# 基于虚拟化技术的客户虚拟机内核模块检测方法①

李 平[1,2], 崔超远[2], 李勇钢[1,2]

[1](中国科学技术大学 自动化系, 合肥 230027)
[2](中国科学院合肥物质科学研究院 智能机械研究所, 合肥 230031)

**摘 要**: 虚拟化技术是云计算的关键技术之一. 同时, 监视虚拟机又是虚拟化平台的一个重要功能. 为了更好的获取客户虚拟机的内部信息, 在 Xen 虚拟化环境中设计并实现了一种轻量级的虚拟机内核模块检测方法 KMDM. KMDM 驻留在宿主机里面, 利用虚拟机自省机制来获取被监控虚拟机的内核模块信息. 实验结果表明, KMDM 能够全面检测客户虚拟机的内核模块信息, 检测结果准确、可靠.

**关键词**: 虚拟化; 虚拟机自省; 虚拟机监视器; 模块检测

## Kernel Module Detection Method of Guest Virtual Machine Based on Virtualization Technology

LI Ping[1,2], CUI Chao-Yuan[2], LI Yong-Gang[1,2]

[1](Department of Automation, University of Science and Technology of China, Hefei 230027, China)
[2](Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230031, China)

**Abstract**: The virtualization technology is one of the key technologies of the cloud computing. Meanwhile, monitoring virtual machine is an important function on the virtualization platform. In order to obtain the internal information of the virtual machine better, a lightweight virtual machine kernel module detection method (KMDM) is designed and implemented on the virtualization environment of Xen. The KMDM resides in the host machine and uses the virtual machine introspection mechanism to obtain the kernel module information of the virtual machine. The experimental results show that the KMDM can detect the kernel module information of the guest virtual machine, and the detection results are accurate and reliable.

**Key words**: virtualization; virtual machine introspection; virtual machine monitor; module detection

随着云计算研究与应用的兴起, 虚拟化技术再次成为业内外的焦点. 虚拟化技术是云计算的基石, 是云计算服务得以实现的最关键技术[1]. 虚拟化技术分为三种: 全虚拟化、半虚拟化和硬件辅助虚拟化. 全虚拟化技术和半虚拟化技术的最大不同在于, 全虚拟化技术不需要修改虚拟机的操作系统内核, 而半虚拟化技术需要对虚拟机的操作系统内核进行修改. Xen[2]是一个开放源代码虚拟机监视器, 由剑桥大学开发. 它是一个基于 X86 架构、性能最稳定、占用资源最少的开源虚拟化技术. 目前 Xen 主要支持半虚拟化和硬件辅助虚拟化. 最初 Xen 采用半虚拟化技术, 通过软件方法实现了 X86 架构的虚拟化, 解决了敏感指令和特权指令无法被虚拟机监视器所捕获的问题. 后来随着硬件辅助虚拟化技术也加入 Xen 的开发中, 通过改变处理器的运行模式和指令集, 使虚拟机只能在受控模式下运行, 当需要由虚拟机监视器进行监控和模拟时, 由硬件支持模式切换, 因而无需修改虚拟机的操作系统内核.

虚拟机自省机制[3]将客户虚拟机内的操作系统状态信息拿到虚拟机监视器外面, 提供隔离的附加层, 为安全性、可靠性和管理带来了新的发展方向. 它主要利用操作系统内核数据结构特征来解决虚拟机监视器及内部操作系统间的语义鸿沟[4]问题. Tripwire[5]是一个基于主机的入侵检测系统实例. 它使用完整性检测方法来检测文件系统的安全性. Rootkit Revealer[6]是在虚拟机系统层次检测隐藏的内核可加载模块. 但是它能被那些阻止或转移磁盘卷或注册表访问的 rootkit 破坏. Strider GhostBuster[7]是基于交叉察看的方法来侦

① 收稿时间:2014-11-18;收到修改稿时间:2014-12-22

测隐藏文件, 注册表项, 进程和加载的模块.

以上方法都驻留在主机系统里面, 因此很容易被攻击者攻击和篡改. 一旦攻击者成功的侵入到系统里面. 恶意代码就会拥有和它们自身一样的特权来运行, 从而干扰系统的正常运行. 恶意软件通常以内核可加载模块的形式存在, 还具有隐藏的特性, 使得系统用户无法察觉. 同时, 内核模块都是运行在内核层, 拥有破坏操作系统数据的权限. 所以, 本文提出了一种基于虚拟化技术的轻量级的客户虚拟机内核模块检测方法 KMDM. KMDM 采用虚拟机自省技术在客户虚拟机外部获取客户虚拟机内部的内核模块列表信息而不用在客户虚拟机内部安装任何代码, 能够很好的避免恶意软件的攻击, 提高了系统的安全性. KMDM 在系统架构上使用了前后端分离的方法来实现, 具有很好的可移植性和通用性.

## 1 背景

### 1.1 Xen 和影子页表

Xen 是一款基于 GPL 授权方式的开源虚拟机软件, 它允许用户在一套物理硬件上执行多个虚拟机. Xen 由管理程序(Xen Hypervisor)和虚拟域组成. 管理程序是位于虚拟域和底层物理硬件之间具有最高权限的软件模块, 能够管理多个虚拟域的资源并监控虚拟域中操作系统的运行. Xen 采用混合模式[8]. 因此, 在 Xen 上的众多域中存在一个特权域(Dom 0)用来辅助 Xen 管理其他域(Dom U), 并提供相应的虚拟资源服务. Dom 0 拥有真实的设备驱动, 具有直接访问底层物理硬件的特权, 并且可以与其他虚拟域 Dom U 进行交互.

Xen 需要为每个虚拟机分配物理内存, 并且负责内存的管理. 为了让客户机操作系统能够使用一个隔离的、从零开始且具有连续性的内存空间, 虚拟机监视器引入了一层新的地址空间, 即客户机物理地址空间[9]. 因此就形成了从应用程序所在的客户机虚拟地址(GVA)到客户机物理地址(GPA), 再从客户机物理地址到宿主机机器地址(HMA)的两层地址转换. 前一个转换由客户机操作系统完成, 后一个转换由虚拟机监视器负责.

为了实现客户机虚拟地址到宿主机机器地址的直接转换, Xen 使用了影子页表技术. 影子页表[10]是被物理虚拟机监视器所装载使用的页表, 虚拟机监视器为每一个客户机操作系统中的每一套页表都维护了一套相应的影子页表. 如图 1 所示, 使用影子页表技术之后, 可以将两次地址转换合并为一次地址转换, 消除了客户机物理地址层, 从而直接实现客户机虚拟地址到宿主机机器地址的转换, 大大降低了额外的性能开销.

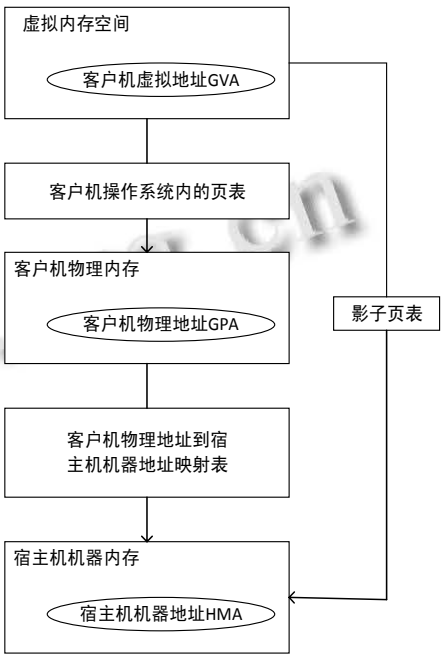


图 1 影子页表

### 1.2 模块管理

模块是一种向 Linux 内核添加设备驱动程序、文件系统以及其他组件的有效方法, 而无需连编新内核或者重启系统[11]. Linux 内核把模块的列表存放在一个双向循环链表中. 链表中的每一项都是类型为 module 的结构体, module 结构体包含了许多和模块描述有关的成员变量, 如状态(state)、模块名(name)和用作模块链表的链表元素(list)等. 只要找到了任意一个模块, 就可以通过当前模块的 list 成员变量中的 next 指针值得到下一个模块的 module 结构体的地址, 依次往下查找即可完成所有模块的遍历. 图 2 是内核模块的 module 结构体之间的链表关系.

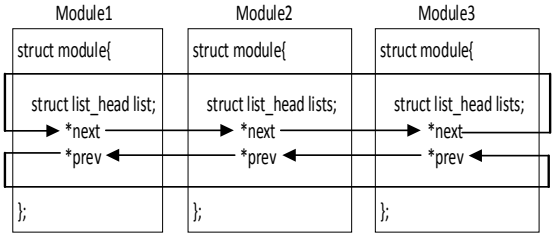


图 2 内核模块的 module 结构体之间的链表关系

## 2 系统实现

KMDM 采用虚拟机自省技术来获取虚拟机的物理内存信息. 考虑到系统的准确性、可移植性和高效性, KMDM 系统架构采用前后端分离的方法实现. 因此, KMDM 由前端驱动和后端驱动两部分组成. 后端驱动负责获取内核模块相应结构体的语义信息, 前端驱动负责完成虚拟机内的内核模块列表检测. KMDM 系统架构及前后端驱动之间的关系如图 3 所示.

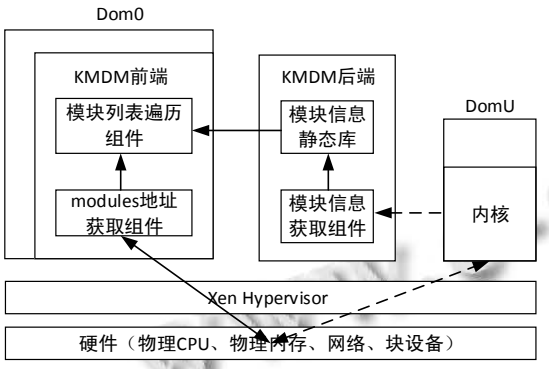


图 3　KMDM 系统架构及前后端驱动之间的关系

### 2.1 前端驱动

KMDM 的前端驱动主要由 modules 地址获取组件和模块列表遍历组件两个部分组成. 其中, 前者根据 System.map 文件获取 modules 的地址, 后者根据上述地址完成整个模块列表的遍历.

前端驱动在虚拟机外部完成虚拟机内部的内核模块列表检测. 它根据后端驱动提供的模块信息静态库, 解析物理内存, 将从外部获得的虚拟机的底层状态信息还原为虚拟机内部与内核模块相关的语义内容, 构建内核模块的高级语义视图, 实时检测虚拟机内部的内核模块信息.

KMDM 进行客户虚拟机内部的内核模块列表检测的一般步骤如下:

① 根据 System.map 文件读取内核模块列表的地址信息.

由于每个客户虚拟机操作系统的/boot 目录下都有一个 System.map 文件. 而 System.map 文件是一个特定内核的内核符号表, 因此根据 System.map 文件就可以得到内核模块列表的地址信息. 以 Ubuntu 14.04 为例, 可以通过如下命令得到内核模块列表的地址.

```
$sed -n '/d modules$/p' System.map
ffffffff81c52ff0 d modules
```

其中 ffffffff81c52ff0 就是客户虚拟机 Ubuntu 14.04 的内核模块列表地址. 表 1 列出了不同客户虚拟机内部的内核模块列表地址信息.

表 1　不同客户虚拟机内部的内核模块列表地址

| 操作系统 | 模块列表地址信息 |
|---|---|
| Ubuntu 14.04 | ffffffff81c52ff0 |
| Debian 7.6.0 | ffffffff8161db40 |
| Centos 6.5 | ffffffff81aa8090 |
| Fedora 19 | ffffffff81c4a250 |

② 调用后端的 getnameoffset 函数, 获得 module 结构体中 name 成员变量的偏移量 o_name. 利用影子页表 SPT(addr_module, o_name)函数将 name 成员变量的客户机虚拟地址转换成宿主机机器地址. 根据 name 成员变量的宿主机机器地址输出模块名称.

③ 调用后端的 getlistoffset 函数, 获得 module 结构体中 list 成员变量的偏移量 o_list. 利用影子页表 SPT(addr_module, o_list)将 list 成员变量的客户机虚拟地址转换成宿主机机器地址. 读取 list 成员变量的宿主机机器地址里的内容, 并根据 o_list 的值来得到模块列表中下一个模块的地址信息.

④ 判断是否完成模块列表的遍历. 若没有, 则重复执行步骤②和步骤③的内容遍历模块信息, 否则结束模块列表的遍历.

### 2.2 后端驱动

KMDM 的后端驱动主要由模块信息获取组件构成. 在离线状态下解析客户虚拟机操作系统的内核数据结构(如 module 结构体), 生成一系列描述模块信息的访问函数, 构建模块信息静态库, 提供模块信息的访问接口, 并将模块的相关信息传递给前端驱动. 例如, 通过如下函数就可以获得客户虚拟机内核模块名称的相关信息:

```
unsigned int getnameoffset(void)
{
        return &(((struct module *)0)->name);
}
```

## 3 实验结果

### 3.1 实验环境

为了验证系统的可行性, 在开源虚拟化平台 Xen 4.1.2 上实现了虚拟机内核模块检测方法 KMDM. 宿主机运行于 Intel Core i5-3470 (四核, 主频 3.20GHz)、4GB 内存的硬件之上. 实验环境配置如表 2 所示. 其

中, Dom 0 是宿主机, Dom 1 到 Dom 4 是客户虚拟机.

表 2 实验环境配置

| 虚拟域 | CPU | 内存 | 操作系统 |
|---|---|---|---|
| Dom 0 | 4 核 | 4GB | Ubuntu 12.04.4 |
| Dom 1 | 1 核 | 1GB | Ubuntu 14.04 |
| Dom 2 | 1 核 | 1GB | Debian 7.6.0 |
| Dom 3 | 1 核 | 1GB | Centos 6.5 |
| Dom 4 | 1 核 | 1GB | Fedora 19 |

### 3.2 内核模块检测

下面以客户虚拟机 Ubuntu 14.04 为例, 从客户虚拟机外部采用虚拟机自省技术对客户虚拟机内部内核模块列表信息进行检测, 并与从客户虚拟机内部得到的内核模块列表信息相比较来验证 KMDM 检测结果的准确性.

图 4 是某一时刻客户虚拟机内外获取的内核模块列表对比. 图 4 的左边部分是通过 KMDM 在客户虚拟机外部生成的模块列表, 图 4 的右边部分是在客户虚拟机内部执行“lsmod”命令获取的模块列表. 通过比较可以发现, 使用 KMDM 得到的模块列表和客户虚拟机内部得到的模块列表完全一致. 因此, 图 4 的实验结果证实了 KMDM 能够准确无误的检测出客户虚拟机内的模块列表信息.

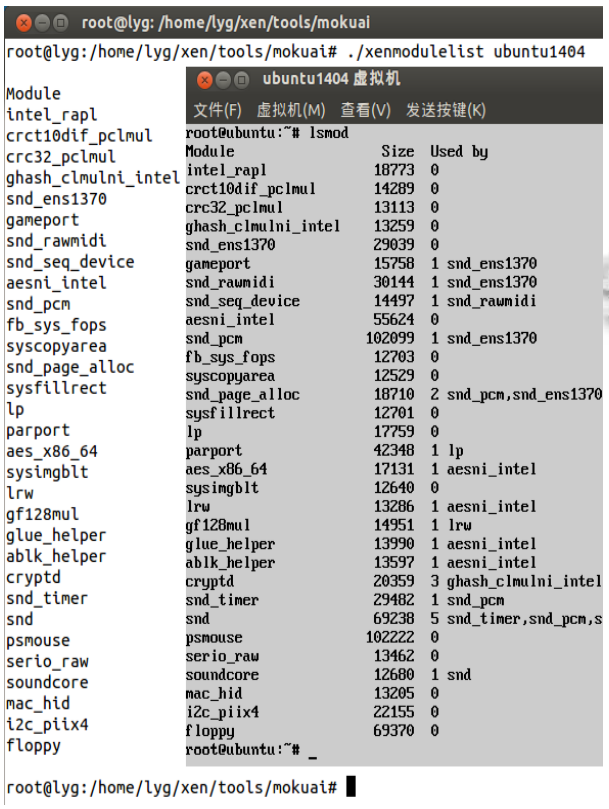


图 4 客户虚拟机内外获取的内核模块列表对比

为了进一步证实 KMDM 检测内核模块列表信息的准确性, 在客户虚拟机 Ubuntu 14.04 内部执行“insmod hello.ko”命令插入 hello 模块后, 在客户虚拟机外部执行客户虚拟机内核模块列表检测命令来检测客户虚拟机内部的内核模块列表信息. 图 5 显示了客户虚拟机内部新插入模块的检测结果. 从图 5 中的检测结果可以看出, KMDM 能够实时的检测出新插入的模块信息.

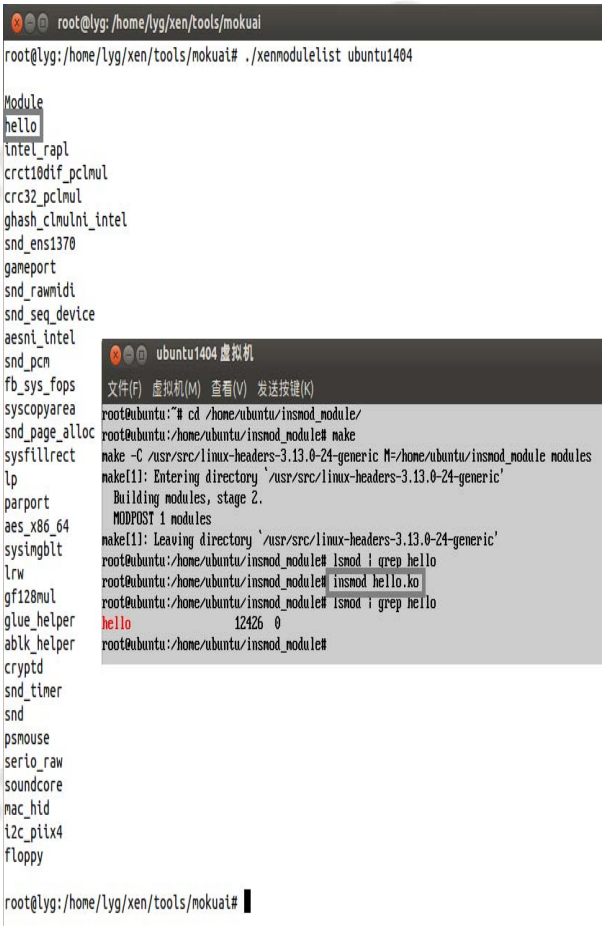


图 5 客户虚拟机内部新插入模块的检测结果

### 3.3 执行效率

图 6 是四种不同客户虚拟机操作系统环境下, KMDM 检测内核模块所需的时间. 其中, 横坐标是客户虚拟机的类型, 纵坐标是 KMDM 进行 10 次模块信息检测耗费时间的平均值. 由图 6 可以看出, 在四种不同客户虚拟机操作系统下, KMDM 检测模块列表所花费的时间相差不大, 检测时间都在 10 毫秒左右. 因此, 实验结果说明 KMDM 可以快速的检测出客户虚拟机内的内核模块列表. 而且, 不同的客户虚拟机操

作系统对 KMDM 的检测时间影响很小, 说明 KMDM 具有很好的可移植性和复用性.

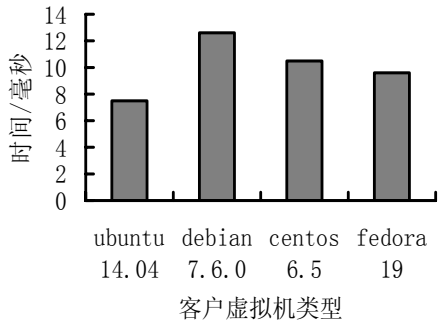


图 6 KMDM 检测模块时间

## 4 结论

本文提出了一种轻量级的客户虚拟机内核模块检测方法 KMDM. KMDM 驻留在被检测虚拟机的外部, 利用虚拟机自省技术来获取客户虚拟机的内核模块信息. 所以, 无需在客户虚拟机内部安装任何代码, 真正做到透明、实时监控. 同时, KMDM 的系统架构采用了前后端分离的设计方案来实现检测内核模块列表信息的功能. 因此, 能够很容易扩展到其他操作系统上, 增强了 KMDM 的可移植性和复用性. 最后, 通过在典型的 Linux 操作系统上进行测试, 证明了 KMDM 能够快速的进行模块列表检测, 检测效率高, 并且检测结果准确、可靠.

## 参考文献

1 广小明,胡杰,陈龙,郭京,等.虚拟化技术原理与实现.北京:电子工业出版社,2012.

2 Barham P, Dragovic B, Fraser K, Hand S, Harris T. Xen and the Art of virtualization. Proc. of the 19th ACM Symposium on Operating Systems Principles. New York. ACM. 2003. 164–177.

3 Garfinkel T, Rosenblum M. A virtual machine introspection based architecture for intrusion detection. Proc. of the 10th Annual Network and Distributed System Security Symposium. San Diego, California. 2003.

4 Chen PM, Noble BD. When virtual is better than real. Proc. of the Eighth Workshop on Hot Topics in Operating Systems. Elmau, Germany. IEEE, 2001. 133–138.

5 Kim GH, Spafford EH. The Design and implementation of tripwire: a file system integrity checker. Proc. of the 2nd ACM Conference on Computer and Communications Security. New York. ACM. 1994. 18–29.

6 Rootkit Revealer 1.71. http://www.filehippo.com/zh/ download _rootkit_revealer.

7 Wang Y, Beck D, Vo B, Roussev R, Verbowski C. Detecting Stealth Software with Strider GhostBuster. Proc. of International Conference on Dependable Systems and Networks. Washington, DC. IEEE. 2005. 368–377.

8 石磊,邹德清,金海.Xen 虚拟化技术.武汉:华中科技大学出版社,2009.

9 英特尔开源软件技术中心,复旦大学并行处理研究所.系统虚拟化:原理与实现.北京:清华大学出版社,2009.

10 Wang Z, Jiang X, Cui W, Ning P. Countering kernel rootkits with lightweight hook protection. Proc. of the 16th ACM Conference on Computer and Communications Security. New York. ACM. 2009. 545–554.

11 Mauerer W. Professional Linux Kernel Architecture. Wrox Press Ltd, 2008.