台中的客户虚拟机，用于运行客户的应用负载，GOS要求是Linux发行版，Linux内核版本在2.6以上。

### 2.1.2功能需求

* 地址空间隔离

创建一个在非可信虚拟化层环境下的相对安全的隔离的地址空间，该空间可以为后续的两个功能提供执行环境。地址隔离空间实现的目的是，当虚拟化层不安全，受到攻击者攻击的时候，另外的2类功能可以安全地运行在隔离的地址空间中。同时提供对系统中的关键数据的保护。

* 虚拟机与Hypervisor的安全切换

Hypervisor在虚拟化底层为上层的虚拟机提供资源管理和资源分配的功能，所以Hypervisor与上层的虚拟机需要大量的交互过程，并且一个物理核每次只能运行一个系统，依据分时的方法，Hypervisor和虚拟机会在不同时间段使用物理核，所以它们之间需要一个安全的切换，防止在切换的过程中受到恶意攻击，避免不必要的危害。

* 虚拟机映射监控

上层虚拟机需要运行，其地址映射过程需要Hypervisor的参与。因为Hypervisor在虚拟化底层管理资源的分配，其中就包括内存资源的分配，那么地址映射的过程中会有Hypervisor的参与，最终上层虚拟机的虚拟地址会映射到真实的物理内存上。

* 功能需求

将虚拟机安全套件主要分为3个部分，分别提供对应的功能，保障其系统的安全性。三个功能分别是地址空间隔离、虚拟机映射监控、虚拟机与Hypervisor的安全切换。

根据用户的需求，设计虚拟化安全套件的功能，功能要达到相对的完善性，能覆盖用户的所有需求。达到虚拟机隔离的目的，以及能够抵御用户需求相关领域的黑客攻击。

### 2.1.3性能需求

1. **性能损耗**

  针对地址空间隔离、虚拟机映射监控和虚拟机上下文切换的性能损耗都要小于10%。

1. **时间特性**

* 环境切换

对于隔离的新地址空间和原先的Hypervisor之间切换的时间要相对较小。

* 虚拟机与Hypervisor的安全切换

虚拟机在处理敏感指令的时候，会退出到Hypervisor中处理，从非root模式到root模式的过程被称为虚拟机退出，它的反过程被称为虚拟机进入，需要确保切换的过程时间开销相对较小。

1. **适应性**

* 操作方式

通过加载内核模块的方式（LKM）加载代码，方便对软件进行更改和操作，不会对原先的Hypervisor产生大量的更改，移植性和操作性强。

* 运行环境改变

当运行环境发生变化，一般指运行的平台或者内核发生变化，但需要保证基本的内核版本要求。但是只需要对内核做部分改动，以及直接将其余代码以LKM的方式加载运行。

* 用户需求改变应对

将来用户需求可能发生的改变是增加更多的功能需求。依据软件设计的特点，通过打补丁的方式，完成功能添加部分，能够灵活应对需求改变。

### 2.1.4运行需求

1. **用户界面**

因基于linux平台，故软件是以软件包的形式使用，故不存在图形界面的交互，需保证命令行交互。

1. **软件接口**

重要的软件接口是软件的安装过程。主要的软件代码分为两个部分：对Hypervisor小部分更改的代码，和附加的LKM模块中的代码。

1. **故障处理**

* 系统发生异常

当系统发生异常的时候，将相关的日志信息记录下来。并进行软件维护。

* 出错处理需求

虚拟化安全套件的需求设计尽量趋于全面，根据用户的需求，指定合适的设计和测试方案，测试要全面，测试用例要合适和全面，发现尽可能多的软件错误和缺陷，并加以纠正，从而减少后期维护所付出的成本。

系统发生执行异常，进行日志记录，将异常信息记录在日志中。

## 可行性分析

从需求和技术成熟度、前瞻性、技术风险等方面分析可行性。

方案可行性 论文 实验证明技术问题

虚拟机安全套件该项目具有明确的前瞻性和较强的创新性，充分考虑了理论技术问题及其解决方案，形成了深入、先进的思路和方法。从创建可信执行环境、监控虚拟机与宿主机交互、隔离虚拟机内存三个方面解决关键问题，控制内存信息访问，提供了对虚拟机中敏感信息的安全防护。

**1）提出的同层隔离机制的地址空间隔离方案可行。**

第一，由于普通的系统中，HYpervisor处于最高特权级别，通过创建与Hypervisor同一特权层的地址空间，能够实现非可信Hypervisor下的可信执行环境（TEE），保护系统关键数据。

第二，当Hypervisor不可信的时候，不依赖于创建的更高特权级别组件，也不依赖于特定的硬件设备创建的组件，减少特权级别之间的切换开销，避免硬件设备带来的移植性差等问题。

第三，SKEE已将同层隔离的技术应用到手机中，使得创建的地址空间与内核属于同一特权级别，该技术被应用到手机中。

**2）提出的监控虚拟机与宿主机交互方案可行。**

1. 由于虚拟机与宿主机交互的方式是唯一的，可以通过监控唯一的交互

数据，避免非可信的Hypervisor攻击交互数据，保障虚拟机运行时状态信息安全性。

1. HA-VAMI针对ARM架构64位系统，提出了虚拟机退出时对交互数

据进行监控的方案，说明监控虚拟机与素质及交互数据的方案是可行的。同样实验室中已完成该部分实验。

**3）提出的虚拟机地址映射监控方案可行。**

1. 当系统运行时，由于所有的数据存放在内存中，通过控制内存访问，

实现对虚拟机中用户数据的保护。

1. 由于虚拟机的地址映射是宿主机通过写EPT表（唯一）完成的，通过

监控EPT写操作，实现内存访问控制。

1. 通过内存动态标记技术，实现对虚拟机的内存高强度隔离，控制内存

访问，能够避免非可信的Hypervisor或者恶意的虚拟机对虚拟机的恶意访问，减弱跨域攻击和同驻攻击带来的威胁。

# 总体技术

总体设计是针对当前的研究问题分析威胁模型，对研究目标进行设计与实现的过程。首先，仔细分析了威胁模型，并通过假设排除一些并不考虑的威胁与攻击。其次，对整个系统进行架构设计，明确各个功能模块之间的关系，对各个功能的实现提出关键技术点，并完成各个的技术偏离表。

## 威胁模型和假设

### 威胁模型

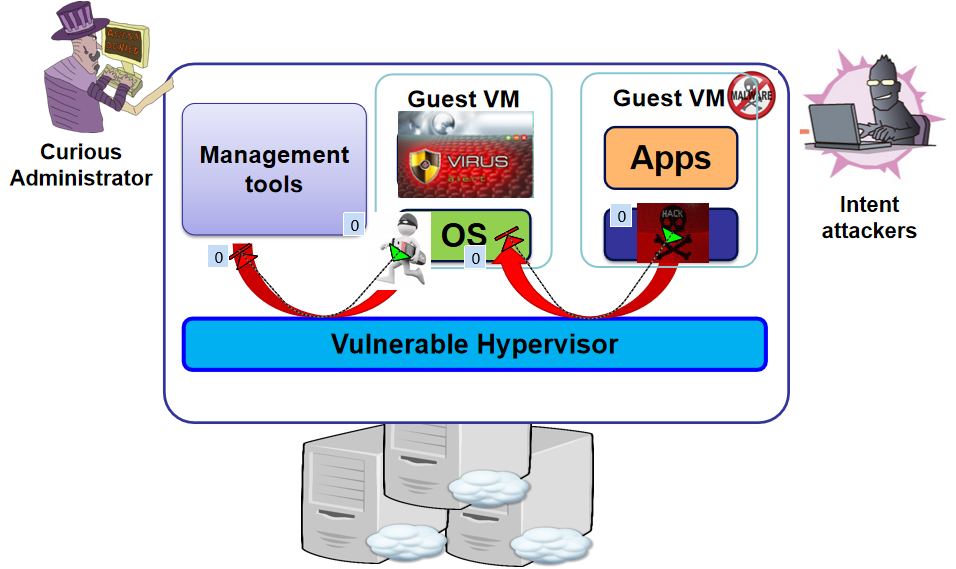


图4. 威胁模型示意图

攻击者主要从本地或远程的方式攻陷Hypervisor，随后可以进行各种攻击，主要有两种攻击，一种是攻击者篡改虚拟机和宿主机进行上下文切换时的交互数据，另一种攻击是内存越界访问攻击，例如多映射攻击和双映射攻击。

1. 攻击Hypervisor

远程攻击者和本地攻击者都可以攻击Hypervisor。远程攻击者可以通过同驻攻击和跨域攻击的方式攻陷Hypervisor，主要是利用虚拟机上软件漏洞先攻击其中的一台虚拟机，随后再通过逃逸的方法攻击Hypervisor。本地攻击者可以直接利用Hypervisor本身的漏洞直接攻击Hypervisor。

1. 攻击虚拟机与宿主机交互数据

当虚拟机和宿主机进行交互，产生虚拟机退出和虚拟机进入。采用分时策略，交互最终可以实现一个处理器上运行虚拟机或者宿主机，主要是通过虚拟机上下文切换实现。其中，上下文主要保存在VMCS，VMCS中包含虚拟机和宿主机的重要信息，以便上下文正常切换，处理器获取这些重要信息后能够正常运行虚拟机/宿主机。虚拟机退出时，虚拟机的特权寄存器等信息重新保存到VMCS，宿主机的信息被从VMCS中加载，随后宿主机正常运行。虚拟机进入时，宿主机的特权寄存器等信息重新保存到VMCS，虚拟机的信息被从VMCS中加载，随后虚拟机正常运行。攻击者篡改交互数据，篡改虚拟机地址映射页表地址（EPTP），导致加载恶意页表；篡改虚拟机/宿主机的RIP，导致控制流劫持攻击等；篡改其它数据，导致加载错误的信息；泄露VMCS中的系统关键信息。

1. 内存越权访问攻击

虚拟机的地址映射需要两套页表，自身系统中的一套页表和宿主机管理的扩展页表（EPT）。EPT作为一套页表，包含GPA与HPA的地址映射关系。攻击者攻陷Hypervisor之后，可以通过访问VMCS获得EPT的地址EPTP，随后可以直接访问EPT，更改EPT中的地址映射，造成内存恶意访问攻击，例如多重映射、双映射。

多重映射（remap）：假设两台虚拟机VM1和VM2，VM1作为远程攻击者使用的虚拟机，VM2作为受害者虚拟机。当VM2使用的物理页A被使用完释放后，VM1会重新映射到这个物理页A，那么会导致VM1可以访问物理内存页A上的信息，造成信息泄露攻击。

双映射（double map）：假设两台虚拟机VM1和VM2，VM1作为远程攻击者使用的虚拟机，VM2作为受害者虚拟机。VM1使用虚拟地址C1、C2，对应的物理地址是D1、D2，VM2使用虚拟地址C3、C4，对应的物理地址是D3、D4。攻击者去修改C1对应的物理地址，通过更改C1对应VM1的EPT页表的最后一级，使得最后一级指向物理页D3，那么最终会造成虚拟地址C1对应的物理内存页是D3，而不是D1。这样可以访问D3上的数据，从而实现了用户数据泄露攻击。

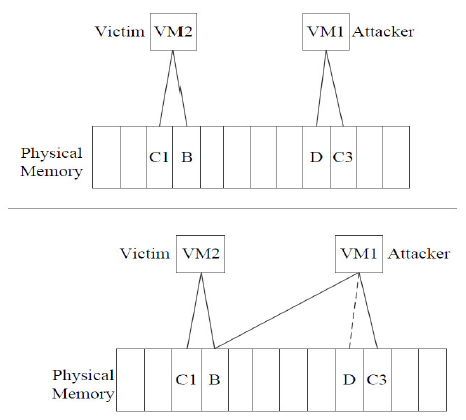


图5. 重映射攻击

1. DMA攻击

当Hypervisor被攻陷后，攻击者可以直接通过一定的内核函数直接构成虚实地址映射，即DMA映射，从而直接访问某些关键的数据结构。这些关键数据包括Hypervisor关键交互数据(VMCS和EPT)，当这些数据被直接访问后，会造成系统信息泄露，关键的控制寄存器信息泄露，导致控制流劫持攻击，内核安全机制（DEP、SMEP）被关闭，整个系统安全受到严重威胁。

### 假设

1. 假设使用的资源是可信的，包含处理器、总线、IO传输。
2. 假设可信启动，在启动过程中并不受到恶意的攻击。
3. 不考虑攻击者实施DOS攻击，侧信道攻击，基于硬件的攻击（冷启

动攻击或者Rowhammer攻击）。针对Hypervisor的DOS攻击会导致Hypervisor无法正常运行，VM所依赖的Hypervisor无法给VM提供资源管理服务，整个系统都会停止运行和服务。侧信道攻击可以导致内存信息泄露，假设不存在这样的攻击方式。

## 总体设计

该节一般简要介绍下系统的构成，画出系统的总体结构图和总体流程图。

虚拟机隔离与监控技术，主要是在KVM中添加虚拟机安全套件的方式完成，用于对虚拟机的运行时安全进行监控，尽可能减弱虚拟机跨域攻击造成的信息泄露。该方法是基于软件实现的，相比基于硬件方法实现的虚拟机安全防护[6][8]，其优点在于可移植性强，适用的平台广；相比于基于软件的大规模修改Hypervisor的方法[9][10],本方法对系统的修改小，适用于商业平台，作为内核模块可加载。架构图如下：

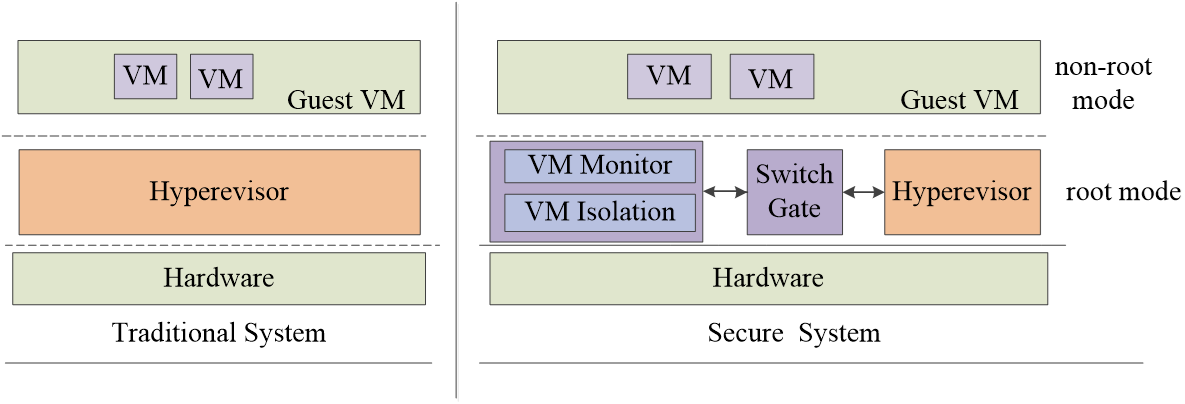


图6. 系统总体架构图

该系统的组成分为传统的Hypervisor和虚拟机安全套件，以及切换门。

虚拟机安全套件，由安全执行环境，虚拟机映射安全监控，虚拟机安全隔离组成。

切换门，用于两个环境进行切换，能够保障安全性，环境切换过程不可绕过，不可伪造，不可中断。

系统的主要特点是通过监控虚拟机与宿主机交互，监控虚拟机的地址映射，将交互和地址映射放在创建的隔离地址空间中进行处理，从而使得非可信的Hypervisor无法直接访问重要的交互数据、无法直接访问虚拟机地址映射使用到的页表，阻止攻击者进一步实施破坏虚拟机运行时状态数据攻击和内存多映射、双映射攻击。基本实现方案是通过对原Hypervisor系统VM与HOS交互、VM地址映射设置挂钩，进而实现监控。代码是通过使用LKM模块方式实现。

系统执行流程：加载虚拟机安全套件系统模块（LKM方式），开启虚拟机；通过挂钩的方式监控原Hypervisor系统中VM与HOS的交互操作，监控VM地址映射；当所监控事件发生时，判断是VM地址映射还是交互操作，随后切换到新的地址空间中进行VM地址映射或者交互操作；当处理完成后，需进行环境切换跳回到原Hypervisor去执行；此过程可以不断循环，如若关闭系统，先关闭虚拟机，随后卸载虚拟机安全套件系统模块，结束系统运行。

## 系统架构

需要画出系统的总体架构图，并且详细介绍各个模块以及模块之间的接口。

虚拟机安全套件系统包含三个模块，地址空间隔离、VM与HOS交互监控、VM地址映射监控。为虚拟机的创建、运行、销毁提供安全服务。

地址空间隔离，采用同层隔离的方法实现了地址空间隔离，创建与hypervisor同层的隔离执行环境，目的是减少性能开销，同时增强安全性。当虚拟机安全套件运行在一个相对安全的隔离的执行环境中，可以免受非可信虚拟化层的攻击威胁，提供对虚拟机安全组件的保护。针对隔离执行环境中使用到的特权寄存器，MMU和DMA，当然要设置一些安全策略对安全隔离空间进行保护，包括监控特权寄存器访问、监控对MMU的访问，监控I/O访问地址，避免安全隔离地址空间受到攻击，从而绕过安全监控甚至破坏安全隔离空间的完整性。

虚拟机映射监控，为了使得VM的内存资源隔离，需要在地址映射的时候对内存资源进行隔离。同时对地址映射进行监控，保护关键的数据结构，防止跨域攻击从而泄露免敏感数据。

虚拟机与宿主机安全切换，监控的主要内容是Hypervisor 和VM 之间的交互关键数据，即切换上下文（VMCS）。分为监控虚拟机上下文切换和虚拟机退出处理模块。

系统基本架构：在宿主机内核层创建新的地址空间（HyperMI World），与原Hypervisor（称为Hypervisor）通过安全门进行交互，安全切换门作为交互的唯一通道，不可绕过，且有一定的安全性。 Hypervisor中部分功能会被保留、添加 或者移除。其中，虚拟机退出事件处理功能被保留，挂钩监控功能被添加，将虚拟机地址映射（EPT操作）和交互操作移除。Hypervisor中添加挂钩来监控VM与HOS的切换、监控VM的地址映射。在HyperMI World中有虚拟机安全套件系统使用的关键数据（Mark表），交互监控包含上下文切换和退出重定向，地址映射监控包含物理页动态标记与跟踪和EPT地址映射监控。该隔离的地址空间运行过程中产生的代码段和数据段在HyperMI内存区域中，该区域不可被Hypervisor随意访问。整个系统架构图如下：

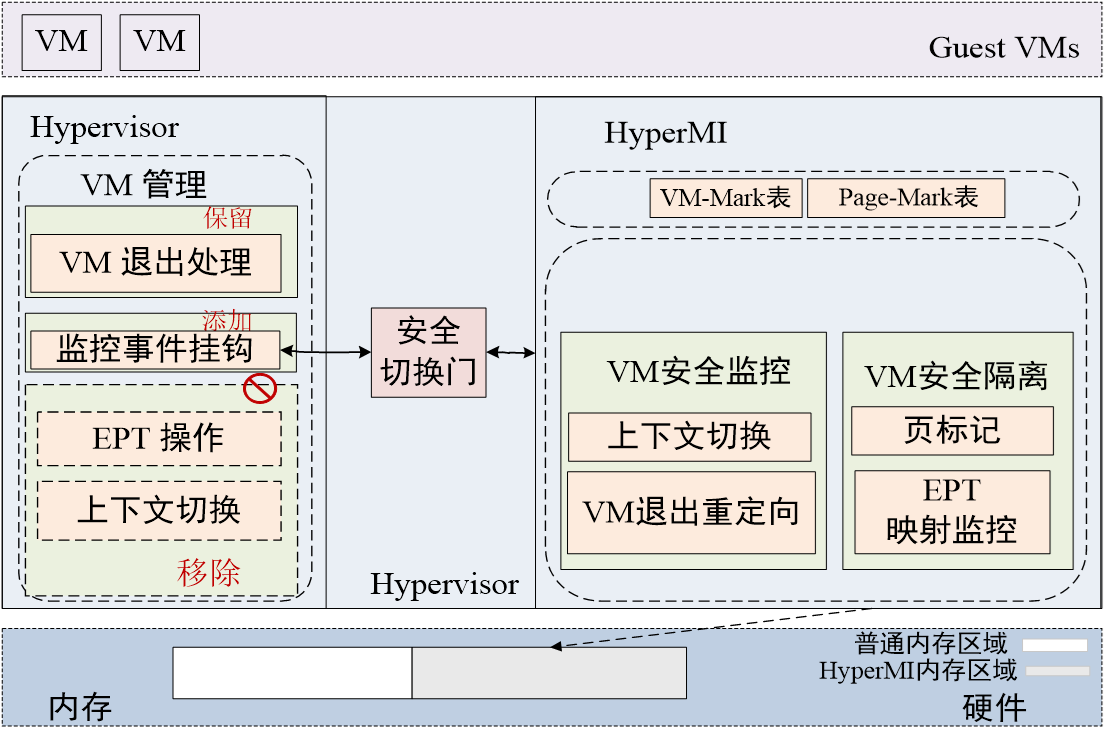


图7. 系统架构图

地址空间隔离提供了一个安全的隔离的地址空间，虚拟机安全映射和虚拟机与宿主机安全切换是运行在隔离的地址空间中，切换门是用来使得原来Hypervisor和虚拟机安全套件进行切换的接口。这三个子系统存在一定的关系，地址空间隔离模块为交互监控模块和虚拟机地址映射监控模块提供安全隔离的执行环境。交互模块和虚拟机地址映射模块为虚拟机的运行提供安全保护。其中，交互模块为虚拟机运行时状态信息提供保护，保证虚拟机的正常安全运行；虚拟机地址映射模块为虚拟机内存提供安全隔离和访问控制。三个子系统模块之间的关系如图所示：



图8. 子系统交互关系图

## 系统流程

画出系统总体流程图，并详细介绍每个流程的工作过程。  
**系统执行流程**

加载虚拟机安全套件系统模块（LKM方式），开启虚拟机；通过挂钩的方式监控原Hypervisor系统中VM与HOS的交互操作，监控VM地址映射；当所监控事件发生时，判断是VM地址映射还是交互操作，随后切换到新的地址空间中进行VM地址映射或者交互操作；当处理完成后，需进行环境切换跳回到原Hypervisor去执行；此过程可以不断循环，如若关闭系统，先关闭虚拟机，随后卸载虚拟机安全套件系统模块，结束系统运行。

控制流程图如下：

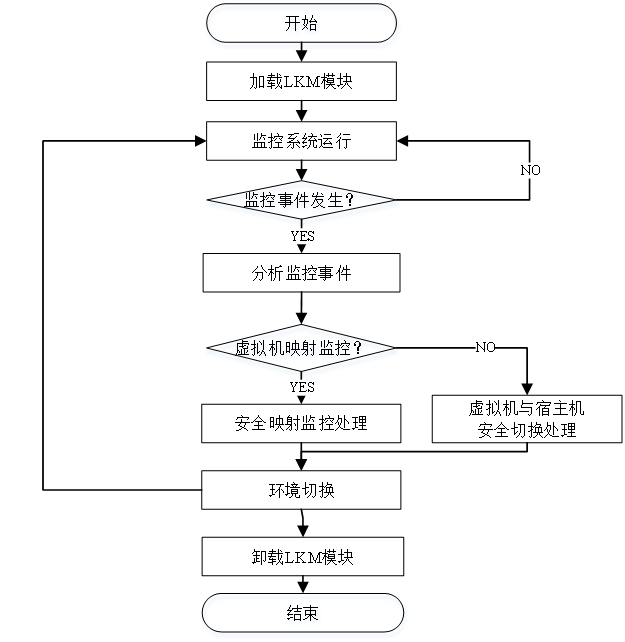


图9. 控制流程图

1. 监控事件

通过挂钩的方式在Hypervisor监控交互过程和虚拟机地址映射中使用。

交互关键数据主要是指VMCS，需要将VMCS放在新地址空间HyperMI World（HW）中；因Hypervisor会有一些函数访问到VMCS，同样这些函数会被移到新地址空间HyperMI World中，这样监控内容会增加；那么有关VMCS主要监控包括VMCS读，写，清除过程。

虚拟机地址映射监控的主要是关键数据EPT，EPT是虚拟机地址映射表，其中包含了虚拟机物理地址到宿主机物理地址的映射。为了保护EPT，需要将EPT放在HW中；同样类似于VMCS，EPT作为地址映射中的关键数据，Hypervisor中会有许多函数直接访问EPT去完成系统功能，当EPT被放在HW中时，这些函数就无法直接访问，所以需要将EPT访问的相关函数防在HW中执行，这样监控的内容增加；那么有关EPT主要监控的关键数据

包括EPT本身、EPT创建函数、EPT遍历函数、EPT销毁函数、EPT更新函数。

这些关键数据的监控方式是通过在Hypervisor中设置hook，在函数入口设置跳转指令，导致这些函数一旦被执行，会通过安全切换门跳转到HW中去执行另外一些函数（功能相同）。从而达到实时监控的目的。

B.分析监控事件的过程

整个操作系统安全启动后，整个系统会被划分为两个区域，原执行环境和隔离的执行环境。监控系统中的关键事件，当这些事件发生的时候，会到隔离的地址空间中运行。分别对事件进行判断，判断依据是EPT是否创建、加载、更新、遍历、销毁、缺页、VMCS上下文环境是否切换，随后做出相应的事件处理。处理结束之后，切换回到原运行系统，当模块被卸载的时候，整个系统运行结束。实现流程如下图：



图10. 监控事件执行流程图

## 技术偏离表

阐述设计与需求目标的正负偏离。

|  |  |  |
| --- | --- | --- |
| **需求目标** | **设计** | **是否符合** |
| 能够确保新地址空间的隔离和安全性 | 同层隔离机制，新页表实现新地址空间保证隔离性，安全切换保证安全性 | 符合 |
| 能够确保虚拟机动态物理页跟踪及内存高强度隔离 | 针对Balloon机制和KSM机制，对物理内存进行标记实现页跟踪 | 符合 |
| 能够保证虚拟机运行时状态数据安全性 | 监控虚拟机与宿主机唯一的交互通道，确保交互数据的安全性 | 符合 |
| 整个系统开销高于原系统10%以下 | 同层隔离，TLB使用，虚拟机退出尽可能少，确保性能开销小 | 符合 |

表1. 技术偏离表

# 关键技术1：基于同层隔离机制的地址空间隔离

## 基本原理

介绍下该关键技术的大体原理。包含若干技术点。

基于同层隔离机制的地址空间隔离技术，可以为一些安全工具创建安全隔离的执行环境。在虚拟化层创建隔离执行环境，实现的方法有多种，根据国内外研究，主要有两种，嵌套虚拟化和微型Hypervisor，嵌套虚拟化的方法会导致大量的性能开销，微型Hypervisor在安全性和性能上有所缺陷，故需要安全隔离的环境，控制恶意Hypervisor随意访问该隔离的安全环境，阻止对安全隔离环境的破坏，能够以相对较低的性能开销运行。

采用同层隔离的思想创建安全隔离的地址空间。即创建与Hypervisor处于同一特权级别的地址空间，可以减小环境切换带来的性能开销。与微型Hypervisor比较，同层隔离地址空间可以防御针对Hypervisor的攻击，同时还能抵御来自恶意Hypervisor的攻击。

两个地址空间的切换要求是安全的，安全切换门可以保证两个地址空间之间切换的安全性。

为了保证隔离地址空间的安全性，需要对隔离地址空间进行安全保护。

综上所述，基本技术点包含同层隔离技术、安全切换门技术、对隔离地址空间的安全防护技术。

## 技术点：同层隔离

介绍该技术点模块的实现原理。

同层隔离技术的基本原理是创建的地址空间位于与Hypervisor相同特权级别，不依赖于更高特权级别，也不依赖于额外定制的硬件，能够提供安全隔离的执行环境，避免来自非可信Hypervisor的攻击。创建地址空间的实现方法是在虚拟化层创建另一套页表，该页表包含了原虚拟化层的地址空间，同时包含了新的地址空间，新的地址空间主要是用于保护虚拟机与宿主机交互的关键系统数据、用于虚拟机地址映射监控功能，保护这些进程运行时的代码段和数据段的安全性。

首先通过创建新的页表，64位系统上创建适合64位系统的页表，高虚拟地址空间表示内核空间，低地址空间表示用户空间。地址空间包含原地址空间的代码段和数据段。新的页表入口地址被保存在安全的地方。

## 技术点：安全切换门



图11. 安全切换门

安全切换门的目的是保证两个地址空间的切换是安全的，从原地址空间切换到新的隔离地址空间，或者从新隔离地址空间切换到原地址空间，该切换过程必须是安全的，才能避免给新隔离地址空间带来安全威胁。

切换门包含进入门、退出门和共享缓冲区，进入门是用于从原地址空间切换到新隔离的地址空间，退出门是用于从新隔离地址空间退出到原地址空间。共享缓冲区包含公共的代码段和数据段，公共的代码段是指切换的代码段，公共数据段是指切换需要的地址空间入口地址。这些切换的地址空间入口地址需要被保护，以防攻击者获取这些信息

切换的基本过程是：1）读取新页表的入口地址，2）跳转到新页表所在地址，3）切换到新的隔离地址空间完成。相关进程运行。4）读取原页表的入口地址，5）跳转到原地址空间，6）切换回原地址空间完成。

为了保证切换过程的安全性，在进入和退出过程中，使用原子操作技术保证切换的原子性，地址空间切换不可绕过性。添加原子操作后的进入过程如下：1）保存内核信息状态（特权寄存器、中断状态等）到栈中，2）关中断，3）加载新页表的入口地址到CR3寄存器中，刷新TLB，4）再次关中断，5）跳转到新隔离地址空间中。反之，退出的操作相反。在该过程中，两次中断操作可以防止攻击者跳过第一次中断获取入口与地址再次跳到新隔离地址空间中。

## 技术点：隔离空间安全防护

在通过同层隔离技术创建隔离的地址空间，使用安全切换门进行地址空间切换，那么需要对地址空间的隔离性和安全性进行防护，第三个技术点是对隔离地址空间进行安全防护。隔离地址主要使用新的页表创建新地址空间，特权寄存器存放页表入口地址，那么就需要对新页表、特权寄存器的访问进行安全防护。此外，通过DMA访问可以直接访问物理内存空间，并不经过页表，还需要对DMA访问进行控制。综上，防护措施主要是三点：对新页表的访问控制、特权寄存器访问控制、DMA访问控制。

1. 新页表访问控制

虚拟化层Hypervisor在当前系统都拥有一定的权限，拥有对页表访问的权限，一旦Hypervisor被攻击，攻击者就可以通过破坏页表对新隔离地址空间进行破坏。可能的攻击如下：1）在切换的过程中，攻击者可以加载恶意的页表到CR3寄存器中，或者篡改原地址空间中的页表来映射新隔离地址空间中的地址（敏感数据的地址），最终恶意访问敏感数据。2）因新页表中包含原地址空间中的代码段和数据段，当新隔离地址空间在进行时，原地址空间中的代码依旧可以运行，一旦这些代码含有一定的漏洞，被攻击后，可能会破坏新地址空间中的一些进程。

针对如上攻击，防护措施如下：1）针对第一种攻击，虚拟机安全套件系统的代码和数据段在原地址空间（页表）中并没有映射，为了保护新页表入口地址不被泄露，从原系统页表上移除了新页表的入口地址，剥夺了访问CR3寄存器的能力，这样就可以避免加载非法页表攻击和绕过安全隔离地址空间攻击。2）针对第二种攻击，在新页表中对原地址空间中的代码段进行访问控制，使得代码段不具备运行权限。对页表写权限进行控制，防止攻击者直接篡改页表内容更改代码段的访问权限，或者直接篡改页表映射，导致错误映射并泄露敏感信息。

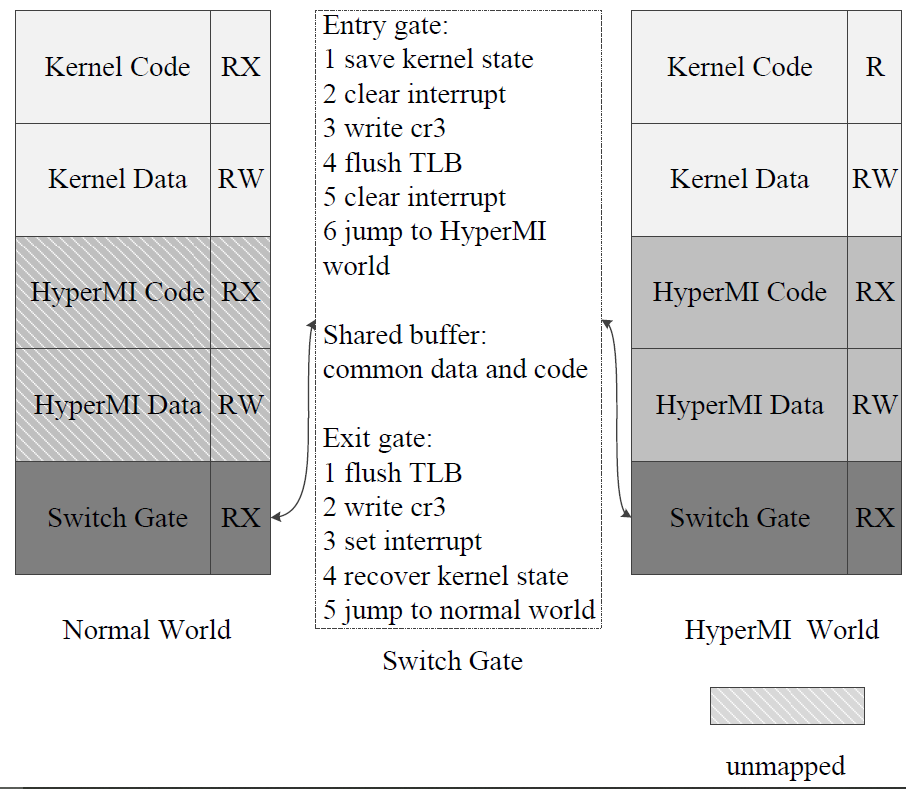


图12. 页表安全防护

1. 特权寄存器访问控制

需要对一些特权寄存器进行访问控制，分别是CR0、CR3、CR4寄存器。通过剥夺这些特权寄存器的访问权限，让这些寄存器的访问在新的隔离地址空间中运行。

1. DMA访问控制

DMA访问控制。DMA的特点是直接访问物理内存，为了避免攻击者直接通过DMA方式访问敏感数据所在内存，采用IOMMU方式对映射的地址进行访问控制，对IOMMU中的页表删除映射到隔离空间的地址映射，同时在映射发生时验证映射地址的合法性。

## 技术评价

### 技术特色

介绍其技术上的独到之处或特点。

同层隔离技术是一种软件的方法，可以提供安全隔离可执行环境（TEE），保证系统运行的安全性，相比较其余的硬件方法和软件方法，其移植性相对较好，性能开销相对较低。对于云平台提供商，硬件需要定制，移植性相对较差，可广泛适用性相对较差；其余软件方法主要包含嵌套虚拟化方法，嵌套虚拟化通过创建比Hypervisor层更高层的软件层，但是频繁的特权级别切换会导致性能开销增大。综上，同层隔离技术移植性较好，适用于云平台提供商，性能开销相对较低。

### 创新性证明

三星公司通过使用SKEE技术，将该技术应用到手机中，SKEE在ARM系统上创建了与内核同特权级别的地址空间，即同层隔离思想。Skee: A lightweight secure kernel-level execution environment for arm. In: Network and Distributed System Security Symposium (2016)

南京大学创建ED-Monitor框架，与Hypervisor同地址空间，与Hypervisor同特权级别，该框架为虚拟机提供了安全的执行环境，同样使用了同层隔离思想。

Deng, L., Liu, P., Xu, J., Chen, P., Zeng, Q.: Dancing with wolves: Towards practical event-driven vmm monitoring. Acm Sigplan Notices 52(7), 83–96 (2017)

西安交通大学的SecPod系统创建了与虚拟化层同级别的可执行环境。

这几年的学术研究和商业技术研究表明，同层隔离思想或者类似同层隔离技术被运用到很多地方，可以减小特权级别之间的切换，作为软件方法，可以避免大量使用硬件，同层隔离技术的安全性相对较高，不再使用更高级别的方法，对系统的改造相对较小。

### 成果说明

此关键技术申请的专利、发表的文章或系统应用情况的说明。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 专利 | 文章 | 系统应用 |
| 成果说明 | 《一种安全虚拟机实现的方法与装置》 | 无 | 国家重点研发项目子项目——虚拟机安全套件 |

表2. 成果说明

# 关键技术2：监控VM与HOS的交互关键数据

## 基本原理

介绍下该关键技术的大体原理。包含若干技术点。

因为虚拟机监控器和虚拟机共享底层的物理资源，同时各个虚拟机之间没有完全的隔离性，同时虚拟机（VM）和宿主机（HOS）交互方式是通过在同一CPU上进行上下文切换，关键交互数据主要是VMCS结构体。所以存在这样的威胁，恶意的攻击者在攻破Hypervisor后，恶意访问上下文切换过程中使用的VMCS结构体中的数据信息，篡改数据，导致宿主机或者虚拟机的特权寄存器（CR0、CR3、CR4）等关键信息被篡改，导致系统被进一步攻击，系统数据被泄露等严重问题。为了阻止攻击，提出了虚拟机与宿主机交互监控技术，该技术主要包含两个技术点，隐藏VMCS结构体在隔离地址空间技术和剥夺宿主机的VMCS结构体访问功能技术[5]。



图13. 虚拟性与宿主机交互监控过程示意图

## 技术点：隐藏VMCS

介绍该技术点模块的实现原理。

**监控上下文切换过程，并隐藏VMCS结构体的地址。**

宿主机和虚拟机交互的方式是在同一个虚拟CPU（vCPU）使用分时策略，当vcpu上运行虚拟机时，需要从宿主机状态切换到虚拟机状态，基本过程是：1）保存宿主机各信息到VMCS结构体中的宿主机部分，2）将VMCS结构体中虚拟机部分写到当前vcpu中的各个特权寄存器等，3）运行vcpu。当vcpu运行宿主机时，过程相反。那么，VMCS结构体十分重要，是两者交互的关键系统数据，必须要阻止攻击者恶意访问VMCS结构体。

为了达到这样的目的，避免攻击者在Hypervisor被攻击后恶意访问VMCS结构体，虚拟机安全套件系统将VMCS结构体的地址进行了隐藏，将其地址隐藏在新的安全隔离地址空间中，这样可以避免攻击者恶意访问VMCS结构体，从而避免攻击者进行进一步的攻击，阻止系统信息的泄露。

## 技术点：监控上下文切换

介绍该技术点的实现原理。

**剥夺宿主机的VMCS结构体访问功能技术。**

上一个技术点介绍了隐藏VMCS结构体的地址，避免攻击者恶意访问VMCS结构体，但是正常程序的运行也需要访问VMCS结构体，例如，VMCS创建，VMCS访问，VMCS销毁。为了使得系统能够正常运行，虚拟机安全套件系统将VMCS结构体访问功能剥夺了，并将所有的VMCS访问功能在新的安全隔离地址空间中运行，实现技术上主要使用hook挂钩技术。

将有关VMCS访问的函数在隔离地址空间中执行，或者对某些访问函数（不需要VMCS结构体地址的函数）只返回信号信息，并不真实返回VMCS的地址。防止VMCS被恶意访问和篡改。

## 技术评价

### 技术特色

介绍其技术上的独到之处或特点。

虚拟机与宿主机交互关键数据监控技术包含两个技术点，隐藏上下文切换中VMCS结构体地址技术和剥夺宿主机访问VMCS结构体功能技术。

其独到之处在于虚拟机与宿主机交互的唯一关键数据和方式被严格监控，同时为了保障系统的正常运行，还剥夺了宿主机对VMCS结构体的访问功能，整个交互过程被监控，所以攻击者无法通过非可信Hypervisor恶意篡改虚拟机的运行时状态信息，同时可以保护EPTP（VMCS结构体中的一个关键数据）的安全访问。该技术可以保证虚拟机的运行时关键系统数据不被篡改，CR3、CR4、CR0等，可以保证DEP、SEMP机制的正常运行，阻止系统被进一步攻击。

### 创新性证明

通过查新（专利、公开网站、学术论文）证明此关键技术的先进性。

中科院信工所发表了HA-VAMI，该论文提出了在ARM架构上完成监控VM与HOS的交互，监控进程是在TrustZone中的安全区域完成。

ED-Monitor 创建了与内核同特级别和同地址空间的隔离空间，在该隔离空间中监控了对VMCS的访问。

### 成果说明

此关键技术申请的专利、发表的文章或系统应用情况的说明。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 专利 | 文章 | 系统应用 |
| 成果说明 | 《一种安全虚拟机实现的方法与装置》 | 无 | 国家重点研发项目子项目——虚拟机安全套件 |

表3. 成果说明

# 关键技术3：动态内存标记与跟踪

## 基本原理

介绍下该关键技术的大体原理。包含若干技术点。

因为虚拟机监控器和虚拟机共享底层的物理资源，同时各个虚拟机之间没有完全的隔离性，尤其是内存这样的硬件，所以存在这样的威胁，恶意的虚拟机或者恶意的Hypervisor还可以对受害虚拟机的物理资源随意访问，实现跨域攻击，导致信息泄露等严重问题，在这些攻击过程中可能使用针对内存映射的多重映射攻击和双映射攻击。通过使用动态标记与跟踪技术，使得所有虚拟机之间的内存实现高强度隔离，如图。



图14. 内存高强度隔离效果图

## 技术点：虚拟机地址映射监控

介绍该技术点模块的实现原理。

虚拟机地址映射监控，需要监控关键数据结构EPTP。将其地址进行隐藏，同时将访问EPTP数据相关的函数放在隔离地址空间中执行，或者返回信号信息，这些主要是通过挂钩的方式实现，并不直接返回EPTP的地址，目的是防止攻击者直接恶意访问EPTP，从而篡改EPTP加载恶意的EPT（扩展页表）。

## 技术点：内存动态标记与跟踪

介绍该技术点的实现原理。

内存动态标记与跟踪技术，主要包含两种技术，内存标记与跟踪技术和内存动态标记跟踪技术，这两种技术都是在安全隔离的地址空间中运行。

内存标记技术主要是通过对真实的物理内存，即虚拟机或者宿主机使用的物理内存进行标记，标记每一个物理内存页的使用属主。当内存被系统分配使用的时候，能够对内存的使用进行分辨，对当前正在被使用中的内存，不再给宿主机或者其余是虚拟机进行分配，可以防止内存双映射攻击；对于未被使用内存直接进行分配。对于即将被释放的内存，先清空其物理页上的信息，随后进行释放，目的是防止其余恶意虚拟机或者被攻陷的宿主机重新映射到物理页上进行内存多重映射攻击。

|  |  |  |
| --- | --- | --- |
| **Page—Mark表** | | |
| **标记** | OwnerId | Used |
| **描述** | 属主VM或  Hypervisor的ID号 | 是否被使用 |

表4. page-mark表（不考虑动态标记）

内存动态标记跟踪技术是针对KSM机制和Balloon机制对内存标记跟踪技术进行了扩展，KSM机制和Balloon机制可以使得虚拟机所拥有的内存随时发生变化，也就是在系统运行的过程中，内存会被分配、释放、重新分配，在内存释放之前还可能会被重新分配给其他虚拟机，所以可能会造成内存标记的属主发生变化，与内存标记与跟踪功能中内存属主唯一性相互矛盾，导致内存标记跟踪功能失败。为了防止这种情况发生，通过对KSM机制和Balloon机制中内存易主进行监控，动态地变化内存的属主标识，保证内存动态标记跟踪功能的正常运行。

|  |  |  |  |
| --- | --- | --- | --- |
| **Page—Mark表** | | | |
| **标记** | OwnerId | Used | SharedBit |
| **描述** | 属主VM或  Hypervisor的ID号 | 是否被使用 | 是否是共享页 |

表5. page-mark表（考虑动态标记）

## 技术评价

### 技术特色

介绍其技术上的独到之处或特点。

虚拟机地址映射监控技术主要包含两个技术点，虚拟机地址映射监控和内存动态标记与跟踪技术。首先在隔离地址空间中监控虚拟机地址映射，在系统运行时，虚拟机的数据存放在内存中，所以内存是重要硬件之一，虚拟机的内存映射包含两类页表，一个是自身的页表，另一个是由宿主机管理的EPT（扩展页表）。那么为了统一地监控地址映射，虚拟机安全套件剥夺了宿主机管理所有EPT的功能，能够监控内存的分配。其次，在监控内存分配时，对内存进行属主标记并进行跟踪。这两个技术点监控了内存映射时的关键点，卡住了内存映射关键过程，同时考虑内存动态变化属主的问题，对内存运行时过程进行了全面的考虑。

### 创新性证明

通过查新（专利、公开网站、学术论文）证明此关键技术的先进性。

复旦大学提出的CloudVisor对内存进行了标记与跟踪，从而对虚拟机进行了内存隔离。

ED-Monitor 创建了与内核同特级别和同地址空间的隔离空间，在该隔离空间中监控了对EPT的访问，保证不可非法篡改EPT，从而阻止进一步的攻击。

### 成果说明

此关键技术申请的专利、发表的文章或系统应用情况的说明。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 专利 | 文章 | 系统应用 |
| 成果说明 | 《一种安全虚拟机实现的方法与装置》 | 无 | 国家重点研发项目子项目——虚拟机安全套件 |

表6. 成果说明

# 原型系统评估

## 安全性测试

此处安全测试即功能测试，涉及三个部分：地址空间隔离、虚拟机与宿主机交互监控、虚拟机地址映射监控。需要对着三个功能模块进行测试，达到一定的功能和安全性需求。

地址空间隔离功能测试要求在新的隔离的地址空间的数据不可被其余地址空间的进程访问。保证隔离地址空间的绝对隔离性。阻止攻击者破坏隔离地址空间，无法创建安全隔离的可执行环境。

虚拟机与宿主机交互监控功能测试要求虚拟机与宿主机的交互的关键数据在隔离地址空间中不被攻击者恶意访问、篡改。其中，交互关键数据包括虚拟机与宿主机在VCPU切换时的上下文信息，包含客户机和宿主机的特权寄存器、状态信息等。功能要求能够阻止泄露虚拟机和宿主机的关键信息，阻止控制流攻击等。

虚拟机地址映射监控功能测试要求保证虚拟机的地址映射相关信息不被恶意访问、篡改等，能够保证虚拟机内存的高强度隔离，阻止攻击者通过其余被攻陷的虚拟机进行攻击，或者阻止攻击者通过在Hypervisor层上直接篡改地址映射数据（EPT），随后进行内存重映射或双映射攻击。

### 测试环境与配置

测试环境、配置说明，测试用例和测试过程。

测试环境包括硬件环境和软件配置环境。

|  |  |
| --- | --- |
| **资源名称/类型** | **配置** |
| **宿主机KVM** | 服务器，主频2.0GHZ，硬盘1T，内存32G，内核版本linux-4.4.1，系统版本ubutnu-14.05 |
| **客户机** | 内存8G，磁盘大小30G，系统版本ubuntu-12.04.5，2台 |

表7. 测试环境

测试方法：分别实现如下表中的攻击，攻击代码实现方式是通过LKM方式。

|  |  |  |  |
| --- | --- | --- | --- |
| **攻击** | **描述** | **测试功能** | **测试程序** |
| **DMA攻击** | DMA方式访问安全区域 | 地址空间隔离功能 | Test\_Isolation\_DMA.c |
| **代码注入攻击** | 注入代码，覆盖hooked的函数 | 地址空间隔离功能 | Test\_Isolation\_injection.c |
| **CVE-2009-2287** | 加载恶意的客户机CR3寄存器 | 虚拟机与宿主机交互监控功能 | Test\_VMCS\_CR3.c |
| **CVE-2017-8106** | 加载恶意EPTP | 虚拟机与宿主机交互监控功能、虚拟机地址映射监控功能 | Test\_VMCS\_EPTP.c |

表8. 测试方法

测试步骤：

A.按照测试方法，分别执行表中的4类攻击，主要方式是使用加载卸载LKM模块命令。

|  |  |  |  |
| --- | --- | --- | --- |
| **攻击** | **加载模块** | **卸载模块** | **dmesg查看** |
| **DMA攻击** | insmod DMA.ko | rmmod DMA.ko | dmesg grep ‘attackDMA’ |
| **代码注入攻击** | insmod injection.ko | rmmod injection.ko | dmesg grep ‘attackInject’ |
| **CVE-2009-2287** | insmod CR3.ko | rmmod cr3.ko | dmesg grep ‘attackCR3’ |
| **CVE-2017-8106** | insmod EPTP.ko | rmmod EPTP.ko | dmesg grep ‘attackEPTP’ |

表9. 测试步骤

1. 在卸载模块命令执行之后，使用dmesg命令查看攻击成功或失败等信息。

### 测试结果分析

图表展示、客观分析、给出结论。

|  |  |  |  |
| --- | --- | --- | --- |
| **攻击** | **描述** | **防护** | **测试** |
| **DMA攻击** | DMA方式访问安全区域 | DMA映射验证 | 地址空间隔离功能 |
| **代码注入攻击** | 注入代码，覆盖hooked函数 | 页表访问防护 | 地址空间隔离功能 |
| **CVE-2009-2287** | 加载恶意CR3 | 特权寄存器监控 | 虚拟机与宿主机交互监控功能 |
| **CVE-2017-8106** | 加载恶意EPTP | EPT、VMCS隐藏 | 虚拟机与宿主机交互监控功能、虚拟机地址映射监控功能 |

表10. 测试结果分析

结合第三节的技术偏离表，所有的功能通过测试。



## 性能测试

性能需求:三种功能的性能开销平均小于10%。

### 测试环境与配置

测试环境、配置说明，测试用例和测试过程。

测试环境：2台ubuntu14.04 LTS, Linux-4.4.0, 1G内存，磁盘大小20G。宿主机 Linux4.4.1系统，32G内存，硬盘大小256G。

测试工具：SPEC CPU2006微观测试集。

测试步骤：

1. 使用测试工具测50次，将结果进行平均化。
2. 编译运行测试代码

|  |
| --- |
| cd /root/cpu2006/  ./install.sh  echo "starting SPECCPU2006 at $(date)"  source shrc  bin/runspec --action=validate -o all -r 4 -c \ Example-linux64-amd64-gcc43.cfg all  echo "SPECCPU2006 ends at $(date)"  本次示例中runspec脚本用到的参数中，--action=validate表示执行validate这个测试行为（包括编译、执行、结果检查、生成报告等步骤），-o all表示输出测试报告的文件格式为尽可能多的格式（包括html、pdf、text、csv、raw等），-r 4（等价于--rate --copies 4）表示本次将会使用4个并发进程执行rate类型的测试（这样可以最大限度地消耗分配的4个CPU线程资源），--config xx.cfg表示使用xx.cfg配置文件来运行本次测试，最后的all表示执行整型（int）和浮点型（fp）两种测试类型。runspec的参数比较多也比较复杂，可以参考其官方网站的文档了解各个参数的细节。 |

1. 在result目录下将HTML格式的CINT2006.001.ref.html（对整型的测试报告）和CFP2006.001.ref.html（对浮点型的测试报告）两个文件进行处理。获取最后的性能测试结果。

### 测试结果分析

图表展示、客观分析，给出结论。

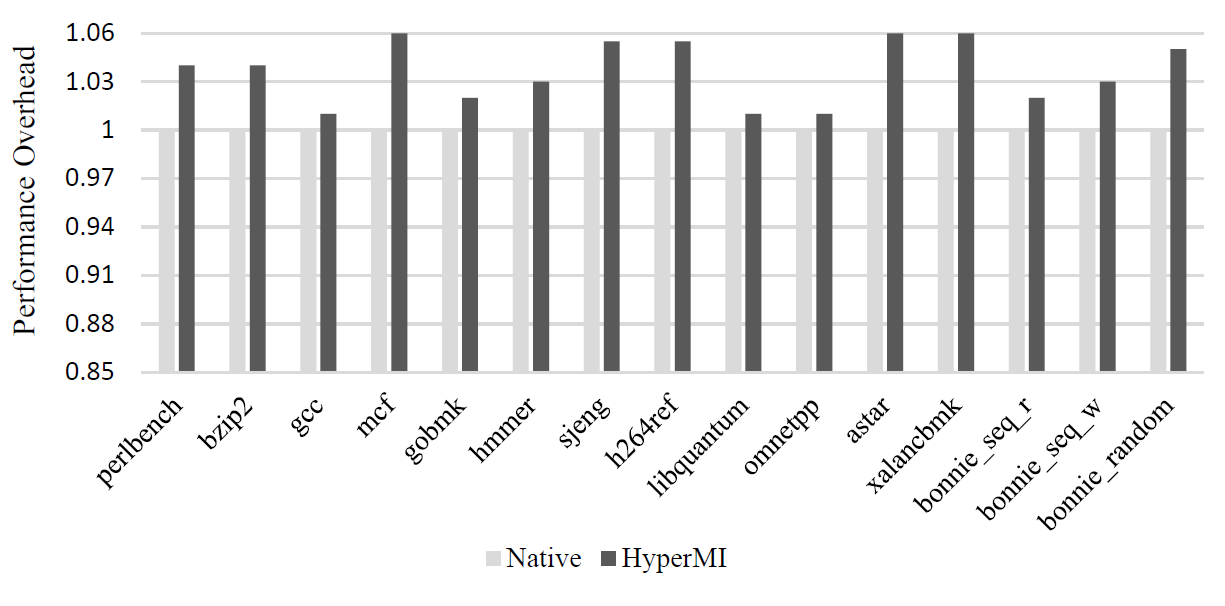


图15. 性能测试结果分析图

结果显示，与内存操作相关的性能开销大，与I/O 操作相关的性能开销小。

分析知与内存操作监控较多，I/O监控操作较少，环境切换需刷新TLB、内存页频繁加载、EPT更新时验证Page信息（延迟内存EPT访问时间），这些都与内存访问相关。

# 环境约束

该节主要详细介绍在做系统开发、部署和运行需要准备的环境。

环境约束主要参考《用户手册》文档[16]。

## 开发环境

介绍系统开发环境的搭建以及完成开发之后，怎么安装、配置、编译并且能够成功运行的

A.开发环境的搭建：

安装内核版本在linux-4.4.0以下的系统，之后重新安装编译已更改的内核linux4.4.1。

B.安装linux-4.4.1：

方法参加第XX节

C.安装secureVM模块：

将该代码放在某个目录下

D.运行secureVM模块：

编译 make

安装加载 insmod secureVM.ko

当系统运行完关闭后，按需卸载模块 rmmod secureVM.ko

## 部署环境

系统部署的环境需求、工具以及运行时维护情况

安装qemu libvirt virsh-install kvm ，部署环境的方法参见《用户手册》[16]。

# 后续说明

## 后续工作

本系统主要完成3个模块，隔离地址空间中保证关键数据不被恶意访问。动态跟踪物理内存信息，完成虚拟机地址映射监控。保证虚拟机运行时状态的安全性，完成虚拟机与宿主机的安全上下文切换监控。除了这些功能完成后，需要完成其余的后续工作。详细情况如下：

1. 内存复用涉及到的兼容性问题

动态跟踪物理信息会涉及到内存复用问题，除了处理KSM机制和Balloon机制之外，可能后续系统涉及新的内存复用方案，需要根据后续的技术不断改进当前的系统，使得提高兼容性能。

1. 隔离地址空间框架功能增加问题

隔离地址空间功能为系统提供了一个安全隔离的可执行环境（TEE），同时该系统提供了虚拟机与宿主机上下文切换监控功能，该上下文切换监控功能监控了虚拟机与宿主机交互的唯一交互通道，那么这两种功能可以保证虚拟机执行环境是安全隔离的。在这种安全隔离的执行环境中，除了监控虚拟机地址映射，还可以监控其余关键软件信息、硬件信息等，可以继续扩展其余监控功能。

## 其他事项

暂无。

# 参考文献

标准文献格式

[1] Azab, A., Swidowski, K., Bhutkar, R., Ma, J., Shen, W., Wang, R., Ning,P.: Skee: A lightweight secure kernel-level execution environment for arm. In: Network and Distributed System Security Symposium (2016)

[2] Baumann, A., Peinado, M., Hunt, G.: Shielding applications from an untrusted cloud with haven. Acm Transactions on Computer Systems 33(3), 1–26 (2014)

[3] CVE-2009-2287: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-(2018)

[4] CVE-2018-1087: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-(2018)

[5] Deng, L., Liu, P., Xu, J., Chen, P., Zeng, Q.: Dancing with wolves: Towards practical event-driven vmm monitoring. Acm Sigplan Notices 52(7), 83–96 (2017)

[6] Hoekstra, M., Lal, R., Rozas, C., Phegade, V., Cuvillo, J.D.: Cuvillo, ”using innovative instructions to create trustworthy software solutions,” in hardware and architectural support for security and privacy. In: 6 IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. (2013)

[7] Lee, H., Moon, H., Jang, D., Kim, K., Lee, J., Paek, Y., Kang, B.H.: Ki-mon: a hardware-assisted event-triggered monitoring platform for mutable kernel object. In: Usenix Conference on Security. pp. 511–526 (2013)

[8] Mckeen, F., Alexandrovich, I., Berenzon, A., Rozas, C.V., Shafi, H., Shanbhogue, V., Savagaonkar, U.R.: Innovative instructions and software model for isolated execution. In: International Workshop on Hardware and Architectural Support for Security and Privacy. pp. 1–1 (2013)

[9] Jin, S., Ahn, J., Seol, J., Cha, S., Huh, J., Maeng, S.: H-svm: Hardware assisted secure virtual machines under a vulnerable hypervisor. IEEE Transactions on Computers 64(10), 2833–2846 (2015)

[10] Keller, E., Szefer, J., Rexford, J., Lee, R.B.: Nohype:virtualized cloud infrastructure without the virtualization. Acm Sigarch Computer Architecture News 38(3), 350–361 (2010)

[11] Ben-Yehuda, M., Day, M., Dubitzky, Z., Factor, M., Har’El, N., Gordon, A., Liguori, A., Wasserman, O., Yassour, B.A., Ben-Yehuda, M.: The turtles project: Design and implementation of nested virtualization. Yehuda pp. 1–6 (2007)

[12] Champagne, D., Lee, R.B.: Scalable architectural support for trusted software. In: HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture. pp. 1–12 (2010)

[13] Evtyushkin, D., Elwell, J., Ozsoy, M., Ponomarev, D., Ghazaleh, N.A., Riley, R.: Iso-x:a flexible architecture for hardware-managed isolated execution. In: Ieee/acm International Symposium on Microarchitecture. pp. 190–202 (2015)

[14] Chhabra, S., Rogers, B., Yan, S., Prvulovic, M.: Secureme:a hardware software approach to full system security. In: International Conference on Supercomputing. pp. 108–119 (2011)

[15] Cho, Y., Shin, J., Kwon, D., Ham, M.J., Kim, Y., Paek, Y.: Hardware assisted on-demand hypervisor activation for efficient security critical code execution on mobile devices. In: Usenix Conference on Usenix Technical Conference. pp. 565–578 (2016)

[16]刘文清，张坤：《虚拟机安全套件用户手册》