**虚拟机安全套件设计报告**

作者： 刘文清

参与者：\_\_\_\_张坤\_\_\_\_

版本信息

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本/状态** | **作者** | **参与者** | **编制/修订历史时间** | **备注(修改单号)** |
| V1.0 | 刘文清 |  | 2018.7.19 | 新建1-9 |
| V2.0 | 刘文清 | 张坤 | 2018-7-25 | 完善2，3，4，5 |
| V3.0 | 刘文清 | 张坤 | 2018-7-26 | 完善4 |
| V4.0 | 刘文清 | 张坤 | 2018-7-26 | 完善1.2 |
| V5.0 | 刘文清 | 张坤，涂碧波 | 2018.8.22 | 完善1, 5, 8 |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

[1 文档介绍 4](#_Toc10156)

[1.1 文档目的 4](#_Toc9338)

[1.2 术语与解释 4](#_Toc8832)

[1.3 参考资料 6](#_Toc10190)

[2 需求背景及国内外发展状况 6](#_Toc17327)

[2.1 项目概述  6](#_Toc11996)

[2.2 项目背景  6](#_Toc10838)

[2.3 国内发展现状 7](#_Toc21653)

[2.4 国外发展现状 7](#_Toc17603)

[2.5 预计今后发展趋势 8](#_Toc6313)

[3 系统概述 8](#_Toc7191)

[4 子系统设计 10](#_Toc3871)

[4.1 地址空间隔离模块 10](#_Toc3581)

[4.2 虚拟机映射监控模块 12](#_Toc22960)

[4.3 虚拟机与宿主机安全切换模块 13](#_Toc27936)

[5  关键技术 15](#_Toc8422)

[5.1 地址空间隔离 15](#_Toc1604)

[5.2 虚拟机隔离 15](#_Toc13136)

[6 系统部署 16](#_Toc14122)

[7 关键数据结构 17](#_Toc25423)

[8 接口设计 19](#_Toc28082)

[8.1 内部接口设计 19](#_Toc16453)

[8.2 外部接口设计 19](#_Toc22732)

[9 安全环境设计 19](#_Toc28197)

## 文档介绍

### **文档目的**

[阐明详细设计说明的目的。]

设计说明文档为明确系统设计、安排项目规划与进度、组织系统开发和测试而编写的。该文档包括整个软件系统的结构、模块划分、模块功能、模块间的联系、模块设计，即各模块的内部功能等。清晰说明模块，后续编码过程可以按照详细设计中对功能实现的描述，直接翻译、转化为用高级编程程序设计语言编写的程序。

主要针对3大功能模块进行详细阐述，地址空间隔离模块、虚拟机和Hypervisor交互监控模块、虚拟机映射模块。

### **术语与解释**

[本小节应提供正确理解此详细设计说明书所需的全部术语的定义、首字母缩写词和缩略语。]

* 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。

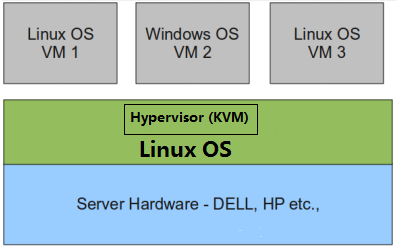


图1. KVM架构图

* HOS 宿主机
* GOS 客户机
* 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。

* 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)。

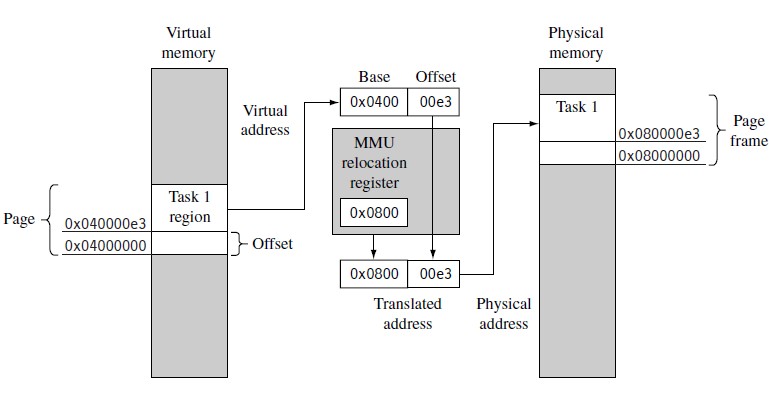


图2. MMU工作机制图

* DEP

数据执行保护技术（Data Execution Prevention，以下简称 DEP）是 win xp 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。

* 虚拟机退出与进入

支持VT-x技术的CPU有两种运行模式，VMX模式和非VMX模式。VMX模式就是支持VT技术的主要模式，而非VMX模式则与不支持VT-x技术的CPU运行模式完全相同。而VMX模式又分为两种操作模式，VMX root模式和为VMX non-root模式，VMX root模式是提供给VMM使用的，在这种模式下可以调用VMX指令集，VMM用以创建和管理虚拟机。而VMX non-root模式就是虚拟机运行的模式，这种模式不支持VMX指令集。从root模式到non root模式是虚拟机退出，从non root模式到root模式是虚拟机进入。

* 多映射攻击

一种攻击手段，指多个虚拟地址映射到一个物理地址，即可以通过多个虚拟地址来访问到某一物理内存页。这种攻击会导致敏感信息泄露。

* 重映射攻击

一种恶意攻击手段，指被释放的物理页b，内容依旧存在，当其余再次映射到某个虚拟地址a时，可以通过访问a得到物理页b上的内容。这种攻击可以泄露敏感数据信息。类似于指针UFA攻击。

* QEMU

专门用来模拟各种器件的应用程序，其中Qemu-kvm是Qemu为了模拟提供给KVM使用的器件，是Qemu的一个分析，这样用户才能在用户空间使用基于KVM的VM。

* Libvirt

Libvirt 提供一种虚拟机监控程序不可知的 API 来安全管理运行于主机上的客户机操作系统。Libvirt本身不是一种工具，它是一种可以建立工具来管理客户机操作系统的API。Libvirt本身构建于一种抽象的概念之上。它为受支持的虚拟机监控程序实现的常用功能提供通用的API。

### **参考资料**

[本小节应完整列出此总体设计说明书中其他部分所引用的任何文档。]

《计算机软件开发规范（GB8566-88）》

《系统虚拟化》：英特尔开源软件技术中心. 系统虚拟化[M]. 清华大学出版社, 2009.

《软件工程导论》：摩尔, R) 著, 马振晗,等. 软件工程导论: 第2版[M]. 清华大学出版社, 2008.

《软件工程技术概论》：朱三元, 钱乐秋, 宿为民. 软件工程技术概论[J]. 2002.

## 需求背景及国内外发展状况

### **项目概述**

当前云计算发展的速度越来越快，底层的虚拟化技术为云计算技术提供着不可或缺的功能。虚拟机的安全性与底层虚拟化层的安全性密切相关，为了保护上层的虚拟机的安全，从整个虚拟化底层提供相应的防护措施。

### **项目背景**

当前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（Hypervisor）进行统一的分配和管理。其中虚拟化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。然而底层的虚拟化层一旦受到黑客攻击后，上层的虚拟机的安全性也受到严重的威胁。所以提出虚拟机安全套件，用来保证在非可信虚拟化执行环境中保护上层虚拟机运行时安全。

### **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。

### **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。

### **预计今后发展趋势**

云计算技术发展越来越快，依据云计算的产品也越来越多，各个数据处理平台会大量用到云计算技术，底层的虚拟化技术也会相应地发展。对于虚拟机敏感数据的保护要求也越来越高。无论是从保护虚拟机，保护底层的Hypervisor，增强Hypervisor的完整性等，最终的目的是保护上层虚拟机上用户敏感数据的安全。

## 系统概述

画出系统的总体架构图，详细介绍主要的功能模块以及控制流程。

图3. 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，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

## 子系统设计

首先介绍各个子系统之间的交互关系。

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5. 子系统交互关系图

### **地址空间隔离模块**

从技术架构和实现流程等方面描述该子系统的设计，需要画出子系统的架构图以及界面设计等。

#### 4.1.1技术架构

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。

为了保证安全性，隔离的地址空间中还包含了安全区域的代码段和数据段，但这些在普通环境中并不存在，如下图：



图6. 地址空间隔离子系统架构图

#### 4.1.2实现流程

* 创建隔离的地址空间

通过创建新页表实现新的地址空间。该页表包含原Hypervisor的内核地址空间之外，还包括安全区域（即原系统页表不包含的安全区代码段和数据段）。

* 创建环境切换门

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，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。

* 实现安全策略，保证环境切换的安全性

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。



图7. 隔离地址空间实现流程图

### **虚拟机映射监控模块**

#### 4.2.1技术架构

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。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。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。同时需要对关键数据结构（EPT）进行保护，主要方法是隐藏其地址与安全隔离的地址空间。



图8. 虚拟机映射监控技术架构图

#### 4.2.2实现流程

* 虚拟机映射接口查找

查找每个VM对应的地址映射表。即根据VM，查找对应的EPT。

* 虚拟机映射接口拦截

从代码实现的角度拦截地址映射接口，从而实现后续的内存页动态标记、多、重映射监控方案。

* 虚拟机映射过程监控

监控虚拟机映射过程，包括EPT创建、更新、遍历和销毁。

* 虚拟机映射过程安全防护

在地址映射更新的过程中，只对当前VM的物理内存进行操作，目的是防御多、重映射，防御虚拟机跨域攻击。



图9. 虚拟机映射监控实现流程图

### **虚拟机与宿主机安全切换模块**

#### 4.3.1技术架构

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。

图10. 虚拟机与宿主机安全切换技术架构图

#### 4.3.2实现流程

* 虚拟机与宿主机切换接口查找

查找切换接口相关的关键数据结构，一般指VMCS结构体。

* 虚拟机与宿主机切换接口拦截

接口拦截，即拦截虚拟机退出函数，以便进行后续的监控。

* 虚拟机与宿主机切换监控

监控切换过程，主要是在虚拟机退出和进入时刻进行，选择退出时进行监控。

图11. 虚拟机与宿主机安全切换实现流程图

## 关键技术

需要详细介绍每个关键技术的原理、突破途径和方法。问题一定要找准，描述清楚，分析准确。

### **地址空间隔离**

#### 5.1.1问题描述

地址空间隔离，可以提供一些安全工具的执行环境，该目标实现的方法有多种，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。

#### 5.1.2突破途径及方法

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。

技术方法主要体现在创建另一份内核页表去实现，在代码copy\_pagetable.h中，cp\_old\_pg\_half()函数创建了一份内核页表，此外 update\_pagetable()用于更新这份内核页表。该页表是64bit系统的4级页表，同时考虑系统中存在的大页情况。

保证环境安全切换的关键步骤是使用原子操作进行安全切换，保证切换过程 不被中断，隔离地址空间的入口地址不被泄露。

为了保证新地址空间运行的安全性，需对页表的访问进行监控，主要防止恶意篡改页表，直接访问安全区域等。通过hook挂钩的方式监控do\_page\_fault()函数，防止恶意改页表；监控特权寄存器的写操作（CR3、CR0、CR4），防止恶意加载页表，恶意关闭DEP机制，恶意关闭SMEP机制，从而避免控制流攻击，页表篡改攻击等。

代码相关详细接口请查看第8节——内部接口。

### **虚拟机隔离**

#### 5.2.1问题描述

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，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。系统中存在KSM机制和Ballon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；Ballon机制中，经常会有物理内存页属主发生变化的情况，那么应该随属主变化而更改物理页的标签信息，否则会造成系统的崩溃。

#### 5.2.2突破途径及方法

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。



图12. 虚拟机隔离架构图

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。

其次，针对Ballon机制可能导致属主经常发生变化，需要及时更改物理页的属主信息，在tdp\_page\_fault()函数中对物理页进行属主标记，当ballon机制发生作用时，hook该函数，更改物理页的属主信息。

## 系统部署

描述整个系统的部署情况，需要画出部署图。



图13. 系统部署图

如上图所示，虚拟机安全套件是部署在服务器端，部署在服务器上。这些服务器为各个终端提供服务。

## 关键数据结构

**客户机物理地址空间**（GPA）

对一个操作系统来说，内存是物理地址从0开始的连续的地址空间。在虚拟化环境下，真正拥有物理内存的是虚拟机监控器VMM。物理内存只有一份，VMM需要在宿主机上为每个客户机操作系统模拟出可以当作物理内存一样使用的虚拟内存。VMM模拟了一层新的地址空间：客户机物理地址空间。客户机把它认为是物理地址空间，但它不是宿主机最终的物理地址空间。



图14. 客户机虚拟地址翻译过程

**EPT**

由于引入了客户机物理地址空间，内存地址转换过程变为：从客户机虚拟地址GVA转换到客户机物理地址GPA；再从客户机物理地址GPA转换到宿主机物理地址HPA。其中，GVA到GPA的转换由客户机操作系统决定，客户机操作系统通过VMCS中Guest状态域CR3寄存器指向的页表来指定；GPA到HPA的转换是由VMM决定，VMM在将物理内存分配给客户机时确定GPA到HPA的转换，这个映射关系往往由VMM中的内部数据结构记录，VMM为每个虚拟机动态地维护了一张客户机物理地址与宿主机物理地址映射表，基于硬件创建的表叫做扩展页表（EPT），如下两张图分别是EPT创建代码和原理图。

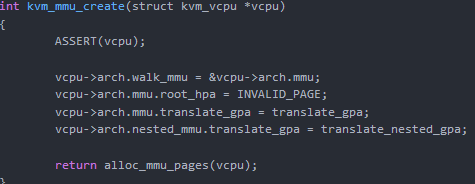


图15. EPT创建函数说明

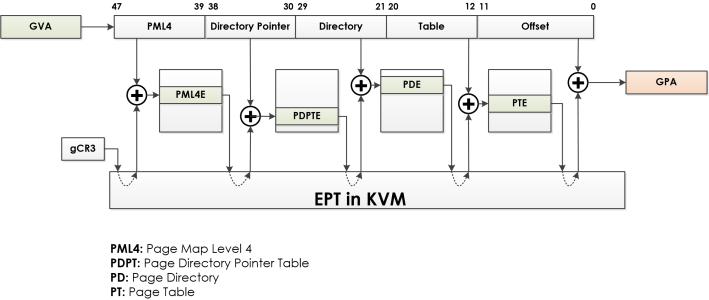


图16. EPT翻译原理图

**内存页标记**

为了保证各个虚拟机之间的隔离性，以及虚拟机与Hypervisor之间的隔离性，创建内存页标记，用来跟踪每个物理内存页的使用情况，如下图。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 标记 | VMID | PageID | UsedID | SharedID |
| 描述 | 对应的属主VM | 物理页ID | 是否被使用 | 是否被共享 |

表1. 内存页标记

## 接口设计

### **内部接口设计**

内部接口主要包括模块或软件构件间的接口。

|  |  |
| --- | --- |
| 模块 | 接口 |
| 隔离的地址空间的接口 | 该地址空间的入口 |

表2. 内部接口设计

关键函数说明如下：

|  |  |  |
| --- | --- | --- |
| 文件 | 函数 | 功能说明 |
| Copy\_pg.h | cp\_old\_pg\_half() | 完成新内核页表创建，创建地址隔离空间 |
| Copy\_pg.h | update\_pagetable() | 更新新页表的内容 |
| Find.h | Find() | 查找页表中某虚拟地址对应的物理地址 |
| Gatefuntion.h | New\_xxx() | 若B函数覆盖函数A。此处是所有B类Hook函数的实现 |
| Gatefunction.h | DISABLE\_INT\_FIRST();  writecr3(\_\_pa(pgd\_addr\_cp\_gate));  DISABLE\_INT\_SECOND(); | 安全切换  进入门代码 |
| Set\_hook.h | Set\_hook\_xxx()等函数的实现 | 若函数B通过方法C覆盖A。此处是钩子函数C的前期准备 |
| Gatefunction.h | ENABLE\_INT();  \_\_flush\_tlb\_all();  writecr3(oldcr3); | 安全切换  退出门代码 |
| Hooker.c | Set\_hook\_xxx()函数的调用 | 各个C类函数总调用接口文件 |

表3. 关键函数功能详细说明

系统架构和代码基本思路如下：

关于地址空间隔离。通过copy\_pg.h中的页表创建函数创建新的内核页表，随后包含update\_pagetable()来更新页表，以防部分信息已过时，影响系统代码运行。通过这样的方式创建安全的隔离地址空间；对于空间之间的相互切换，使用退出和进入门两种方式，查看上表；对于新的地址空间要确保该地址空间的安全性，主要通过对特权寄存器和页表访问操作进行挂钩监控，函数涉及的文件有hooker.c，gatefunction.h, set\_hook.h等。

关于虚拟机监控。通过对VMCS和EPT的监控，主要方式是通过挂钩监控与VMCS和EPT相关的系统函数，对其进行更改。这个过程中监控VMCS保证虚拟机和宿主机的上下文切换安全，监控EPT相关操作确保EPT的准确性及安全性。

关于虚拟机隔离。主要目的是阻止多映射攻击和重映射攻击，在该过程中主要通过标记物理页属主来完成攻击阻止。首先，对于多映射，EPT涉及tdp\_page\_fault()等，在客户机发生缺页时，引发tdp\_page\_fault()，分配物理内存页，此时对于第一次被分配的物理页设置标记，在这个过程中会涉及到KSM和Ballon机制引发的属主不一致性冲突，为了解决这些冲突，有关KSM和Ballon的函数有相关标志和属主信息处理，这样完成了防止内存物理页的多映射攻击阻止；其次，对于重映射攻击，类似指针UAF攻击思想，当物理页被释放时，清除信息，主要关注于shrink\_page\_list()函数。

### **外部接口设计**

主要涉及2类接口设计，系统与用户交互的接口和软件与其他软硬件系统之间的接口。

|  |  |
| --- | --- |
| 模块 | 接口 |
| 系统与用户交互的接口 | 使用命令行，以LKM模式加载security\_com.ko模块，即系统运行开始 |
| 软件与其他软硬件系统之间的接口 | 根据实际创建虚拟机的数量配置对应大小的内存和磁盘，必要的网络设备和防火墙 |

表4. 外部接口设计

## 安全环境设计

主要说明为系统的安全运转，需要提供的外部环境。

**系统启动安全**

确保系统在启动时的安全性，从而为设计的系统提供安全的保证。启动过程的安全性是需要硬件来提供保证，比如安全的总线，内存，硬盘等。

**物理环境安全**

保证物理环境的安全性。需要提供对设备、机房等的安全保障。