基于混合流策略的按需分布式云信息流控制模型

杜远志 杜学绘 杨 智
(中国人民解放军信息工程大学四院 郑州 450001) (河南省信息安全重点实验室 郑州 450001)
(数学工程与先进计算国家重点实验室 郑州 450001)

摘 要 为确保云平台上虚拟机系统用户信息的安全,提出了一种基于混合流策略的按需分布式云信息流控制模型(Mixed Flow Policy Based On-demand Distributed Cloud Information Flow Control Model,MDIFC)。该模型以分布式信息流控制模型为基础,结合中国墙策略形成混合流策略,通过引入污点传播思想跟踪来敏感数据以实现策略,为用户数据提供更好的安全保障。为提高模型的灵活性,考虑到虚拟域行为更具主动性的特征,提出了"按需受控"的概念及与之相适应的"输出型机密性"。同时,通过按需受控显著地降低了污点传播造成的开销。利用π演算对模型规格进行形式化描述,并借助PicNic工具证明模型的无干扰性。最后,通过一个应用示例说明了模型的实用性。

关键词 云计算,信息流控制,按需污点传播,中国墙策略,π演算
中图法分类号 TP316 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.10.029

Mixed Flow Policy Based On-demand Distributed Cloud Information Flow Control Model

DU Yuan-zhi DU Xue-hui YANG Zhi
(College 4,PLA Information Engineering University,Zhengzhou 450001,China)
(Henan Province Key Laboratory of Information Security,Zhengzhou 450001,China)
(State Key Laboratory of Mathematical Engineering and Advanced Computing,Zhengzhou 450001,China)

Abstract In order to protect the security of user information in virtual machine on the cloud platform,this paper proposed a mixed flow control based on-demand distributed information flow control model (MDIFC). This model developes from DIFC,and the taint propagation is introduced to track the sensitive data so that the system can enforce the strategy and the user data can be protected better. In order to improve the flexibility of the model,considering the initiative of virtual domains,the concept of on-demand controlled and output classification were proposed. The model can reduce the workload result from taint propagation at the same time. This paper introduced its specification using π calculus and proved the security property of noninterference of MDIFC system with PicNic tool. Finally,this paper used an example to demonstrate of MDIFC.

Keywords Cloud computing,Information flow control,On-demand taint propagation,Chinese wall policy,π calculus

1 引言

对于云服务供应商而言,如何保护云平台内部的敏感信息,即如何保证云平台内的数据只能被授权主体访问,是云平台需要解决的基础性问题。与以往相比,近年来虽然云计算的安全事故的报告已减少许多,但是每次安全事故带来的危害却在逐年增大,如2014年8月的iCloud照片数据信息事件表明,许多组织对于在云平台上的敏感信息的保护存在重大疏漏。随着社会对信息技术的日渐依赖,信息传输渠道日益多样化,敏感信息的安全问题将越发复杂且难以解决。

云计算数据安全的保护重点在于解决以下3方面的问题[1]:1)云计算服务计算模式所引发的安全问题。在使用云计算服务时,用户或企业需要将数据外包给云服务供应商,将应用交付给供应商执行,这使得供应商具备对这些数据及应用的访问权。如何避免供应商对数据的威胁是云计算安全需要解决的首要问题。2)云计算的动态虚拟化管理方式引发的安全问题。因为云计算通过虚拟租用的方式将资源提供给用户使用,不同的用户很可能使用同一个物理资源,而当这些用户之间存在利益冲突时,应该避免用户数据被其他用户访问。3)云计算中多层服务模式引发的安全问题。此问题不是本模型解决的重点,在此不进行详细说明。

首先,对于如何解决供应商这类第三方用户对数据的威胁这一问题,安全界早在1988年就通过中国墙策略进行了实现。关于如何在云环境下进行中国墙策略的实现已有诸多讨

到稿日期:2016-09-28 返修日期:2017-02-07 本文受基于多维控制的云计算信息流追责、管控技术(863)(2015AA016006),国家重点研发计划项目:协同精密定位总体架构与服务平台设计(2016FYB0501901)资助。
杜远志(1992-),男,硕士生,主要研究方向为信息安全,E-mail:1776253483@qq.com;杜学绘(1968-),女,博士,教授,博士生导师,主要研究方向为空间信息网络、信息安全;杨 智(1975-),男,博士,副教授,主要研究方向为信息安全。

论，各策略在动态性、灵活性等方面仍有很大缺陷，详见2.4节。本文模型则能有效提高中国墙策略在云环境下的动态性与灵活性。

第二个问题仍可通过中国墙策略解决，但其在商业环境下的可用性仍存在很大缺陷，例如处理商业机密时中国墙策略无法对联盟内部的信息流进行控制。为此，本文模型首先借鉴了针对分布式环境的分散式信息流控制模型(Decentralized Information Flow Control，DIFC)[2]。但 DIFC 仍存在以下问题：1)模型策略的设计过于简单，仅提供了机密性保护，无法保证系统的完整性。2)只有数据拥有者可以对数据进行控制，无法满足复杂的云环境下的安全需求。为解决上述问题，本文模型以 Xen 的虚拟化方式作为云安全研究的基础，引入污点分析技术，针对其虚拟化特性，提出了按需分布式云信息流控制模型。

2　相关工作

2.1　Xen 虚拟机结构

Xen 最初是由剑桥大学计算机实验室基于 x86 结构开发的一款半虚拟化虚拟系统，现已支持硬件辅助虚拟化及全虚拟化与半虚拟化混合模式。半虚拟化指虚拟机需要对源码作出修改才能使用，硬件辅助虚拟化及全虚拟化则无需修改。由于本文以半虚拟化的 Xen 为基础进行研究，因此下面将先对半虚拟化的 Xen 系统进行简单的介绍。

如图 1 所示，Xen 虚拟机主要由虚拟机管理程序(Xen Hypervisor)、特权虚拟域 0(Dom0)以及数量众多的非特权虚拟域(DomU)组成。其中，虚拟机管理程序直接位于硬件之上，负责与硬件直接交互并进行管理。为此，管理程序需要工作在 x86 处理器特权级的最高级 ring 0 上，而虚拟域中的操作系统内核则处于次一级的 ring 1 上，应用程序仍处于 ring 3。特权虚拟域是 Xen 系统为了实现操作和控制机制相分离的产物，是最先启动的虚拟机，虚拟机管理程序向特权虚拟域提供对非特权虚拟域的管理接口，如虚拟域的创建、删除、销毁等。由于各虚拟域对物理设备的需求，Xen 系统采用了分离设备驱动模型(Split Device Driver Model)，由特权虚拟域管理系统各物理外设的驱动，非特权虚拟域通过域内的前端设备驱动以及特权虚拟域的后端设备驱动来交互使用设备。

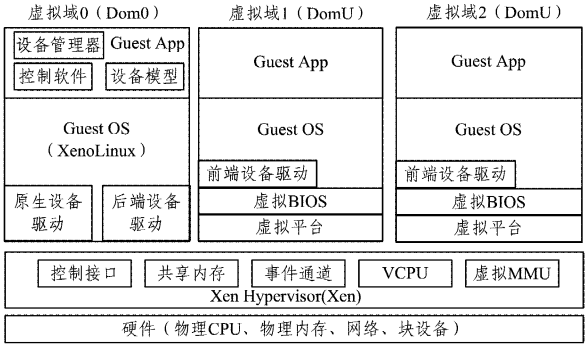


图 1　Xen 虚拟机的结构图

2.2　信息流控制模型

传统的控制模型往往将控制限于最初的访问时刻，在用户获取相应的数据后，数据的安全则交由用户管理，即数据已不再安全。而信息流控制模型则将控制扩展到数据的传播与应用中，从而保护数据的整个生命周期。信息流控制模型(IFC)的核心思想是限制用户信息的流动，防止用户信息被窃取。例如，在一个商业环境中，一个带有敏感内容的文件 F 和源于 F 的数据只能被项目组 G 的成员得到。传统控制模型如 DAC 或 RBAC 只能限制用户对文件 F 的访问，如果存在项目组 G 的成员在文件 F 的基础上生成了新的文件 F'，则 DAC 无法自动地为新生成的文件提供访问控制。IFC 模型则能对其保持相同的权限控制。虽然基于信息流控制的安全机制较传统模型在敏感数据传播方面更具安全性，但在实现上模型往往需要提供额外的机制进行辅助：系统必须能够在数据应用阶段进行跟踪，在主体交互的边界上实现控制。本文不对虚拟域域内的信息流控制进行研究，而以杨智的 GT-PM 为基础进行研究。下面介绍当前云环境下信息流控制方法的研究现状。

文献[3]利用云供应商的角色优势提出了基于可信计算的信息流监控方式，对用户交互行为进行监控与分类，阻止非法通信，然而并未考虑自身可能造成的信息泄露问题。张怀方等[4]通过将显式与隐式安全标记相结合的方法提出了一种基于虚拟机的信息流安全控制方法，旨在弥补可信系统中信息流控制方法的不足。但模型内主体密级固定，其虽然能够提供较好的安全性，但在使用上将造成极大的不便。FlowK[5]和 FlowR[6]则分别以可加载的内核模块和动态链接库的方式在 PaaS 层上实现信息流控制。Jean 等[7]在理论上探讨了 DIFC 在 PaaS 上的可行性。但在 PaaS 层上实现的控制技术往往受限于最初设计模型的平台，难以移植。综上所述，有必要提供一种更为灵活的信息流控制模型，且模型在具备灵活性的同时还应保持租户间与租户-供应商间的有效隔离。

2.3　中国墙模型

Brewer 和 Nash[8]于 1988 年依据现实中的商业规则提出了中国墙模型。之后，Tylin[9]在对实际环境进行更深入的研究后提出了侵略型中国墙安全策略模型，该模型引入了两个新概念(利益冲突关系和同盟关系)以解决原模型中的冲突传递问题。本文是侵略型中国墙模型在云计算环境下的应用，因此下面对该模型进行简要的介绍。如文献[8]所述，模型试图应用于此类场景：一个市场分析人士在向企业提供服务时需要遵守行业规则，即当企业需要分析人员作出分析时，企业向分析人员提供内部计划、财报等信息，分析人员需要为向他提供信息的企业保密，他更不能向该企业的竞争者提供相关信息。当不存在竞争时，分析人员则可自由地向企业提供分析建议。以企业为例，图 2 所示的中国墙模型将所有信息分层存储，从下往上依次是：企业信息项、企业信息集和利益冲突类(Conflict Of Interest classes，COI)。企业信息项是模型最基本的客体，是模型保护的元素。企业信息集是属于同一企业的企业及信息项的集合。利益冲突类是存在竞争关系的企业及其企业信息集的集合。模型最核心的概念是利益冲突类，主体最多只允许访问每个冲突类中的一个企业信息集，从而保证分析人员只能获得与他们已经拥有的其他信息不相冲突的信息。而 Tylin 在对原模型冲突关系进行分析后，针对原模型缺乏利益冲突传递性这一模型缺陷，提出了同盟关系，

中国墙模型的总体架构基本完成。

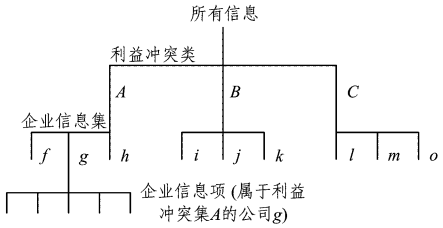


图 2 中国墙模型的层次

在模型的实践上,文献[10]最先将中国墙模型应用于云计算的 SaaS 中,但应用的方式较为陈旧,特别是其以静态的方式对访问区域进行划分的方法与云计算的应用场景有较大脱节。Katsuno[11] 与 Jaeger[12] 也分别从分布式环境和虚拟技术的角度讨论了以静态划分访问区域的方式对中国墙模型进行实现。然而,在另外一些应用场景中,如在动态合作组织中实施访问控制时,也通常需要动态划分访问区域,允许信息在某一个域中流动。为此,程戈[13] 提出了一种基于动态联盟关系的中国墙模型,该模型通过建立联盟关系实现对利益冲突关系的扩展,从而动态地构建访问区域。但他们缺乏实践,仅仅从理论上证明了该模型的可行性,且模型通过访问控制矩阵记录系统状态的方式并不适用于分布式环境。文献[14]通过提出分集的概念,从主、客体两方面研究冲突关系,实现访问区域的动态划分;模型能够有效保证主体与客体间信息流的安全,但没有研究主体间冲突关系的扩展。通过分析以上研究结果可以看出,当前的中国墙模型研究中缺乏动态调整联盟关系的方法,特别是对于当前的云计算应用环境,缺乏实践能力。

3 基于 MDIFC 的云系统信息流控制

3.1 MDIFC 模型的设计思想

基于上述调查研究,本文提出了 MDIFC 模型,该模型由虚拟域域间安全子模型与按需受控子模型两方面组成,其设计思想如下:

虚拟域域间安全子模型以分散式信息流控制模型 DIFC 为基础,在域内通过广义污点传播模型 GTPM[15] 实现虚拟域内系统安全的同时,在虚拟域间利用 GTPM 在域内所使用的污点作为敏感信息在虚拟域间传播时进行控制的依据。然而,在云计算环境下,该模型在应用上仍有许多不足,如本文引言所述的安全挑战。为此,本文引入了中国墙模型,每个组织或企业拥有各自独一无二的标签,同一组织下的文件使用同一个标签,模型通过标签实现中国墙策略。中国墙策略的标签能以污点传播的方式在虚拟域间传播,在传播的过程中,中国墙标签动态地实现了中国墙策略中的联盟关系,使得租户间与租户-供应商间能保持动态有效的隔离。综上所述,本模型结合了信息流控制模型、分布式信息流控制模型以及中国墙模型,因此本文将模型命名为基于混合流策略的分布式信息控制模型。

按需受控子模型的提出是因为传统的信息流控制模型的主体是进程,而虚拟域的功能与应用较进程更加复杂,这就需要访问控制模型提供一个更为灵活的交互方式,因此本文根据信息的流入方向将机密级分为输出机密级与输入机密级,使虚拟域更具灵活性与可用性,其中输出机密级由正处于域内的敏感信息机密级性共同决定,输入机密级由所有曾流入虚拟域的敏感信息机密级的并集组成。该方法的可行性在于虚拟域的内存中并不总是敏感信息,此时内存中的非敏感信息并不具备通过污点跟踪进行访问控制的价值,因此综合考虑安全性与系统开销的问题,提出按需污点跟踪控制的概念,即只跟踪控制敏感信息。通过进一步考虑污点跟踪的实现方式,本文提出了按需受控子模型。按需即在一个虚拟域中,当用户的操作使得敏感数据进入内存时才会触发污点传播控制机制。受控则是受控虚拟域的简称,受控虚拟域是虚拟域处理敏感数据时,系统暂停并复制虚拟域,从而生成的虚拟域副本,该副本通过盒外跟踪技术实现污点控制。因为污点控制的实现往往需要在系统内添加污点跟踪模块,这意味着在部署新系统时需要主动地为系统安装污点跟踪模块,而这无疑将给用户的使用带来极大的不便。同时,模块即便不工作,也会给系统带来不必要的开销。

3.2 模型元素

MDIFC 模型的元素包括主体和客体、动作、安全标记、标签及能力。下面介绍模型元素并给出元素的半形式化描述。

(1)实体 E(Entity)是系统中数据的承载者,实体在系统中依据操作的主动性可分为主体 S 与客体 O。主体 S 是系统操作的执行者,在本模型中为虚拟域 Dom。客体 O 是主体动作的承受者。实体包括虚拟域、文件、外设资源等。

(2)动作集 acts 是主体与客体间的动作,是系统并行的基础。不同的主体有不同的动作集合。虚拟域主体负责虚拟域之间的操作,其动作集合包括对虚拟域的创建(creat)、关闭(close)、暂停(stop)、挂起(suspend)及恢复(resume)等。虚拟域间通信的基本方式包括内存共享和事件通道,内存共享包括内存传递、内存复制、内存映射。

(3)安全标记 Tag 是可信主体判定主体动作安全性的根本依据。按照安全类型可将其分为机密性标记 C、完整性标记 I 以及中国墙标记 CW,即 $Tag=C\cup I\cup CW$。本文中的敏感信息定义为带有机密性标记、完整性标记及非本域中国墙标记的信息。

(4)标签 Label 表示各个主体与客体安全标记的集合,是主、客体间动作判断的直接依据。f 为标签 Label 的提取函数,$f_c(X)$ 表示提取实体 X 的机密性标记集,$f_{Oc}(X)$ 表示提取实体 X 的输出型机密性标记集,$f_I(X)$ 表示提取实体 X 的完整性标记集,$f_{CW}(X)$ 表示提取实体 X 的冲突集标签,$f_{CW0}(X)$ 表示提取实体 X 的初始冲突集标签,即实体 X 的组织信息。

(5)能力 $A=\{C,I,CW\}\times\{+,-\}$,用来表达主体对客体标签内的安全标记的修改能力。其中,A_S^+ 表示主体 S 的安全标记增加能力,A_S^- 表示主体 S 的安全标记删除能力,$A_S^{\pm}$ 表示主体 S 的安全标记增删能力。

(6)冲突域 Conflict 中的实体与其他一些实体存在冲突,定义冲突集函数 $Cf(s)$ 表示与实体 s 存在冲突的实体集合。

(7)安全决定集 $D=\{accept,deny,error\}$,$D(S,O,act)$ 表示主体 S 对客体 O 发起动作 act 的请求,返回值为 $accept$(接受)、$deny$(拒绝)、$error$(未知错误)。

3.3 安全规则

安全规则决定着模型的安全性，本节将对安全规则进行介绍。对于虚拟域域内的操作规则，可依据 GPTM 模型进行分析，本文集中对域间子模型规则进行讨论。

规则 1（安全信息流规则）　当信息流通过动作 act 从虚拟域 S 流向虚拟域 D 时，当且仅当下述条件都满足时，系统通过请求。

$$\begin{cases} f_{OC}(S)-A_S^{\pm}\subseteq f_c(D)\cup A_D^{+} \\ f_I(S)-A_S^{\pm}\subseteq f_I(D)\cup A_D^{+} \\ D\notin Cf(S) \end{cases} \tag{1}$$

除了从完整性和机密性上考虑信息流流动的安全性外，相比于进程子模型，域间子模型需要更多地考虑虚拟域在使用上的冲突情况，若两个虚拟域的标签处于一个冲突集中，则不能进行交互。针对不同的操作类型，有如下示例。

(1)虚拟域 S 可删除虚拟域 D，当且仅当下列条件同时满足：

$$\begin{cases} f_C(S)-A_S^{\pm}\subseteq f_c(D)\cup A_D^{+} \\ f_I(S)-A_S^{\pm}\subseteq f_I(D)\cup A_D^{+} \\ D\notin Cf(S) \end{cases} \tag{2}$$

需要满足相同条件的合法操作还包括：虚拟域 S 暂停虚拟域 D，由虚拟域 S 向虚拟域 D 传递内存，由虚拟域 S 复制内存到虚拟域 D 中，虚拟域 D 读取虚拟域 S 提供的内存映射。

(2)虚拟域 S 可启动虚拟域 D，当且仅当下列条件同时满足：

$$\begin{cases} f_C(S)-A_S^{\pm}\subseteq f_c(D)\cup {}^{A}D^{+} \\ f_I(S)-A_S^{\pm}\subseteq f_I(D)\cup {}^{A}D^{+} \\ f_{CW_0}(S)=f_{CW_0}(D) \end{cases} \tag{3}$$

式(3)中第三条规则与安全规则不一致的原因在于一个组织的虚拟域只能由启动本组织的虚拟域进行启动，这样的控制无疑更符合云计算的使用环境。虚拟域启动时，还需考虑系统之前占用的资源的使用情况，特别是当系统从暂停中启动时，若系统原先占用的资源被其他虚拟域使用，则可能造成未知错误，留下安全隐患。

(3)虚拟域 S 与虚拟域 D 之间可进行域间通信，当且仅当下列条件同时满足：

$$\begin{cases} f_{OC}(S)-A_S^{\pm}\subseteq f_c(D)\cup A_D^{+} \\ f_I(S)-A_S^{\pm}\subseteq f_I(D)\cup A_D^{+} \\ f_{OC}(D)-A_D^{\pm}\subseteq f_c(S)\cup A_S^{+} \\ f_I(D)-A_D^{\pm}\subseteq f_I(S)\cup A_S^{+} \\ D\notin Cf(S) \end{cases} \tag{4}$$

即虚拟域相互满足安全信息流流动条件。通信完毕后，虚拟域双方即可无条件撤销通信。

3.4 标签转换规则

标签转换规则规定了主(客)体标签内标记发生改变的时机，是信息流模型进行跟踪的基础。

规则 2（标签隐式转换规则）　当发生信息流由 S 流向 O 时，实体 O 的标签将发生如下转换：

$$\begin{cases} f_C(O)\leftarrow f_C(O)\cup(f_{OC}(S)-A_S^{\pm}) \\ f_{OC}(O)\leftarrow f_{OC}(O)\cup(f_{OC}(S)-A_S^{\pm}) \\ f_I(O)\leftarrow f_I(O)\cup(f_I(S)-A_S^{\pm}) \end{cases} \tag{5}$$

规则 3（标签显式转换规则）　$Label_S$ 是主体 S 的标签，$Label_S'$ 是 S 请求的新标记集合，从 $Label_S$ 到 $Label_S'$ 的显式变化是安全的，当且仅当：

$$\begin{cases} Label_S'-Label_S\subseteq A_S^{+} \\ Label_S-Label_S'\subseteq A_S^{-} \end{cases} \tag{6}$$

与之类似，$Label_S$ 是主体 S 的标签，$Label_O$ 是客体 O 的标签，$Label_O'$ 是 S 请求的关于 O 的 $Label_O$ 的新标签，当且仅当：

$$\begin{cases} Label_S-A_S^{\pm}\subseteq Label_O\subseteq Label_S\cup A_S^{\pm} \\ Label_S-A_S^{\pm}\subseteq Label_O' \end{cases} \tag{7}$$

系统通过请求。

主体可以在能力范围内主动改变自己的标记，可实现灵活的信息流控制，标记的显式变化是隐式变化的有效补充。

规则 4（中国墙展开规则）　当发生信息流由 S 流向 O 时，则：

$$Cf(O)=Cf(S)=Cf(O)\cup Cf(S) \tag{8}$$

即进行交互的实体交互后拥有相同的冲突集。

4 模型形式化规格及无干扰性分析

4.1 形式化规格

形式化规格是通过数学语言对数学模型进行准确描述的方法。通过形式化规格对模型进行表达后，能够利用已有的理论对其性质进行有效的分析与认证。在本文中，模型使用 π 演算进行形式化描述。π 演算[16-18]是由 Robin Milner[19]于 1991 年基于 CCS 提出的，能够描述和验证并发系统。

如第 3 节所述，本文将 MDIFC 分为两个子模型：1)在 Xen 上的虚拟域层构成的域间子模型，重点关注虚拟域间的信息流；2)按需副本安全子模型，用来提高模型的灵活性。

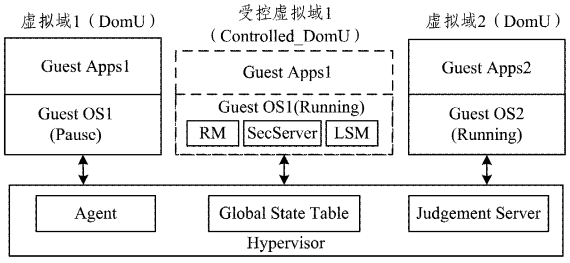


图 3　MDIFC 模型的组成

图 3 给出了 MDIFC 模型的组成，代理(Agent)负责监控虚拟域的 I/O 操作，实现虚拟域与受控虚拟域副本间的转换控制；全局状态表(Global State Table，GST)负责存储和维护云平台内所有虚拟域的标记状态能力等模型相关信息；判决服务器(Judgement Server，JD)依据中国墙模型策略保证域间通信安全。在受控虚拟域中，与 GST 对应的标记状态管理器(Label State Manager，LSM)位于各受控虚拟域内部，负责管理本虚拟域内系统实体的标记状态能力信息；引用监视器(Reference Monitor，RM)依据模型策略在域内实施访问控制；安全服务器(SecServer)则根据 LSM 提供的标记能力信息对申请的操作进行判决并更新 LSM 内的相关标记信息。

4.1.1 域间子模型交互规格的描述

在域间子模型中，虚拟域通过超级调用访问敏感系统资源，虚拟域之间通过超级调用使用事件通道及共享内存进行信息的交互。用 π 演算的过程中 Dom_{U_i} 表示虚拟域 i，域间子模型所有虚拟域的行为描述如下：

Doms=Dom0|! DomU

Dom0 是 Xen 系统启动的第一个虚拟域，负有管理所有虚拟域的责任，可以创建、关闭、重启、销毁、保存、恢复及迁移虚拟域，能管理虚拟域的块设备及网络接口，并对虚拟域进行实时监控。

如图 4 所示，在域间子模型中，管理程序 Hypervisor 掌握源动作域 Dom_{source} 与目标动作虚拟域 $Dom_{destination}$ 的标签信息。当域 Dom_{source} 向管理程序 Hypervisor 查询对域 $Dom_{destination}$ 的权限时，管理程序 Hypervisor 以其内部全局标签表(Global Label Table，GLT)的内部相关数据作为决策服务器(Decision-making Server，DS)的输入，DS 根据预定规则作出判决。若判决为允许，则 Dom_{source} 开始与 $Dom_{destination}$ 进行交互；否则，返回拒绝信息。

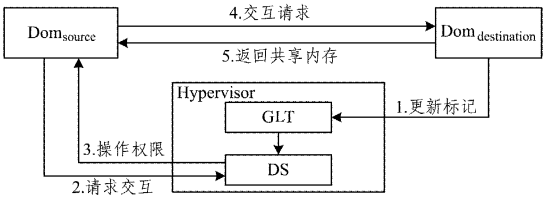


图 4 MDIFC 域间子模型交互规程

域间子模型中包含 3 种域，即 Dom_{source}，Hypervisor 和 $Dom_{destination}$，模型规范的域间详细交互步骤如下：

(1)Dom_{source} 向 Hypervisor 送出域间通信请求(request-StoD)，请求中包含目标域域名及动作需求(actWithD)；

(2)Hypervisor 依据 GLT 中源动作域与目的动作域的标签，通过 DS 作出允许(agreement)和拒绝(rejection)动作的决定；

(3)若 Hypervisor 拒绝动作请求，则向源动作域发送请求拒绝，本次交互结束，否则，执行下一步；

(4)若 Hypervisor 允许动作请求，则向 Dom_{source} 返回同意信息；

(5)Dom_{source} 向 $Dom_{destination}$ 发送动作请求；

(6)$Dom_{destination}$ 查询自身授权表并决定是否接受动作请求，若 $Dom_{destination}$ 拒绝该请求，则向源动作域发送请求拒绝，本次交互结束，否则执行下一步；

(7)$Dom_{destination}$ 向 Dom_{source} 提供共享内存，直到内存回收条件满足，本次交互结束。

下面通过 π 演算描述上述过程，其中 Dom_{source} 与 Hypervisor 内的 DS 间发送请求的信道用 s_ds 表示，Dom_{source} 与 $Dom_{destination}$ 间动作的信道用 s_d 表示，Dom_{source} 与 Hypervisor 内的 DS 间发送请求的信道用 d_ds 表示，虚拟域 Dom_s 与 Hypervisor 内的 GST 间更新标签信息的通道用 Dom_{source}_SetLabel 表示；label 表示虚拟域更新的标记，标记由 $(m+n)$ bit 位组成，前 m 位表示完整型标签，后 n 位表示机密型标签；Dsrc 表示虚拟域域间操作的发起者；Ddes 表示虚拟域域间操作的承受者；act_ID 表示操作和该操作的序列号；agreement_ID 表示同意及该消息的序列号；act 表示执行的操作。

4.1.2 源操作域 Dom_{source} 的描述

设：

$\tilde{a}=\{s\_ds, s\_d, Dom_s\_setLabel, label, Dsrc, Ddes, act\_ID, agreement\_ID, act\}$

$$Dom_{source}(\tilde{a})=\overline{Dom_s\_setLabel}\langle label\rangle.Dom_{source}(\tilde{a})+\overline{s\_ds}\langle Dsrc, Ddes, act\_ID\rangle \cdot s\_ds(message1)$$

$$[message1=agreement\_ID]\,\overline{s_d}\cdot\langle act\rangle\cdot s\_d(message2)\cdot Dom_{source}(\tilde{a}) \tag{9}$$

$$label=I_0\ I_1\cdots I_m C_0 C_1\cdots C_n$$

其中，I_i，C_j 等于 0 或 1，0 表示该虚拟域无此标记，1 表示该虚拟域有此标记。

若信息流由 Dom_{source} 到 $Dom_{destination}$，则其状态转换如下：

$$Dom_d\_label \xrightarrow{act} Dom_d\_label'$$

$$Dom_{d'_{label}}=f_C'(Dom_{destination})\cup f_I'(Dom_{destination}) \tag{10}$$

$$f_C'(Dom_{destination})=f_C(Dom_{destination})\cup(f_C(Dom_{source})-A^{\pm}_{Dom_{source}}) \tag{11}$$

$$f_I'(Dom_{destination})=f_I(Dom_{destination})\cup(f_I(Dom_{source})-A^{\pm}_{Dom_{source}}) \tag{12}$$

4.1.3 管理程序 Hypervisor 的描述

设：$Hypervisor\_label, Dom_1\_label, \cdots, Dom_n\_label$ 分别表示管理程序、虚拟域 1～n 的标记，同理，xxx_abili，xxx_setLabel，xxx_setLabel_cn 结构的符号分别表示 xxx 的能力、其指定更新的新标记、更新标记的通道；getLabel 为查询标签的通道。

$\tilde{b}=\{Hypervisor\_label, Dom_1\_label, \cdots, Dom_n\_label, Hypervisor\_abili, Dom_1\_abili, \cdots, Dom_n\_abili, Dom_s\_abili, Dom_d\_abili, Hypervisor\_setLabel, Dom_1\_setLabel, \cdots, Dom_n\_setLabel, getLabel, label, id, Dom_{id}\_label'\}$

$\tilde{b}'$ 为 $\tilde{b}$ 替换指定 ID 虚拟域后 GST 的状态。

$\tilde{b}'=\{Hypervisor\_label, Dom_1\_label, \cdots, Dom_{id}\_label', \cdots, Dom_n\_label, Hypervisor\_abili, Dom_1\_abili, \cdots, Dom_n\_abili, Dom_s\_abili, Dom_d\_abili, Hypervisor\_setLabel, Dom_1\_setLabel, \cdots, Dom_n\_setLabel, getLabel, label', id', Dom_{id}\_label''\}$

$\tilde{c}_i=\{Dom_{id}\_label, Dom_{id}\_label', Dom_i\_abili, Dom_i\_setLabel, label, id\}$

$$Dom_i\_label=I_{i\_0}\,I_{i\_1}\cdots I_{i\_m}C_{i\_0}C_{i\_1}\cdots C_{i\_n}$$

$$Dom_i\_abili=Ai_{I\_0}Ai_{I\_1}\cdots Ai_{I\_m}Ai_{C\_0}Ai_{C\_1}\cdots Ai_{C\_n}$$

其中，A_{I_i}，A_{C_j} 等于 0，1，2 或 3，0 表示该虚拟域不拥有该标记的任何能力；1 表示该虚拟域拥有此标记的去除能力，如 I_m^-；2 表示该虚拟域拥有此标记的增加能力，如 I_m^+；3 表示该虚拟域同时拥有此标记的去除能力与增加能力，如 $I_m^{\pm}$。I_i (A_{I_i})，C_i (A_{C_i}) 表示依据 label 中第 i 位 I 和 C 判断相应能力：如 I 若为 1，则判断 A_{I_i} 是否具备该标记的增加能力。

$$GLT(\tilde{b})=setLabel\_cn(\tilde{b})\,|\,getLabel(message)\cdot[message=id]\overline{getLabel}\langle Dom_{id}\_label\rangle\cdot GLT(\tilde{b}) \tag{13}$$

$$setLabel\_cn(\tilde{b})=Hypervisor\_setLabel\_cn(\widetilde{c_0})\,|\,Dom_1\_setLabel\_cn(\widetilde{c_1})\,|\cdots|\,Dom_n\_setLabel\_cn(\widetilde{c_n}) \tag{14}$$

$$Labels'-Labels\subseteq A_S^+ \text{ and } Labels-Labels'\subseteq A_S^-$$

$$Dom_i\_setLabel(\widetilde{c_i})=Dom_i\_setLabel(message)\cdot[message=label]\cdot[I_0=I_0(Ai_{I\_0})][I_1=I_1(Ai_{I\_1})]\cdots[I_m=I_m(Ai_{I\_m})][C_0=C_0(Ai_{C\_0})][C_1=C_1(Ai_{C_1})]\cdots[C_n=C_n(Ai_{C\_n})]\cdot GLT(\tilde{b}')+GLT(\tilde{b}) \tag{15}$$

设：$labelChange$ 为标记隐式修改通道；jud_src 和 jud_des 两个函数分别用于判定源和目的虚拟域标签变化的合法

性；Tag_{src_i} 和 $Asrc_{Tag\_i}$ 分别表示源虚拟域第 i 个标签的情况和主体能力；*agreement* 和 *reject* 分别表示系统对操作的许可或拒绝。

$\tilde{d}=\{s\_ds, getLabel, labelChange, Dsrc, Ddes, Dom_{Dsrc}\_label, Dom_{Ddes}\_label, act\_ID, label, agreement, reject\}$

$$jud\_src(Tag_{src\_i}, Asrc_{Tag\_i})=\begin{cases}1, & Tag_{src\_i}=1 \text{ 或 } Asrc_{Tag\_i}=2 \text{ 或 } 43\\ 0, & \text{else}\end{cases} \tag{16}$$

$$jud\_des(Tag_{des\_i}, Ades_{Tag\_i})=\begin{cases}1, & Tag_{des\_i}=1, Ades_{Tag\_i}=2 \text{ 或 } 3\\ 0, & \text{else}\end{cases} \tag{17}$$

$$DS(\tilde{d})=s\_ds(message1, message2, message3)\cdot[message1=Dsrc]\overline{getLabel}\langle Dsrc\rangle\cdot getLabel(message4)\cdot[message4=Dom_{Dsrc}\_label][message2=Ddes]\overline{getLabel}\langle Ddes\rangle\cdot getLabel(message5)\cdot[message5=Dom_{Ddes}\_label]\cdot([jud\_src(Tag_{src\_i}, Asrc_{Tag\_i})=jud\_des(Tag_{des\_i}, Ades_{Tag\_i})]\cdot[message3=act\_ID]\overline{s\_ds}\langle agreement\_ID\rangle\cdot DS(\tilde{d})+[jud\_src(Tag_{src\_i}, Asrc_{Tag\_i})=1]\cdot\overline{s\_ds}\langle agreement\rangle\cdot LabelChange\langle setLabel, Dsrc\rangle\cdot DS(\tilde{d})+[jud\_src(Tag_{des\_i}), Ades_{Tag\_i})=1]\cdot\overline{s\_ds}\langle reject\rangle\cdot DS(\tilde{d})) \tag{18}$$

4.1.4　目的操作域 $Dom_{destination}$ 的描述

设：

$\tilde{e}=\{s\_d, Dom_d\_setLabel, label, act, result\}$

$$Dom_{destination}(\tilde{e})=\overline{Dom_d\_setLabel}\langle label\rangle+s\_d(message)\cdot[message=act]\overline{s\_d}\langle result\rangle\cdot Dom_{destination}(\tilde{e}) \tag{19}$$

4.1.5　按需受控子模型交互规格的描述

如图 5 所示，按需受控子模型的设计思想是：虚拟域通常处于公开态，当从外部获取敏感信息后，进入受控态，按需(on demand)进行信息流跟踪，当虚拟域中无敏感信息后退出受控态，回到公开态。因此在模型中，提出用代理 Agent 与本地标签表 GST 组成系统安全核来保证虚拟域的安全。在模型中，当进程从外设中调用信息并且操作通过系统安全核时，系统安全核将检测所读取的信息是否为带标记的敏感数据。当数据为敏感数据时，管理程序将暂停该进程所在虚拟域，并制作该虚拟域副本。副本以原虚拟域为基础，在操作系统中添加了引用监视器 RM、安全服务器 SecServer 及标记状态服务器 LSM。副本生成后，将读取敏感数据并沿原进程操作直至内存中不再有敏感数据。在副本中，添加的 3 个模块主要完成两个工作：1)当进程对敏感数据执行操作时，由 RM 拦截操作，并由 SecServer 依据 LSM 中存储的标记状态对其进行权限判定，当操作满足既定的安全策略时，进程完成操作；2)当进程完成操作时，跟踪敏感数据的流向，并为敏感数据影响的数据添加与敏感数据一致的标记。当敏感操作完成后，由系统安全核将结果返回给原始进程。其操作过程如下：

(1)Process 通过前端驱动向管理程序 Hypervisor 发出 I/O 请求(getData)获取数据。

(2)Hypervisor 获取到数据后将数据放入 I/O 环中，由 Agent 在 LLT 中查询数据是否带有标签：若数据有标签，则暂停 Process 所在虚拟域，并生成该虚拟域副本 Controlled_Dom，进入(3)；若数据无标签，则由 Process 读取，本次操作结束。

(3)进程副本 Controlled_Process 在 Controlled_Dom 中运行，由 Controlled_Process 从 I/O 环中读取敏感数据，并进行处理，直至系统内存中不再有敏感数据，暂停Controlled_Dom，并将其返回 Agent。不再有敏感数据是指 Controlled_Dom 中所有的敏感数据都得到了释放，因为本模型的标签具有传播能力，在该敏感数据的影响下生成的数据都将拥有其标签，所以在模型的实现时需要监视敏感数据的传播过程。同时还需注意的是，除了需要监视初次引入的敏感数据外，在敏感数据处理的后续过程中可能引入其他敏感数据，其他进程引入的敏感数据都需要被监控。

(4)Agent 将收到的 Controlled_Dom 状态更新到 Dom 中，以保证操作的连贯性。

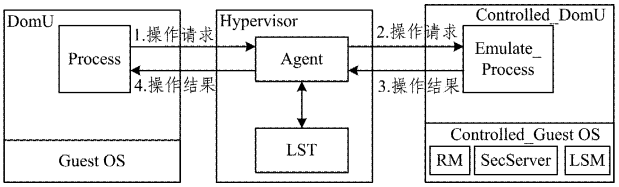


图 5　MDIFC 按需受控子模型的交互规程

4.2　系统无干扰性分析

对于并行系统，学术界已提出许多安全定义，如无秘密数据流入公开数据。而引入的无干扰性质正是多级信息系统信息流安全的形式化定义，即在两个不同的高级环境中插入一个进程时是否发生信息由高级向低级观察者的泄露。

使用 π 演算建模的原因是其能够准确描述并行系统的交互行为，除此外 π 演算拥有证明系统无干扰性的工具。如 PicNic[20] 是一款专门用于证明 π 演算无干扰安全性的工具，其基于文献[21]提出的验证方法进行验证。

本文系统验证环境为 tools version/1.0，os/Ubuntu 12.04，cpu/Intel core i5-4200U 1.60GHz×4，RAM/4GB(工具的状态空间设置为 100)并显示调试信息。

限于篇幅，本文仅给出域间模型的源虚拟域的 π 演算描述：

AGENT

Domsource(*s_ds*, *s_d*, *one*, *dsrc*, *ddes*, *act*, *agreement*, *check1*, *check2*):=*s_ds*⟨*dsrc*⟩. *s_ds*(*message1*:*H*[]). if *message1*=*check1* then {*s_ds*⟨*ddes*⟩. *s_ds*(*message2*:*H*[]). if *message2*=*check2* then {*s_ds*⟨*act*⟩. *s_ds*(*message3*:*H*[]). if *message3*=*agreement* then {*s_d*⟨*agreement*⟩. *s_d*(*message*:*H*[]). 0} else {0}} else {0}} else {0}

实验结果如下，经程序验证，系统满足无干扰性。

虚拟域 Dom 描述：

$\tilde{a}=\{d\_a, a\_l, getData, null, memory, CPU, Stop, Controlled\}$；

$\tilde{a}'=\{d\_a, a\_e, a\_l, getData, Data, memory, CPU, Stop, Controlled\}$

$Dom(\tilde{a})=\overline{d\_a}\langle getData\rangle. d\_a(message1). ([message1=Stop]. d\_a(message2). Dom(message2)+Dom(\tilde{a}'))$

其中，memory 和 CPU 分别表示虚拟域 Dom 内存和 CPU。

代理 Agent 描述如下：

$\tilde{b}=\{d\_a, a\_e, a\_l, Data, Stop, Sensitive, memory, CPU\}$

$Agent(\tilde{b})=d\_a(message1).[message1=getData](\overline{a_l})\langle Data\rangle.\ a\_l(message2)([message2=Sensitive].(\overline{d\_a})\langle Stop\rangle.\overline{a\_e}\langle Controlled\rangle.\overline{a\_e}\langle\tilde{a},Data\rangle.a\_e(message3).\overline{d\_a}\langle message3\rangle+\overline{d\_a}\langle Data\rangle).\ Agent(\tilde{b})$

本地标签表 LLT 描述如下：

$\tilde{c}=\{a\_l,privilegeData,publicData,Sensitive,Public\}$

$privilegeData=\{Data,privilegeData\_1,\cdots,privilegeData\_n\}$

$publicData=\{publicData\_1,publicData\_2,\cdots,publicData\_n\}$

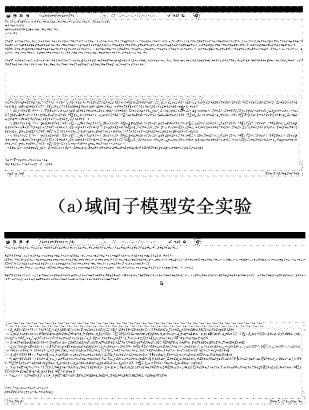
$LLT(\tilde{c})=a\_l(message1).([message1=Data](\overline{a_l})\langle Sensitive\rangle+\overline{a\_l}\langle Public\rangle).LLT(\tilde{c})$

虚拟域副本 *Controlled_Dom* 描述如下：

$\tilde{d}=\{d\_a,a\_e,a\_l,null,Stop,Sensitive,memory',CPU'\}$

$Controlled\_Dom(\tilde{d})=a\_e(message1).[message1=Controlled]\cdots\overline{a}_e\langle\tilde{d}\rangle.Controlled\_Dom(\tilde{d})$

域间子模型安全实验与按需域内子模型安全实验如图 6 所示。



(a)域间子模型安全实验

(b)按需域内子模型安全实验

图 6

5 示例分析及可用性比较

在云计算中，IaaS 服务是云计算服务的重要组成部分，而 IaaS 服务正是一种向用户提供虚拟域的服务。在云计算的共享特性为其赢得广大用户的同时，其同驻引发的问题也日渐成为用户与供应商共同关心的重点安全问题。MDIFC 是一个关注敏感信息安全的分布式信息流控制模型。在虚拟域操作或者计算敏感数据时，该模型能跟踪用户数据传播，并执行适当的安全策略来机制阻止非授权的访问。因此，本节从两方面分析说明 MDIFC 在可用性方面比传统的信息流模型有明显的提高。

一方面，信息流跟踪开销的比较。传统系统中的信息流控制实践表明，全面且最大程度地保护信息安全将会带来巨大的系统开销，往往只能在实验室中进行实现，难以应用于实际生产生活中。而本文模型通过在敏感数据进出系统时动态地使用/关闭信息流跟踪技术，避免了系统资源的浪费，特别是在云计算环境下，众多虚拟机处于同一物理资源上，通过按需地使用信息流跟踪机制无疑将大大提高系统效率。

另一方面，系统安全性的比较。正如本节开始所述，同驻安全问题正成为用户与供应商在提高 IaaS 服务使用率方面的阻碍，对此，本文通过一个典型的应用场景来说明本模型的安全性优于其他模型。图 7 给出了一个面向 IAAS 的云平台的应用场景：云平台由计算节点 A、计算节点 B 和控制节点 C 组成。计算节点 A 上有虚拟域 CorpA_Center，CorpA_WorkspaceX，CorpA_WorkspaceY，CorpA_Software，CorpA_Antivirus，CoprA_Backup。虚拟域 CorpA_Center 是公司在云系统上的核心，通过配置 Domain Config 文件来管理其他虚拟域。虚拟域 CorpA_WorkspaceX，CorpA_WorkspaceY 供公司员工使用的两台虚拟机，员工可利用该虚拟机进行办公，员工会处理一些敏感信息，如 CorpA_WorkspaceX Data 是虚拟域 CorpA_WorkspaceX 中的敏感信息。虚拟域 CorpA_Software 是公司的云端软件库，负责为公司提供安全的软件，软件来源于公司自行开发与网络下载。虚拟域 CorpA_Antivirus 是云查杀服务器，为公司名下的虚拟域提供杀毒与木马检测服务，需要定时更新病毒库。Download Data 是从互联网中下载的数据。虚拟域 CorpA_Backup 是公司的备份服务器，存有公司所有虚拟域数据的备份。计算节点 B 上有虚拟域 CorpB_WorkspaceX，CorpB_WorkspaceY，CorpC_WorkspaceX。控制节点 C 上有 CorpC_WorkspaceX。现有 3 个公司(CorpA，CorpB，CorpC)使用该云平台，本文以虚拟域名称前缀的方式说明虚拟域的所有权。

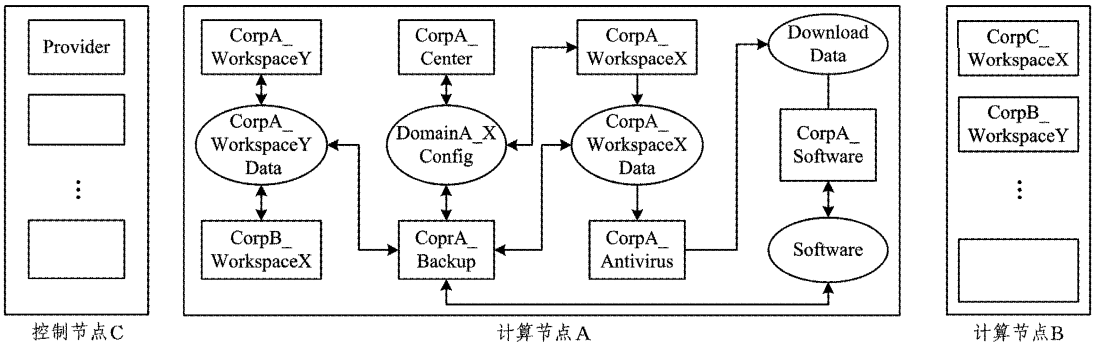


图 7 一个典型的基于 IaaS 的云平台应用场景

接下来说明示例的安全需求：

(1)公司 A 与公司 C 存在利益冲突，其名下的虚拟域之间无法交互，也无法位于同一计算节点上；

(2)公司 B 与公司 A 和公司 C 都不存在利益冲突，CorpB_WorkspaceX在与 CorpA_WorkspaceY 交互信息后，无法与公司 C 名下的虚拟域进行交互，CorpA_Center 可恢复 CorpB_

WorkspaceX 与公司 C 名下的虚拟域交互能力；

(3)公司 A,B,C 的任何数据都不会流入云服务供应商下的虚拟域；

(4)允许 CorpA_Center 修改 Domain Config，各虚拟域能修改存放于 Center 中的自身 Domain Config，但是当虚拟域下载了 Download Data 后，将失去修改 Domain Config 的能力；

(5)虚拟域下载了 Download Data 后失去了修改 Domain Config 的能力，只能在病毒查杀后才可恢复修改能力；

(6)在经 CorpA_WorkspaceX 授权前，除 CorpA_Backup，CorpA_WorkspaceX 外，其他虚拟域无法访问 CorpA_WorkspaceX Data，CorpA_WorkspaceX 拥有该数据的传播权限；

(7)CorpA_Backup 恢复某虚拟域数据时不会泄露其他虚拟域信息。

下面将通过比较各传统模型在此场景下的应用来说明本文模型的可用性。

(1)经典的信息流控制模型

经典的信息流控制模型是 BLP[22] 与 Biba[23] 的结合，因此系统中主、客体的安全级固定，无法满足需求(4)和(5)。除此之外，传统的信息流控制模型由于不具备标签变化的能力，虽然较好地保障了系统安全性，但也使得系统中的信息流动变得困难，这样的问题在云计算环境下更为凸显。

(2)分布式信息流控制模型

分布式信息流控制模型可以通过对同一冲突域的企业使用不同的标签来实现中国墙策略，并通过对标签的重定义机制实现上述标签动态变化的需求。但标签内容重定义在实际应用中会造成较大的开销。而对于中国墙策略的实现，还需要管理人员对属于中国墙的标签进行特别管理，以防止同一冲突域的企业虚拟域拥有相同的标签，从而造成系统的安全隐患。

(3)广义污点传播模型

广义污点传播模型主要考虑传统操作系统，因此受限于传统操作系统并不存在多组织共享的问题，若要使之适应云计算的共享特性，则需要针对需求(1)、(2)和(3)作出较大改变。

依据安全需求，首先依据各虚拟域属公司分配中国墙标记，其中由于互联网数据的来源未知，因此不对它设中国墙标记。为保护 CorpA_WorkspaceX 的 CorpA_WorkspaceX Data，我们为其及 CorpA_WorkspaceX BackupC 创建机密性标记 C，从而掌握敏感信息流向，对其进行安全控制。为防止 Download Data 破坏系统的安全性，对从互联网流入的数据统一附上完整性标记 I。表 1、表 2 所列为 MDIFC 在此情境下系统中各实体的能力和初始标记设置。

表 1　应用示例中各主体在系统中的能力和初始标记设置

Subject	Integrity label	Confidentiality label (input/output)	Chinese wall label	cability
CorpA_Center	{}	{}/{}	{CorpA}	$\{CorpA^{-}\}$
CorpA_WorkspaceX	{}	{C}/{}	{CorpA}	$\{C^{\pm}\}$
CorpA_WorkspaceY	{}	{}/{}	{CorpA}	{}
CorpA_Software	{}	{}/{}	{CorpA}	{}
CorpA_Antivirus	{}	{}/{}	{CorpA}	$\{I^{-}\}$
CoprA_Backup	{}	{C}/{}	{CorpA}	{}
CorpB_WorkspaceX	{}	{}/{}	{CorpB}	{}
CorpB_WorkspaceY	{}	{}/{}	{CorpB}	{}
CorpC_WorkspaceX	{}	{}/{}	{CorpC}	{}
Provider	{}	{}/{}	{Provider}	{}

表 2　应用示例中各客体在系统中的能力和初始标记设置

Object	Integrity label	Confidentiality label	Chinese wall label
Domain Config	{}	{}	{CorpA}
Download Data	{I}	{}	{}
Software	{}	{}	{CorpA}
CorpA_WorkspaceX Backup	{}	{C}	{CorpA}
CorpA_WorkspaceX Data	{}	{C}	{CorpA}
CorpA_WorkspaceY Data	{}	{}	{CorpA}
CorpB_WorkspaceX Data	{}	{}	{CorpB}

中国墙标记的冲突关系：

Cf(CorpA)＝{CorpC,Provider}；

Cf(CorpB)＝{Provider}；

Cf(CorpC)＝{CorpA,Provider}；

Cf(Provider)＝{CorpA,CorpB,CorpC,…}。

从表中可以看出，CorpA_Center 带有中国墙标记CorpA，因此，其生成的各虚拟域都将拥有此标记，从而保证了公司旗下的虚拟域始终有归属；而 $CorpA^{-}$ 则使得其他公司的虚拟域在结束与 A 公司的交互后可恢复其与其他虚拟域的交互，不受公司 A 冲突策略的限制；CorpA_WorkspaceX 是敏感信息 CorpA_WorkspaceX Data 的拥有者，因此设置其能力为 $\{C^{\pm}\}$，表示虚拟域可以传播该敏感信息；CorpA_Antivirus 拥有 I^{-} 的能力，能够在对危险文件进行检查后去除其完整性标记，提高其完整级。

以下是 MDIFC 对各需求的可满足性：

(1)因为 Cf(CorpA)＝{CorpC,Provider}，Cf(CorpC)＝{CorpA,Provider}，所以公司 A 与公司 C 的虚拟域之间无法交互，也无法位于同一计算节点上。

(2)CorpB_WorkspaceX 在与 CorpA_WorkspaceY 交互信息后，CorpB_WorkspaceX 的中国墙标记中添加了 CoprA，因此 CorpB_WorkspaceX 无法与公司 C 名下的虚拟域进行交互。CorpA_Center 的 $CorpA^{-}$ 能去除 CorpA 标记，从而恢复 CorpB_WorkspaceX 与公司 C 名下虚拟域的交互能力。

(3)公司 A,B,C 的任何数据都不会流入云服务供应商下的虚拟域，Cf(Provider)＝{CorpA,CorpB,CorpC,…}，所有虚拟域在被创建后都会带上 Corp 标记，保证了新生成虚拟域无法与供应商 Provider 下的虚拟域进行交互；同时，系统中虚拟域的中国墙标记无法为空，使得公司虚拟域始终无法与 Provider 下的虚拟域交互。

(4)允许 CorpA_Center 修改 Domain Config，各虚拟域能修改存放于 Center 中自身的 Domain Config，但是当虚拟域下载了 Download Data 后，将失去修改 Domain Config 的能力。

(5)虚拟域下载了 Download Data 后，由于 Download Data 所具有的完整性标记 I 使得虚拟域的完整级下降，失去了修改 Domain Config 的能力，因此虚拟域只有通过 CorpA_Antivirus 对 Download Data 及与之交互而带上标记的数据进行病毒查杀后，由 CorpA_Antivirus 去除 Download Data 及相关数据携带的完整性标记，从而提高虚拟域的完整级，恢复修改能力。

(6)在经 CorpA_WorkspaceX 授权前，除 CorpA_Backup 和 CorpA_WorkspaceX 外，其他虚拟域为低机密级，故无法访

问 CorpA_WorkspaceX Data，$C^{\pm}$ 的能力保证了只有 CorpA_WorkspaceX 拥有该数据的传播权限。

(7)CorpA_Backup 拥有高输入安全级，故公司 A 下的虚拟域都可将数据备份至该服务器，当没有敏感信息虚拟域需要恢复数据时，CorpA_Backup 可以直接从硬盘中获取备份数据并进行恢复；当需要恢复的虚拟域含有敏感数据时，读入的敏感数据提高了 CorpA_Backup 的输出安全级，此时 CorpA_Backup 只会将敏感数据输出至安全级与之匹配的虚拟域中，当还有普通虚拟域需要进行数据恢复时，若 CorpA_Backup 内存中还保留有敏感数据，则此时无法进行恢复，从而保证了敏感信息的安全性；当 CorpA_Backup 中敏感数据都被清除后，方可进行恢复。

从上述讨论可以看出，本文模型在云计算环境下的可用性较诸多传统模型有很大提高，其系统安全性也得到了较好的保证。

结束语 本文针对虚拟技术易于复制重建的特性，提出了更具灵活性的按需污点传播模型，并首创性地将中国墙模型与分布式信息流模型进行有机结合。本文在分析模型安全性时，利用 π 演算对模型进行了形式化描述，并给出了管理程序 Hypervisor、操作域等主体行为的形式化语义，进而使用工具 PicNic 完成了系统无干扰性的证明。最后，通过一个云计算应用场景说明了本文模型较现有模型更适应于云计算环境。

本文在域内使用现有的 GTPM 模型，而该模型只是单纯地针对操作系统设计，并未考虑虚拟化这一特性，因此下一步工作将着手研究如何对其进行改进，提高模型的能力。

参 考 文 献

[1] FENG D G, ZHANG M, ZHANG Y, et al. Study on Cloud Computing Security[J]. Journal of Software, 2011, 22(1): 71-83. (in Chinese)
冯登国，张敏，张妍，等. 云计算安全研究[J]. 软件学报，2011，22(1)：71-83.

[2] MYERS A C, LISKOV B. A decentralized model for information flow control[J]. Acm Sigops Operating Systems Review, 1997, 31(5): 129-142.

[3] TUPAKULA U, VARADHARAJAN V. Trust Enhanced Security for Tenant Transactions in the Cloud Environment[J]. Computer Journal, 2014, 58(10): 2388-2403.

[4] ZHANG H F, ZUO X D, LIU G. An Information Flow Security Control Method Based on Virtualization Technology[C]// Information Security & Technology. China Center of Information Industry Development. Beijing, 2013: 46-49. (in Chinese)
张怀方，左晓栋，刘刚. 基于虚拟化技术的信息流控制方法[C]// 2013中国信息安全技术大会(CISTC 2013). 暨工业控制系统安全发展高峰论坛论文集. 北京：中国电子信息产业发展研究院，2013：46-49.

[5] PASQUIER J M, BACON J, EYERS D. FlowK: Information Flow Control for the Cloud[C]// International Conference on Cloud Computing Technology and Science, 2014. 2014: 70-77.

[6] PASQUIER J M, BACON J, SHAND B. FlowR: Aspect oriented programming for information flow control in ruby[C]// ACM International Conference on Modularity. 2014: 37-48.

[7] BACON J, EYERS D, PASQUIER J M, et al. Information Flow Control for Secure Cloud Computing[J]. IEEE Transactions on Network & Service Management, 2014, 11(1): 76-89.

[8] BREWER D F C, NASH M J. The Chinese Wall Security Policy [C]// IEEE Symposium on Security and Privacy, 1989. IEEE, 1989: 206-214.

[9] LIN T Y. Chinese wall security policy - an aggressive model [C]// Computer Security Applications Conference. 1990: 282-289.

[10] GUPTA V. Chinese Wall Security Policy[D]. San Jose: San Jose State University. 2009.

[11] KATSUNO Y, WATANABE Y, FURUICHI S, et al. Chinese-wall process confinement for practical distributed coalitions[C]// ACM Symposium on Access Control MODELS and Technologies, Sophia Antipolis(SACMAT 2007). France, 2007: 225-234.

[12] JAEGER T, SAILER R, SREENIVASAN Y. Managing the risk of covert information flows in virtual machine systems[C]// ACM Symposium on Access Control MODELS and Technologies, Sophia Antipolis(SACMAT 2007). France, 2007: 81-90.

[13] CHENG G, JIN H, ZOU D Q, et al. Chinese wall model based on dynamic alliance[J]. Journal on Communications, 2009, 30(11): 93-100. (in Chinese)
程戈，金海，邹德清，等. 基于动态联盟关系的中国墙模型研究[J]. 通信学报，2009，30(11)：93-100.

[14] JIANG L, HE R Y, WEI Y F. Chinese Wall Model Based on Dynamic Divided-set[J]. Computer Science, 2015, 42(1): 159-163. (in Chinese)
姜路，鹤荣育，魏彦芬. 基于动态分集的中国墙模型研究[J]. 计算机科学，2015，42(1)：159-163.

[15] YANG Z, YIN L H, DUAN M Y, et al. Generalized Taint Propagation Model for Access Control in Operation Systems[J]. Journal of Software, 2012, 23(6): 1602-1619. (in Chinese)
杨智，殷丽华，段洣毅，等. 基于广义污点传播模型的操作系统访问控制[J]. 软件学报，2012，23(6)：1602-1619.

[16] MILNER R, PARROW J, WALKER D. A calculus of mobile processes, II[J]. Information and Computation, 1992, 100(1): 41-77.

[17] MILNER R, PARROW J, WALKER D. Modal logics for mobile processes[J]. Theoretical Computer Science, 1993, 114(1): 149-171.

[18] MILNER R. Communicating and mobile systems: the π-calculus [M]. Cambridge University Press, 1999.

[19] MILNER R. Lectures on a calculus for communicating systems: Seminar on Concurrency[M]. Springer Berlin Heidelberg. 1985: 197-220.

[20] CRAFA S, MIO M, MICULAN M, et al. PicNIc-Pi-calculus non-interference checker[C]// International Conference on Application of Concurrency to System Design. 2008: 33-38.

[21] CRAFA S, ROSSI S. P-congruences as non-interference for the pi-calculus[C]// ACM Workshop on Formal Methods in Security Engineering(Fmse 2006). Alexandria, Va, USA, 2006: 13-22.

[22] PASQUIER T F J M, BACON J, EYERS D. FlowK: Information Flow Control for the Cloud[C]// International Conference on Cloud Computing Technology and Science. 2014: 70-77.

[23] Biba K J. Integrity Considerations for Secure Computer System [OL]. http://www.cerias.purdue.edu/apps/reports-and-papers/view/2834.