**\_\_\_\_\_虚拟机安全套件\_\_\_\_测试报告**

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 |  | 2019.1.2 | 文档书写 |
| V2.0 | 刘文清 | 张坤 | 2019.3.4 | 功能测试 |
| V3.0 | 刘文清 | 张坤 | 2019.3.6 | 功能测试 |
| V4.0 | 刘文清 | 张坤 | 2019.4.2 | 性能、功能、稳定性测试 |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

[1 概述 4](#_Toc2757545)

[1.1 编写目的 4](#_Toc2757546)

[1.2 参考资料 4](#_Toc2757547)

[2 测试环境 4](#_Toc2757548)

[3 测试结果 5](#_Toc2757549)

[3.1 功能测试 5](#_Toc2757550)

[3.2 性能测试 13](#_Toc2757551)

[3.3 稳定性测试 16](#_Toc2757552)

[3.4 异常情况测试 20](#_Toc2757553)

[4 评价 22](#_Toc2757554)

[4.1 软件能力 22](#_Toc2757555)

[4.2 缺陷与限制 22](#_Toc2757556)

[4.3 建议 22](#_Toc2757557)

[4.4 测试结论 22](#_Toc2757558)

## 概述

### 编写目的

[阐明编写测试大纲的目的及其指明读者对象。]

本测试报告目的在于总结测试阶段的测试情况及分析测试结构，描述系统

是否符合用户需求，是否达到用户预期的功能目标，并对测试质量进行分析。

测试报告参考文档提供给用户、测试人员、开发人员、项目管理者，其它管理人员和需要阅读本报告的其余人。

### 参考资料

[本小节应完整列出此测试大纲中其他部分所引用的任何文档。]

《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.

《需求说明书》：虚拟机安全套件需求说明书.

《设计说明书》：虚拟机安全套件设计说明书.

## 测试环境

测试需要服务器、多台客户虚拟机、VNC viewer。

|  |  |
| --- | --- |
| 资源名称/类型 | 配置 |
| 宿主机KVM | 服务器，主频2.0GHZ，硬盘1T，内存16G，linux-4.4.1，ubutnu-14.05 |
| 客户机 | 内存8G，磁盘大小30G，系统版本ubuntu-12.04.5 |
| VNC viewer | 6.17.1113版本 |
| Libvirt | 3.0.0版本 |
| Qemu | 2.6版本 |

## 测试结果

### 功能测试

功能测试涉及四个部分：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、动态内存标记与跟踪。需要对着四个功能模块进行测试，达到一定的功能需求。

#### 3.1.1测试用例

【主要说明测试的步骤及操作】

主要对四个功能进行测试。

* 地址空间隔离功能

地址空间隔离功能测试方案：在安全隔离地址空间中设置新的变量variety，在其普通地址空间中去访问该变量。访问不成功表明该功能成功实现。该功能主要是保证在新的安全隔离地址空间（安全执行环境）中的变量不能被运行在普通地址空间的进程访问到，保证空间的隔离性和安全性。

|  |
| --- |
| //(add\_isolation\_test.c) address isolation space function test  #include <linux/init.h>  #include <linux/module.h>  #include<linux/kernel.h>  Static int \_\_init my\_init(void)  {  Unsigned long pgd\_addr\_gate = get\_pgd\_cp();  Unsigned long oldcr3 = read\_cr3();  DISABLE\_INT\_FIRST();  writecr3(\_\_pa(pgd\_addr\_cp\_gate));  \_\_flush\_tlb\_all();  DISABLE\_INT\_SECOND();  unsigned long testaddress = get\_zeroed\_page(\_\_GFP\_HIGHMEM);  ENABLE\_INT();  \_\_flush\_tlb\_all();  writecr3(oldcr3);  \_\_flush\_tlb\_all();  Printk(“test addrss is %lx\n,testaddress”);  }  Static void \_\_exit my\_exit(void)  {  Printk(“This function (address space isolation) test is done!”);  }  Module\_init(my\_init);  Module\_exit(my\_exit);  MODULE\_LICENSE(“GPL”); |

|  |
| --- |
| 1. 1）如果系统崩溃，出现访问错误（page fault），即地址空间隔离功能实现了隔离性和安全性。   2）如果打印结果是：“ test address is ……”则表明没有实现地址空间隔离功能。 |

* 虚拟机与宿主机交互监控功能

虚拟机与宿主机交互监控功能测试方案：该功能主要是防止普通地址空间中的进程恶意篡改VMCS结构体，进而攻击客户机。因此，该测试方案就是在加载虚拟机安全套件与否，测试直接访问VMCS结构体是否成功。在加载了套件系统之后，普通地址空间中的进程访问VMCS结构体时，会出现禁止访问VMCS结构体的警告。

|  |
| --- |
| 篡改VMCS测试步骤 |
| 1 被hooked的函数—vmcs\_load，hooked后跳转到函数—vmcs\_load\_attack |
| 2 复制hook指令(mov vmcs\_load\_attack,eax jmp eax ) |
| 3 覆盖vmcs\_load函数的前8 bytes |
| 4 在hooked后的函数中访问实参 \*(vmcs\_signal->data) |
| 5 实现攻击 |



|  |
| --- |
| //实施VMCS篡改测试的关键代码  //vmx.c 内核更改部分  void vmcs\_load\_attack(struct vmcs \*vmcs\_signal)  {  vmcs\_load\_part(vmcs\_signal);  printk(“The content of VMCS is %lx\n!”,\*(vmcs\_signal->data));  } |

|  |
| --- |
| //VMCS\_attack.c  攻击实施,调用Set\_hook\_VMCS(vmcs\_load , vmcs\_load\_attack) |

|  |
| --- |
| 1. 未加载虚拟机安全套件，会打印\*(vmcs\_signal->data)（VMCS内容）的值。 2. 加载虚拟机安全套件后，不会打印\*(vmcs\_signal->data)（VMCS内容）的值，并出现访问错误（page fault）警告，无法在普通地址空间中访问VMCS。 |

* 虚拟机地址映射监控功能

虚拟机地址映射监控功能测试方案：模拟本地攻击者行为，直接篡改EPT，随后根据篡改的EPT内容，进行多重映射攻击。如若攻击成功，表明该功能没有实现。

|  |
| --- |
| 篡改EPT攻击步骤 |
| 1 被hooked的函数— mmu\_spte\_walk，hooked后跳转到函数— mmu\_spte\_walk\_attack |
| 2 复制hook指令(mov mmu\_spte\_walk\_attack,eax jmp eax ) |
| 3 覆盖mmu\_spte\_walk函数的前8 bytes |
| 4 在hooked后的函数中访问实参vcpu-> arch.mmu.root\_level |
| 5 实现攻击 |



|  |
| --- |
| //实施EPT篡改攻击的关键代码  //mmu\_audit.c 内核更改部分  void mmu\_spte\_walk\_attack(struct kvm\_vcpu \* vcpu, inspect\_spte fn)  {  mmu\_spte\_walk\_part(vcpu,fn);  printk(“The address of EPT is %lx\n!”,vcpu->arch.mmu.root\_level);  } |

|  |
| --- |
| //EPT\_attack.c  攻击实施,调用Set\_hook\_EPT(mmu\_spte\_walk，mmu\_spte\_walk\_attack) |

|  |
| --- |
| 3）未加载虚拟机安全套件，会打印\*(vmcs\_signal->data)（VMCS内容）的值。   1. 4）加载虚拟机安全套件后，不会打印vcpu->arch.mmu.root\_level（EPT）的值，并出现访问错误（page fault）警告，无法在普通地址空间中访问VMCS。 |

* 动态标记与跟踪功能

动态内存标记与跟踪功能测试方案：保证内存可以得到正确的标记；并能够对KSM机制、Balloon机制的有一定的适应性，当这些机制开启后，内存复用的现象会发生，为了测试内存的标记随着内部复用动态变化，设计了测试方案。1）系统未开启KSM、Balloon机制时，打印各个内存页物理地址和页框号、SharedID信息。2）当系统开启这两个机制时，筛选ShareID标志位为1的内存页，并打印被复用的内存页物理地址和页框号信息。

#### 3.1.2预期结果

* 地址空间隔离功能测试：可以阻止普通地址空间的进程访问新的地址空间（安全执行环境）中的变量variety。
* 虚拟机与宿主机交互监控功能测试：可以阻止普通地址空间中的进程访问VMCS结构体。
* 虚拟机地址映射监控功能测试：通过阻止恶意篡改EPT操作，从而能够阻止随后的多重映射攻击实施。
* 内存动态标记与跟踪功能测试：通过在未开启内存复用功能（KSM、Balloon机制）时，打印内存页ID号和属主信息；以及通过在开启复用功能时，打印内存页ID 号和sharedID信息，来验证该功能的可行性。如若在内存复用机制开启后，存在部分内存页SharedID的值为1，则表明内存动态标记与跟踪功能被实现。

#### 3.1.3实测结果

四种功能通过测试结果如下，测试的过程完全相同，因将代码集体合成到一个代码中secureVM\_test.c中，随后直接执行命令make && insmod secureVM\_test.ko，开启虚拟机（virsh start vm1），查看日志(dmesg)。

1. 地址空间隔离功能测试：

地址空间隔离创建于HyperMI world（安全执行环境）中的变量不能被非HyperMI world的进程访问，访问请求被阻止；隔离成功。

1. 虚拟机与宿主机交互监控功能测试：

未加载虚拟机安全套件时，打印VMCS的值：



加载虚拟机安全套件后，无法打印VMCS的值，提出警告：



VMCS中有关虚拟机的信息不能在HyperMI world外部访问，外部访问会被阻止；交互监控成功。

1. 虚拟机地址映射监控功能测试：

未加载虚拟机安全套件时，打印EPTP的值：



加载虚拟机安全套件后，无法打印EPTP的值，提出警告：



阻止非HyperMI world访问HyperMI world中的EPTP，从而阻止攻击者更改EPT，阻止了接下来的多重映射攻击。

1. 内存动态标记与跟踪功能：

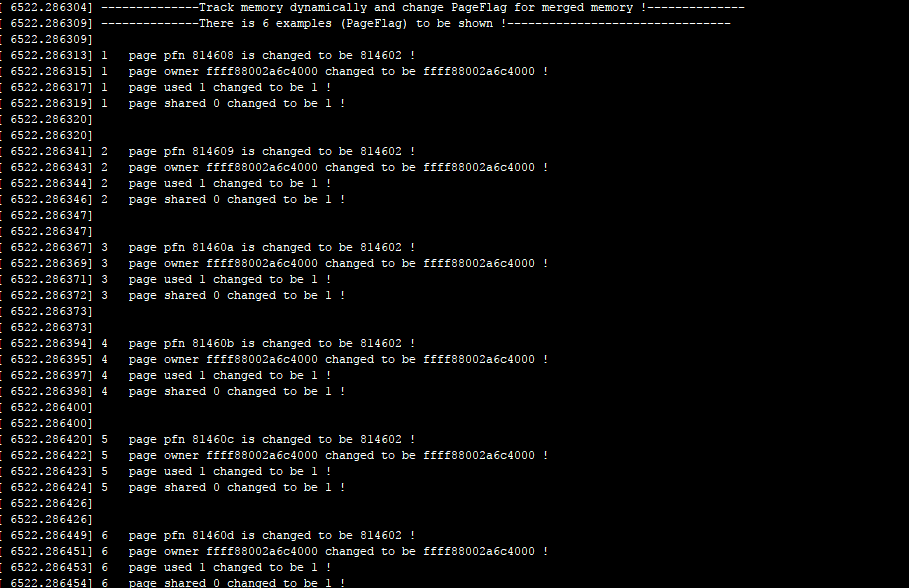
内存复用机制不开启时，打印内存使用信息（页物理地址，页框号以及SharedID标志位）。

|  |
| --- |
| [ 303.132040] The pages are as follow:  [ 303.132051] page addr is : b64fb  [ 303.132063] current\_num\_pfn is : 3255, SharedID is 0.  [ 303.132099] page addr is : 2b0a7b  [ 303.132107] current\_num\_pfn is : 3256, SharedID is 0.  [ 303.132142] page addr is : 2b0c76  [ 303.132151] current\_num\_pfn is : 3257, SharedID is 0.  [ 303.132184] page addr is : 2b0cc7  [ 303.132193] current\_num\_pfn is : 3258 , SharedID is 0.  [ 303.132246] page addr is : 2b0ccb  [ 303.132255] current\_num\_pfn is : 3259, SharedID is 0.  [ 303.132286] page addr is : 2b0c67  [ 303.132297] current\_num\_pfn is : 3260, SharedID is 0. |

内存复用机制开启后，打印内存使用信息（页物理地址和页框号）。

|  |
| --- |
| [ 303.132040] The page whose SharedID is 1 as follow:  [ 303.132051] page addr is : b64fb  [ 303.132063] current\_num\_pfn is : 3255  [ 303.132099] page addr is : 2b0a7b  [ 303.132107] current\_num\_pfn is : 3256  [ 303.132142] page addr is : 2b0c76  [ 303.132151] current\_num\_pfn is : 3257  [ 303.132184] page addr is : 2b0cc7  [ 303.132193] current\_num\_pfn is : 3258  [ 303.132246] page addr is : 2b0ccb  [ 303.132255] current\_num\_pfn is : 3259  [ 303.132286] page addr is : 2b0c67  [ 303.132297] current\_num\_pfn is : 3260 |

再次试验，打印出前6项被合并的页。分别打印出各项中原物理页的PageFlag中的各数据和合并到的物理页的PageFlag，如下图所示。



测试结果总结如下：



### 性能测试

系统包含三大模块，安全执行环境的创建，虚拟机与Hypervisor交互关键数据监控，以及虚拟机内存隔离。安全执行环境的创建理论上并不需要大量的操作，也不会带来大量的性能开销，但是环境切换的过程会因为挂钩函数数量的多少，挂钩函数的使用频率受到影响。虚拟机与Hypervisor交互关键数据的监控模块中由于VMCS和EPT是被存储在HW中，访问这两个关键数据结构体的操作越频繁，那么会导致过多的环境切换，从而带来一定性能开销。同时访问VMCS和EPT的操作主要是虚拟机进入和退出，虚拟机内存缺页访问操作。虚拟机内存隔离功能模块会在物理页被分配时访问Page Mark 表和VM Mark表，过多的内存分配会导致过度地访问这两个表，从而带来一定的性能开销。

根据上述分析，对环境切换、虚拟机与Hypervisor的安全切换、内存分配三个角度进行性能分析。由于这三种功能无法被直接测试，本文通过基准测试集和虚拟机启动时间来确定HyperMI在这三个功能上引入的性能开销。

#### 3.2.1测试用例

【主要说明测试的步骤及操作】

三种功能的性能开销平均小于原系统的10%。使用微观测试工具集，SPEC CPU和bonnie++进行测试。

测试步骤：

1. 使用SPEC CPU 2006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。 |

3)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。

1. 使用bonnie++测试工具测50次，将结果进行平均化。
2. 在test 目录下执行，针对内存大小为1000M的系统。执行该命令

./bonnie++ -d /test -s 2000 -u root

1. 在terminal得到数据信息并整理。

·对虚拟机进行启动和关闭测试。

·对内存带宽进行测试，使用stream工具，使用5台虚拟机，在宿主机和虚拟机上都运行stream工具，测试宿主机上的内存带宽。

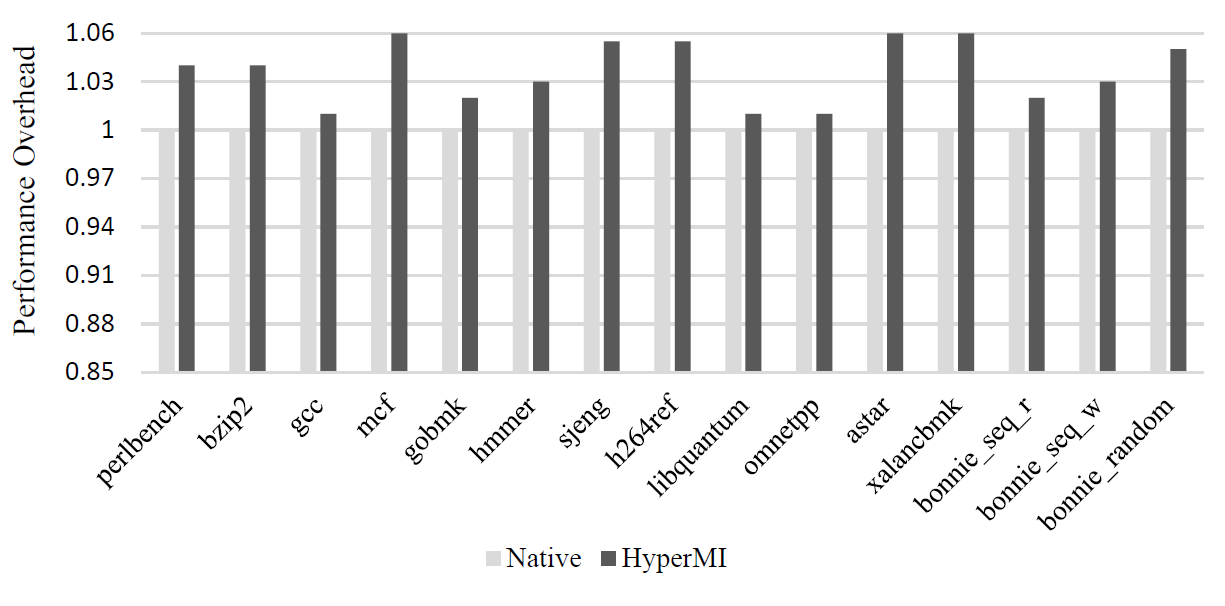
·使用lmbench工具进行基准测试。 使用5台虚拟机，在虚拟机上都运行负载，测试宿主机上运行lmbench的各种指标。

#### 3.2.2预期结果

综上，预期有关内存和IO操作的性能开销会提升，其中有关内存的性能开销较大，有关IO的性能开销较小。

#### 3.2.3实测结果

A．基准测试



基准测试测评结果

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。

分析上图得知内存操作监控较多、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。根据上图的测试结果，最高的性能开销是原KVM系统的1.06倍，主要是Mcf, astart, xalancbmk，这三种测试集主要的功能是大量分配内存，从而在EPT更新时访问page mark表，验证地址映射的正确性；访问Page Mark会导致环境切换；EPT更新会导致虚拟机退出/进入,并频繁访问VMCS这三个部分发生性能损耗。总体的性能是低于10%，满足性能需求。上图中的后三项数据是bonnie++的测试结果，分别进行了顺序读、顺序写以及随机写操作测试，这些测试是预先分配内存再进行读或者写操作。根据结果分析，性能开销不大于无HyperMI组件的KVM系统的5%，性能开销相对较低，低于原系统的10%，可以接受。

**B. 虚拟机启动关闭时间测试**

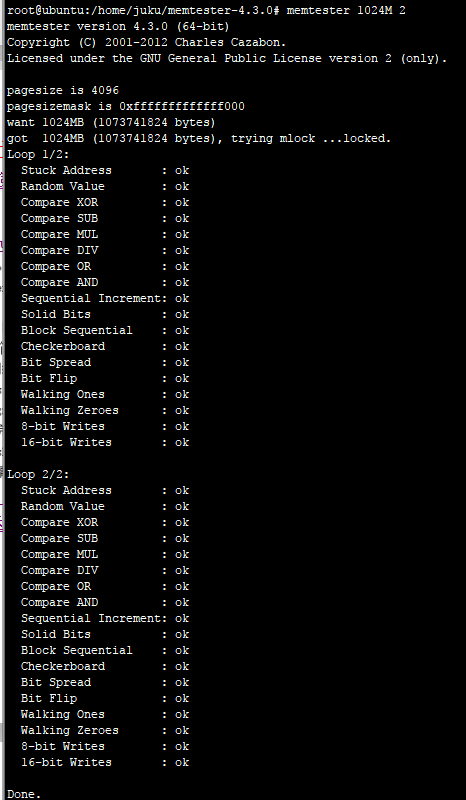
测试结果如下：

|  |  |  |
| --- | --- | --- |
|  | 启动（s） | 关闭（s） |
| 不启动HyperMI | 11.79 | 1.75 |
| 启动HyperMI | 12.97 | 1.89 |
| 效率 | 1.1 | 1.08 |

通过程序运行过程分析，当宿主机安全启动后，HyperMI会创建新的一套页表完成安全执行环境的创建。随后，创建虚拟机，在启动虚拟机的过程中，会分配大量的内存，引发虚拟机退出和虚拟机进入操作；同时在内存分配，页表更新的过程中访问Page Mark 表和VM Mark表，验证EPT更新的正确性和页更新的正确性；由于访问在虚拟机退出/进入和页表分配中，访问了位于HW中的VMCS和EPT，导致大量的环境切换，引发一定开销。综上，虚拟机启动的过程中会引发大量的环境切换、Page验证、虚拟机与Hypervisor的交互操作。根据上述测试结果分析，当使用HyperMI时，给虚拟机启动和关闭带来的时间开销分别是1.1倍和1.08倍，相对较小，这种开销是可以接受的。

1. 内存带宽memtester测试

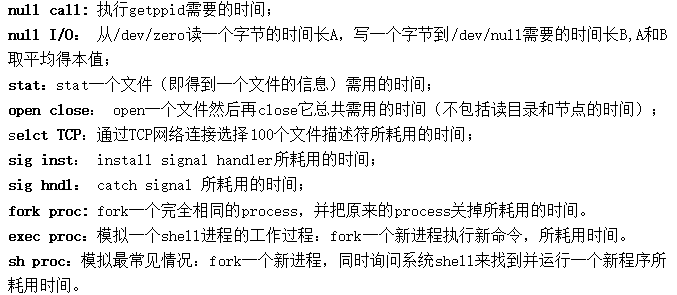
通过使用memtester测试工具测试内存带宽，申请1024M内存进行内存压力测试2次，测试指令和测试结果如图，表明在开启虚拟机安全套件系统后，宿主机可以通过内存压力测试。



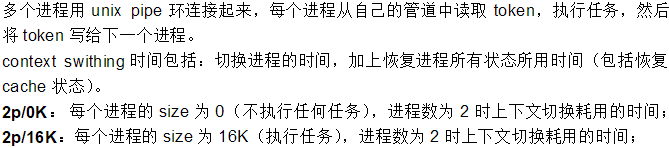
1. Lmbench基准测试

通过两次测试，第一次是虚拟机有负载运行，第二次是虚拟机没有负载运行，测试的结果如下。

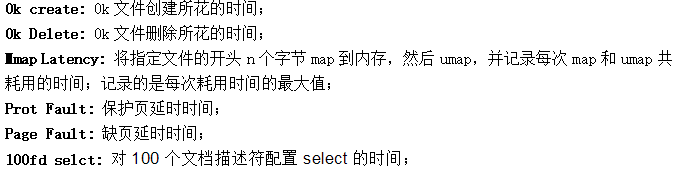
* 处理器上运行null系统调用、空的IO访问等，参数解释如下。



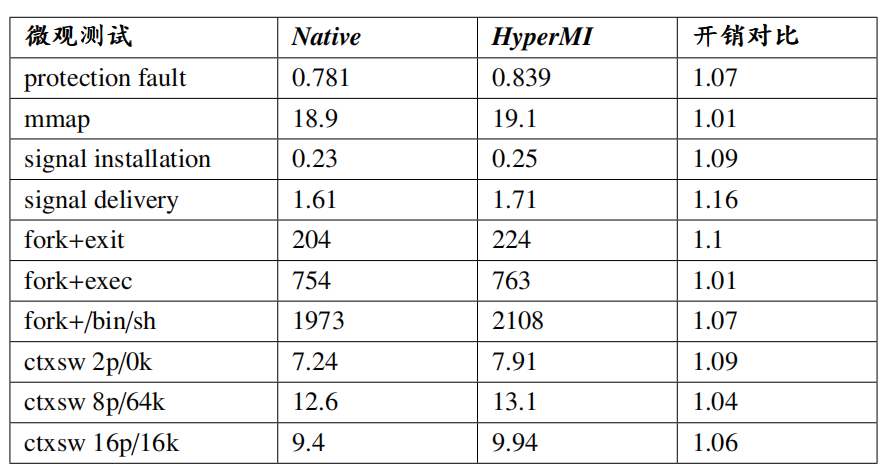
* 上下文切换中运行各个不同负载的进程，参数解释如下。



* 文档、内存延迟针对物理页做了一些测试，参数解释如下。



结合下图中的测试数据分析，在内存延迟操作上不同，其余在文件、中断上下文上、数值计算上、通信上的延迟相差不大。



|  |  |  |  |
| --- | --- | --- | --- |
| 测试项目 | 测试工具 | 测试方法 | 通过测试(Y/N) |
| 宏观基准测试 | SpecCPU2006 | 相关使用操作测试 | Y |
| 内存压力测试 | memtester | 每台虚拟机测试2G内存使用 | Y |
| 微观基准测试 | Lmbench | 有关进程的测试 | Y |
| 虚拟机启动关闭时间 | 代码测试 | 启动关闭与原KVM的比较 | Y |

### 稳定性测试

#### 3.3.1测试用例

【主要说明测试的步骤及操作】

稳定性测试主要涉及系统的稳定性，本系统的稳定性是指系统运行中不会崩溃，主要针对高频率地系统加载和卸载、在虚拟机安全套件加载前后服务器本身承受虚拟机开启的最大数量是否发生变化、SecureVM加载前后宿主机服务器上虚拟机运行的最长时间是否发生变化。

测试要求：

1. 高频率加卸载虚拟机安全套件系统和虚拟机不会导致自身和宿主机系统崩溃。
2. 在虚拟机安全套件加载前后服务器本身承受虚拟机开启的最大数量变化不大。
3. SecureVM加载前后宿主机服务器上虚拟机运行的最长时间变化不大。

#### 3.3.2预期结果

高频率加卸载软件不会导致自身和宿主机崩溃；当虚拟机安全套件系统运行后，服务器本身承受虚拟机开启的最大数量变化不大；SecureVM加载前后宿主机服务器上虚拟机运行的最长时间变化不大；达到测试要求。

#### 3.3.3实测结果

注：测试结果是在规定的实验环境下进行的，即服务器上，非台式机上。测试结果通过图形界面显示的主要原因：在另一台Windows系统平台上使用VNC软件，可以将服务器上的虚拟机以图形界面的方式展现，方面使用。

实测结果如表格所示：

|  |  |
| --- | --- |
| 测试项目 | 测试通过（Y/N） |
| 高频率加卸载软件 | Y |
| 服务器本身承受虚拟机开启的最大数量变化不大 | Y |
| 虚拟机运行的最长时间变化不大 | Y |

高频率加卸载软件：

通过shell脚本测试1000次系统的加卸载，每次时长11s多；运行load\_module.sh代码。

|  |
| --- |
| #load\_module.sh  #!/bin/bash  for ((i=1; i<=1000; i ++))  do  insmod secureVM.ko  virsh start vm1  sleep 5s  virsh destroy vm1  sleep 6s  rmmod secureVM.ko  done |

服务器本身承受虚拟机开启的最大数量变化不大：

原系统可开启20个虚拟机(4G 内存，4个VCPU，20G硬盘大小)，在安装了虚拟机安全套件之后，系统仍能够启动20个同等规模虚拟机。主要测试方法是在运行的每一个虚拟机上运行lmbench测试工具（内存测试工具）。记录原系统达到系统开销的某一情况状态，即临界状态，terminal界面打印数据不流畅，sar命令查看系统平均开销为80%。记录重新开机测试，在虚拟机安全套件系统运行后，记录系统开销达到80%左右（82%）时所能开启的虚拟机个数。实验发现，虚拟机开启所承受最大个数相同。

虚拟机运行的最长时间变化不大：

通过日志记载，在宿主机正常运行下，每个虚拟机运行时长超过7\*24\*3小时。

稳定性测试结果如下：

|  |  |
| --- | --- |
| 测试项目 | 测试通过（Y/N） |
| 高频率加卸载软件 | Y |
| 服务器本身承受虚拟机开启的最大数量变化不大 | Y |
| 虚拟机运行的最长时间变化不大 | Y |

### 异常情况测试

【包括非法参数、异常重启等。】

一种异常情况处理：

客户机异常中止，整个宿主机系统能正常运行，再次启动后，不影响其余客户机的正常运行。

#### 3.4.1测试用例

【主要说明测试的步骤及操作】

1. 客户机异常中止

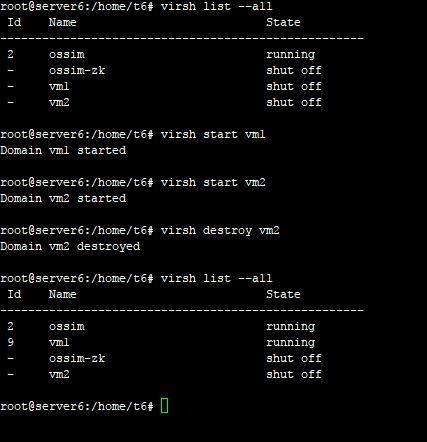
客户机因本身系统运行问题异常中止。通过destroy强制关闭客户机。

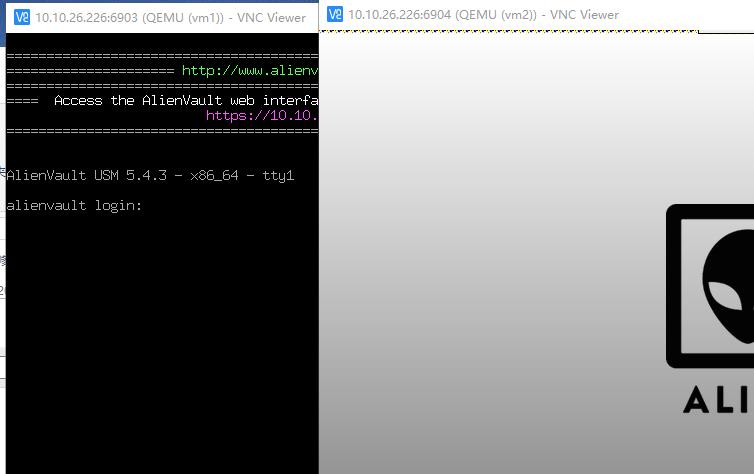
#### 3.4.2预期结果

宿主机正常运行，其余客户机正常运行。

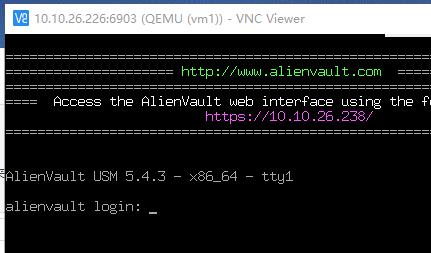
#### 3.4.3实测结果

1. destroy强制关闭客户机VM2



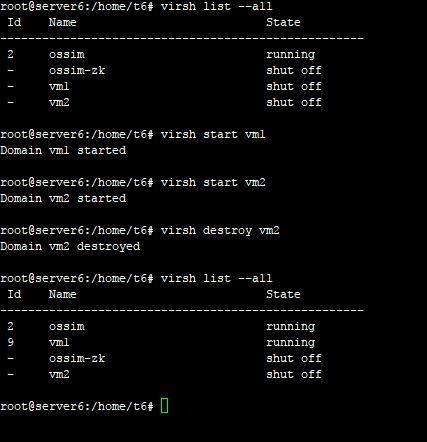


1. 其余客户机正常运行



1. 宿主机正常运行

当客户机运行时，宿主机正常运行，通过打印当前运行虚拟机信息来显示宿主机正常运行，如图。



## 评价

### 软件能力

经过项目测试，该虚拟机安全套件系统已达到交付标准，该系统能够实现用户需求说明书上的功能，能够满足用户的需求。

### 缺陷与限制

【说明测试所揭露的软件缺陷和不足，以及可能给软件运行带来的影响。】

1）当虚拟机安全套件系统被关闭后，客户机就不再受到安全保护。

### 建议

【提出弥补上述缺陷的建议。】

功能健全方面：在虚拟机退出的时候监控IO操作，可以对虚拟机的IO操作进行监控，随后对IO操作数据传输过程进行监控，保护传输中的IO数据。

### 测试结论

【说明能否通过】

通过功能、性能、异常、稳定性测试等，该系统能够实现用户需求说明书上的功能，测试通过。