附件 5

研究生学位论文中期报告

报告题目 虚拟机隔离与监控技术的研究

学生姓名 刘文清 学号 **201628018629130**

指导教师 涂碧波 职称 研究员

学位类别 工学硕士

学科专业 网络空间安全

研究方向 虚拟化安全

研究所（院系） 信息工程研究所

填表日期 **2018.12.3**

中国科学院大学制

目录

1 课题主要研究内容及进度情况

1.1 课题主要研究内容

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 33600 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 相关的漏洞，180 个和 KVM 相关的漏洞[3]，比如，CVE-2009-2287[4]在提权后，用于加载恶意的 EPT 页表。CVE-2018-1087[5]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。

本文提出了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按防护[6][7][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：

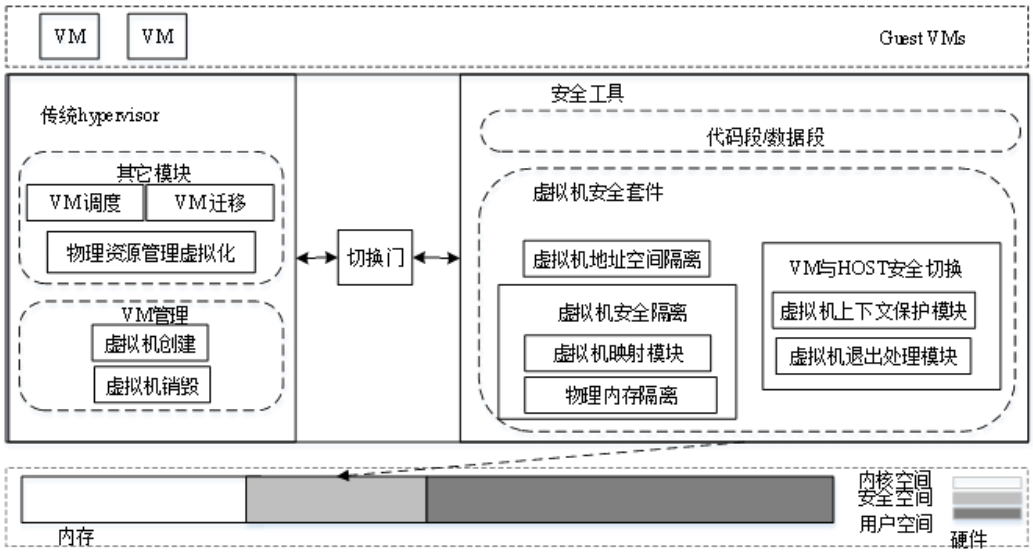

图 1. 系统总体架构图

该系统的组成分为传统的 Hypervisor 和现在的虚拟机安全套件，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，由地址空间隔离，虚拟机映射安全监控，虚拟机与宿主机安全切换组

成。为虚拟机的创建、运行、销毁提供安全服务。

地址空间隔离[11][12]，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。

虚拟机地址映射监控[13]，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[14]。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。

虚拟机与宿主机运行时监控，监控的主要内容是 Hypervisor 和 VM 之间的交互过程。分为监控虚拟机上下文切换和虚拟机退出处理模块。

控制流程图如下：

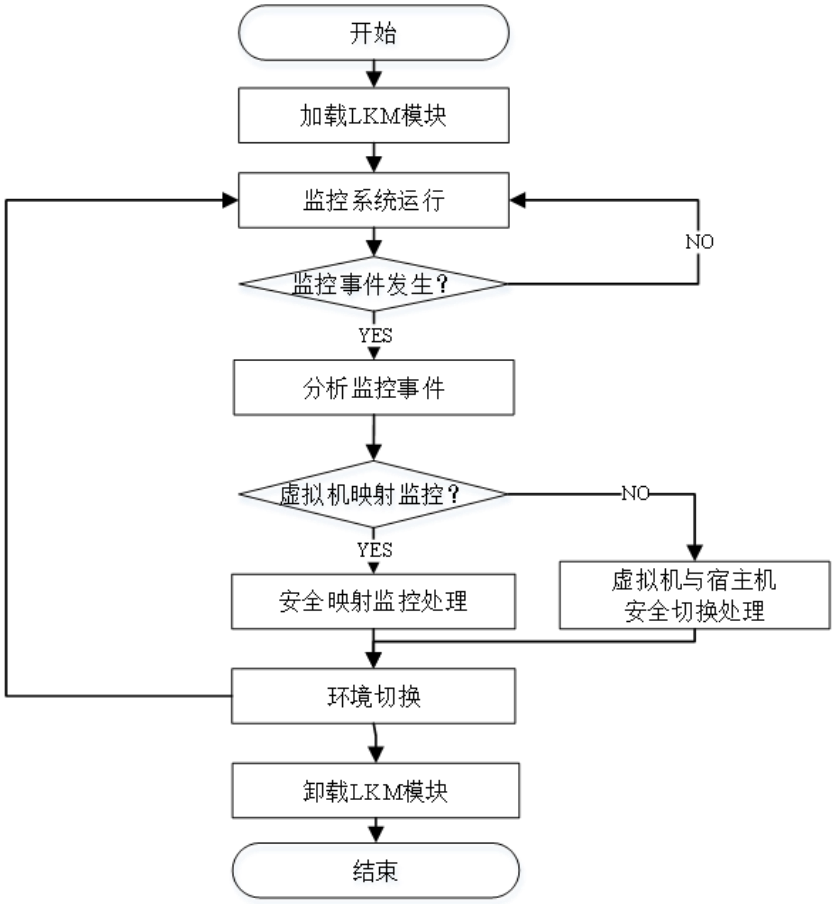

图 4. 控制流程图

1.2 进度情况

下表为原开题计划安排：

表 1. 开题计划安排

项	工作计划	进度安排
1	安全隔离执行环境的设计与实现	2018.6-2018.8
2	虚拟机监控技术的设计与实现	2018.8-2018.10
3	虚拟机隔离技术的设计与实现	2018.10-2018.12
4	论文撰写	2019.1-2019.4

表 2. 研究进度

项	工作计划	研究进度
1	安全隔离执行环境的设计与实现	100%
2	虚拟机监控技术的设计与实现	100%
3	虚拟机隔离技术的设计与实现	100%
4	论文撰写	1%

2 目前已完成的研究工作及结果

系统主要由 3 大模块构成，地址空间隔离，VM 与 HOS 安全上下文切换，VM 映射监控。3 大模块的关系如图 2.

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。

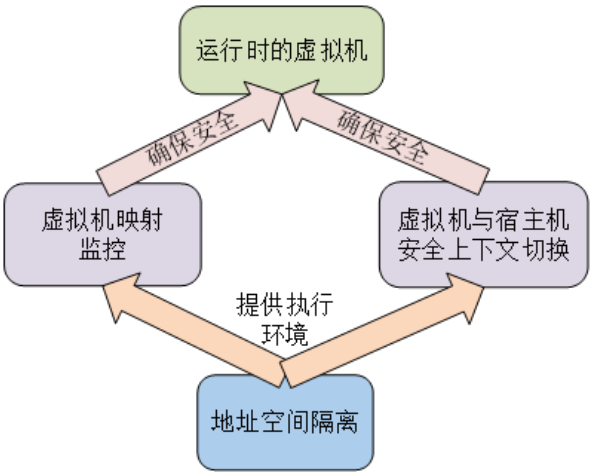

图 2. 子系统交互关系图

2.1 地址空间隔离

2.1.1 隔离空间的建立

当系统组件运行的环境并没有足够的安全保障时，其对应的功能也很难发挥它的作用，系统安全很难得到相应的保障，那么一个安全的可执行环境是系统必不可少的。当系统组件运行在安全的环境中，并且该安全环境能够达到一定的要求，能够防御一定的外部攻击，同时系统组件能够发挥其功能，更全面地服务于整个系统。

目的是实现一个安全的可执行环境，在该安全环境中可以运行一些系统组件，包括虚拟机上下文安全切换、虚拟机退出处理、虚拟机映射监控、虚拟机内存标记与跟踪等，这些组件用来进行监控防护虚拟机与虚拟机监控器的交互、虚拟机隔离等操作，与外部的内核攻击进行隔离。该地址空间与 Hypervisor 处于同一特权级别。如下图.

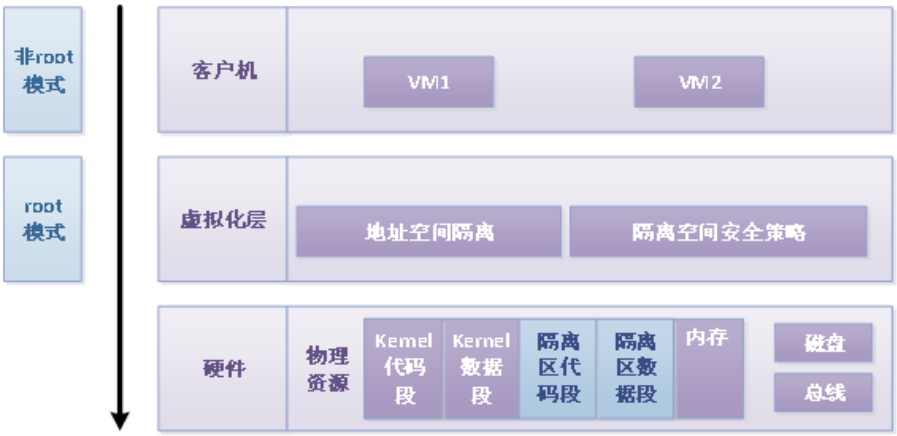

图 3. 地址空间隔离子系统架构图

2.1.2 安全切换门

为了使得虚拟机环境和虚拟机监控环境进行安全地切换，同时保证隔离地址空间的安全性，于是创建安全切换门[15]，主要包括 2 个部分，进入门和退出门。进入门是进入到隔离的地址空间，退出是从隔离的地址空间跳到原先的地址空间。进入到 secure world 的流程：保存现场、中断、CR3、新 TLB、关中断、跳到安全区。退出相反。

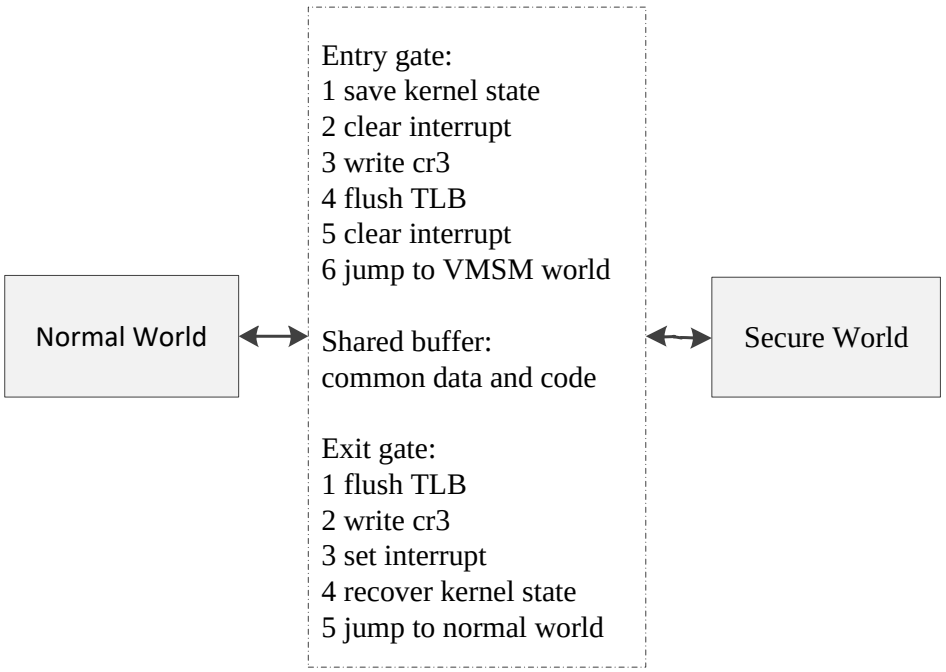

图 4. 安全切换门

2.1.3 隔离空间的安全防护

安全策略主要包括对特权寄存器的监控、对 MMU 的监控、以及防止 DMA 攻击、防止对 CR3、CR0、CR4 寄存器操作，加载恶意页表，关闭 DEP 和 SMEP 机制、防止对页表进行恶意的更改，随意访问或篡改页表内容，从而泄露敏感信息、防止通过 DMA 方式随意访问任意地址空间，从而导致敏感信息的泄露。主要是对页表的访问进行监控，同时使得隔离空间的代码和数据段在原系统中没有映射，基本原理如下图。

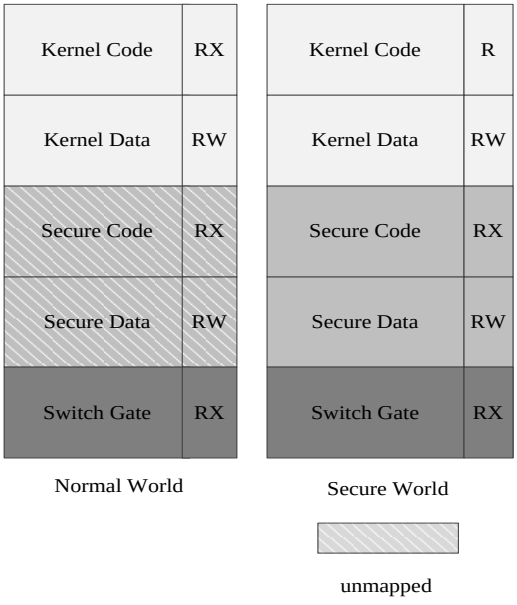

图 5. 隔离空间安全防护

2.2 VM 与 HOS 安全上下文切换

2.2.1 上下文切换

虚拟机与宿主机的切换时刻主要是虚拟机退出，在该过程中监控上下文切换的过程，安全切换保证切换过程中关键数据不被泄露，阻止跨域攻击、隐秘信息泄露等攻击。

上下文安全切换主要涉及 VMCS 结构体，其包含宿主机和客户机的各种寄存器信息，特权寄存器、指令寄存器等，如若被攻击者恶意篡改，会受到控制流攻击，关键信息泄露等。流程如图 6.

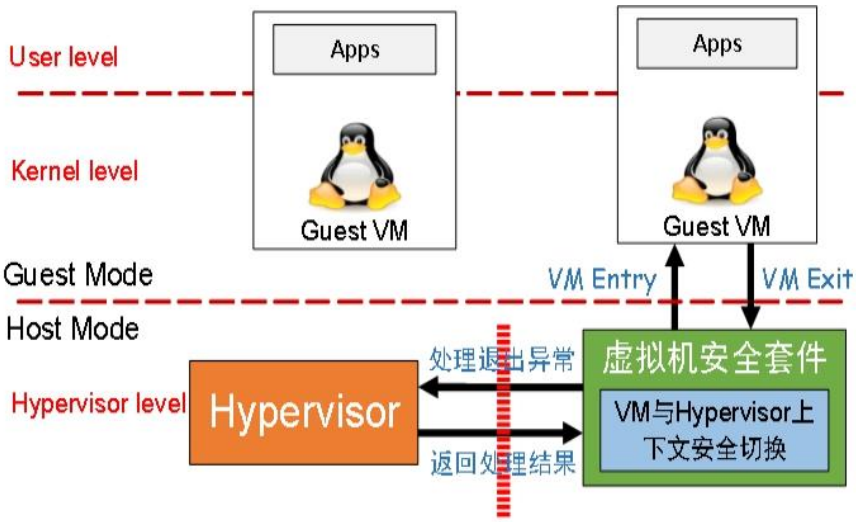

图6. 虚拟机与宿主机上下文安全切换

2.2.2 VM 退出重定向

1）VM 退出

VM 退出主要是因为 VM 中会有特权指令、敏感指令无法处理，必须交由 Hypervisor 进行处理，所以会产生虚拟机退出。

2）虚拟机退出处理流程

该过程主要涉及 VM 上下文切换、VM 退出、退出事件处理操作。

3）实施方案

待上下文切换过程结束后，跳出安全执行环境，在原系统中进行虚拟机退出事件的处理。整个的实施流程如图。

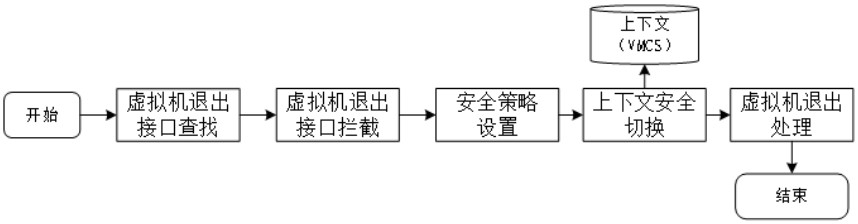

图 7. 虚拟机与宿主机安全切换流程图

2.3 虚拟机地址映射监控

虚拟机地址映射监控模块主要包括内存动态标记与跟踪和虚拟机地址映射关键数据结构（EPT）保护。涉及到虚拟机映射，虚拟机映射是指虚拟机本身的虚拟地址需要映射到宿主机物理地址。主要的实现过程需要虚拟机映射关键数据结构的协助，通过对这个映射过程进行监控，设置安全的策略保护各个虚拟机的物理资源，达到安全性的目的。基本架构如下图。

图 8. 虚拟机映射监控技术架构图

2.3.1 内存动态标记与跟踪

内存标记是通过对VM和Hypervisor使用的物理内存页进行标记，这个标记过程是在内存页分配时进行，当缺页发生时，EPT进行更新时，才会进行内存页分配。通过内存标记与跟踪的方法，将物理内存分为Hypervisor 和各个VM。内存标记的方法是使用Page-Mark表，基本内容如下表。

表3. 页标记示例表

标记	Page frame	OwnerID	UsedID	SharedID
描述	页框号	属主 ID	被使用标记	共享页标记

2.3.2 VM 地址映射监控

基本功能是监控EPT更新，在该过程中完成内存标记与跟踪，主要是因为EPT更新主要是对内存进行了分配并更新了页表。同时防止恶意加载EPT页表，篡改页表映射，防止造成重映射攻击和双映射攻击。最终的结果如图，能够保证各自的VM访问自己的内存空间，不可相互访问，防止VM跨域访问导致的信息泄露攻击。

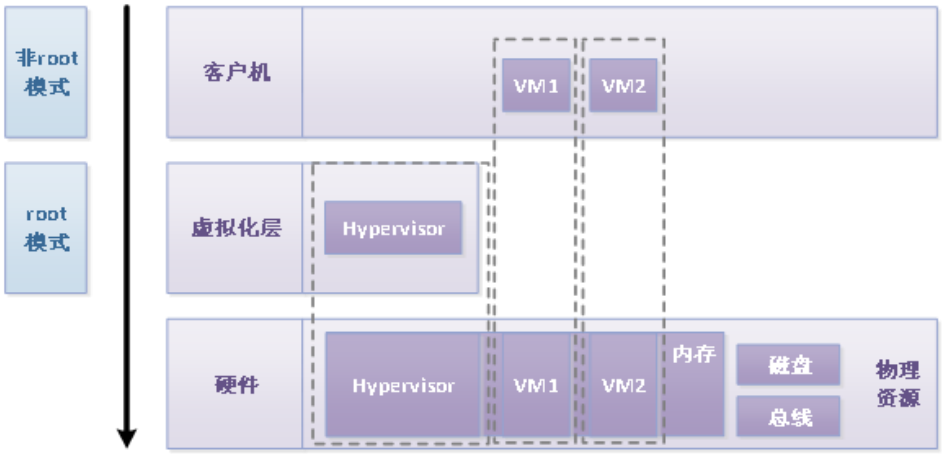

图9. 虚拟机内存隔离

2.3.3 内存共享接口设定

系统中存在 KSM 机制，KSM 机制会导致多个 VM 使用同一份物理内存页，这样就和物理页属主唯一性冲突了。

对虚拟机的保护，主要体现在对虚拟机的物理资源隔离，主要方法是内存动态标记和跟踪，各个虚拟机以及 Hypervisor 只能对自己的资源进行访问，可阻止跨域攻击和信息泄露，阻止针对内存的重、多映射攻击。同时因系统可能会开启共享内存页功能，故为了兼容该功能，添加内存页共享页接口设置模块。

首先，针对 KSM 机制可能导致原代码不兼容问题。添加共享接口处理功能来解决，在 gatefunction.h 中 ksm_page_modify()函数对物理内存页进行标记，标记 shared 属性，针对内核挂钩的函数是 ksm_do_scan()。详细的标记方法是，根据 KSM 机制的请求，随后切换到安全执行空间中对页进行合并，包含两棵树，稳定树和非稳定树，首先查找稳定树，如若存在则合并，否则在非稳定树中进行查询，存在则合并，放到稳定树中，非稳定树销毁，不存在则加入到非稳定树中。

整个的合并操作在安全隔离的地址空间进行，shared 标识标记，随后将合并后的结果返回到原系统中，切换回到原系统。

针对 KSM 机制的物理页验证方法如下：当虚拟机地址映射的时候，关键函数是 tdp_page_fault()，当映射物理页时，对于 shared 标记的物理页忽略属主检查，非 shared 标记的物理页检查属主情况，不一致则报警恶意访问。

2.4 系统初步成果

1）安全执行空间创建

主要完成隔离空间的创建，安全门的创建，同时完成该隔离空间的访问保护。能够在系统启动的时候，创建隔离空间，能够随时通过安全切换门进行环境切换，能够保证隔离空间的安全性，防止攻击者通过恶意篡改页表和特权寄存器破坏隔离空间,如图 10。

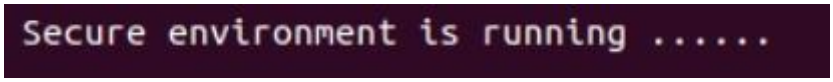

图 10.安全执行空间创建

2）系统数据初始化

主要完成内存标记与跟踪操作，虚拟机退出重定向需要的初始化操作。初始化操作涉及接口数据结构的创建等,如图 11。

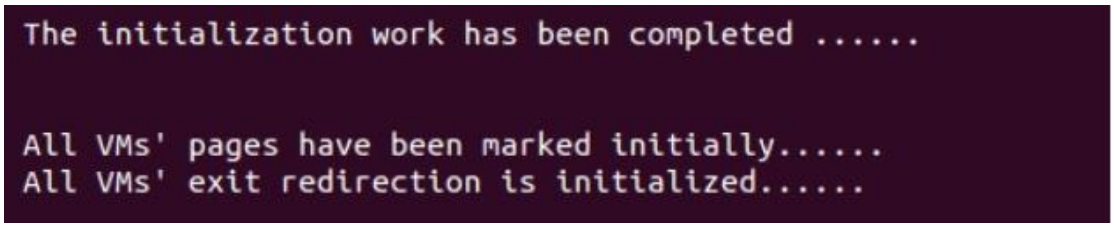

图 11. 系统初始化

接口数据结构如下：

表 4. 安全执行空间创建

名称	功能
Create_VM_Metadata	虚拟机元数据创建
Delete_VM_Metadata	虚拟机元数据删除
Alloc_Memory_to_VM	内存分配标记
Create _Isolate_space	隔离空间创建
Redirect_VM_Exit	虚拟机退出重定向
Switch_Host_Guest	Hypervisor 和 VM 之间上下文切换

3）VM 与 HOS 安全切换

主要完成在安全隔离空间中的上下文切换，同时完成虚拟机退出重定向，在该过程中主要是通过监控虚拟机退出处理函数，接下来再进行进一步的处理，将处理权限交于原系统进行处理，如图 12。

图 12. 虚拟机退出事件监控

4）VM 之间内存高强度隔离

主要在 EPT 更新时完成物理内存的分配，通过监控 tdp_page_fault 函数来完成，该函数是 EPT 更新的主要执行函数。如图 13。

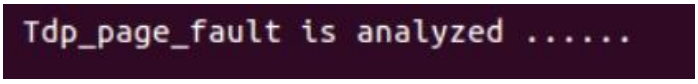

图 13. 虚拟机内存更新函数监控

3 存在的问题与解决方案

3.1 存在的问题

1）Balloon 机制导致程序崩溃

虚拟化技术常使用 Balloon 机制。该机制会导致物理内存页的属主经常发生变化，这样只在页表更新的时候去标识物理页属主信息会造成系统的崩溃，在 Balloon 机制发生作用的时候，物理页的属主发生了变化。

2）VMCS 结构体访问

在虚拟机监控模块中完成了虚拟机与宿主机上下文安全切换过程，上下文数据结构即 VMCS 数据结构。上下文切换主要是通过将 VMCS 结构体隐藏在安全隔离空间中，即该结构体的地址映射隐藏在安全隔离空间中，隔离空间之外的地址空间是无法访问，所以当原系统运行到有关上下文切换过程，所有的操作会在切换到安全隔离空间中去运行。但是有关 VMCS 结构体的访问不仅存在于上下文切换过程中，而且在 VMCS 结构的创建，随着虚拟机的销毁 VMCS 结构体被销毁过程中也存在。只考虑上下文切换，会造成系统崩溃。

3）EPT 访问

EPTP 的地址隐藏在 VMCS 中，但是直接访问 EPT 地址的函数不仅仅是上下文切换相关函数，包含虚拟机地址映射相关函数。若只考虑上下文切换相关函数，因原系统执行虚拟机地址映射，在当前的地址空间中无法访问到 EPT，导致系统崩溃。

3.2 解决方案

问题	解决方案
Balloon 机制	在 Balloon 机制发生作用时，KVM 中的“气球”会发生膨胀和压缩，当膨胀的时候，内存会被释放给 Hypervisor，此时修改物理页属主信息为 Hypervisor，当“气球”发生压缩的时候，将内存还给 VM，此时将该物理内存页的属主改为 VM 的标识 ID。
VMCS 访问	在隔离地址空间中监控 VMCS 结构体的所有地址访问函数，VMCS 创建、访问、销毁，即 VMCS_readl() VMCS_writel() VMCS_clear() free_VMCS() create_VMCS()
EPT 访问	在隔离地址空间中监控所有有关直接访问 EPT 地址的函数，EPT 创建、遍历、缺页处理、销毁，即 create_ept() mmu_spte_walk() set_vmx_cr3() tdp_page_fault() free_ept()

4 后期拟完成的研究工作及进度安排

项	工作计划	进度安排
1	系统性能测试	2018.12.1~2018.12.15
2	系统安全测试	2018.12.16~2018.12.31
3	论文撰写	2018.1~2018.4

5 已取得研究成果

成果	名称
专利	一种内存高强度隔离的安全虚拟机的实现方法及装置
论文	已投稿

6 学术论文提纲

虚拟机隔离与监控技术的研究

摘要
目录
第一章 绪论
1.1 研究背景与意义
1.1.1 虚拟化安全
1.1.2 虚拟机隔离技术
1.1.3 意义
1.2 国内外研究现状
1.2.1 基于硬件的虚拟机隔离
1.2.2 基于软件的虚拟机隔离
1.3 本文主要研究内容
第二章 HyperMI 背景技术
2.1 虚拟化技术
2.1.1 CPU 虚拟化
2.1.2 内存虚拟化
2.2 虚拟机隔离与监控技术
第三章 HyperMI 设计
3.1 威胁模型
3.2 假设

参考文献

[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)