

一种云平台可信性分析模型建立方法^{*}

赵 波^{1,2}, 戴忠华^{1,2,3}, 向 驥^{1,2}, 陶 威^{1,2}

¹(武汉大学 计算机学院, 湖北 武汉 430072)

²(空天信息安全与可信计算教育部重点实验室(武汉大学), 湖北 武汉 430072)

³(中国信息安全测评中心, 北京 100085)

通讯作者: 戴忠华, E-mail: zhonghuadai@163.com



摘 要: 如何使得用户信任云服务提供商及其云平台, 是云计算普及的关键因素之一. 针对目前云平台可信性所包含的内容与分析评价依据尚不完善的现状, 且缺乏从理论层次对于云平台的部分可信属性进行分析与评估方法的问题, 首先对云平台的可信性进行定义, 并结合国内外相关云安全标准与可信性规范以及作者的理解, 明确了云平台可信性的子属性与具体分析内容, 从而明确了所提出模型的适用范围、分析目的以及依据. 在此基础上, 提出模型建立方法. 该方法以标记变迁系统作为操作语义描述工具, 从云平台内部组件交互过程出发, 将平台对外提供服务过程刻画为用户与云的交互以及云平台内部实体间的交互, 并利用模型分析检测工具 Kronos 从可用、可靠、安全等多个角度对平台内部状态变化过程进行分析. 分析结果不但能够发现已知的可信性问题, 还发现了一些未知的隐患, 说明了模型建立方法的有效性, 并为如何评价云平台的可信性, 进而构建可信云提供了理论支撑.

关键词: 云平台; 可信性分析; 平台建模; 标记变迁系统; Kronos

中图法分类号: TP301

中文引用格式: 赵波, 戴中华, 向驥, 陶威. 一种云平台可信性分析模型建立方法. 软件学报, 2016, 27(6): 1349–1365. <http://www.jos.org.cn/1000-9825/4994.htm>

英文引用格式: Zhao B, Dai ZH, Xiang S, Tao W. Model constructing method for analyzing the trusty of cloud. Ruan Jian Xue Bao/Journal of Software, 2016, 27(6): 1349–1365 (in Chinese). <http://www.jos.org.cn/1000-9825/4994.htm>

Model Constructing Method for Analyzing the Trusty of Cloud

ZHAO Bo^{1,2}, DAI Zhong-Hua^{1,2,3}, XIANG Shuang^{1,2}, TAO Wei^{1,2}

¹(School of Computer Science, Wuhan University, Wuhan 430072, China)

²(Key Laboratory of Aerospace Information Security and Trusted Computing Ministry of Education (Wuhan University), Wuhan 430072, China)

³(China Information Technology Security Evaluation Center, Beijing 100085, China)

Abstract: Trust is one of the key factors that affect people's preferences in choosing cloud computing. However, the content and the evaluation basis of cloud trust are still not perfect, and there are lack of strict theoretical analysis and evaluation methods on its dynamic properties. For the status quo, this paper first defines cloud trust, and describes in details its sub-properties according to related international cloud security standards and trust specifications. Then, the basis, purpose, and scope of a model for analyzing cloud trust is presented. Next, to the process for building this model is described. Using LTS as an operational semantics profiling tool, starting from

* 基金项目: 国家重点基础研究发展计划(973)(2014CB340600); 国家高技术研究发展计划(863)(2015AA016002); 国家自然科学基金(91118003, 61173138, 61272452, 61332019)

Foundation item: National Program on Key Basic Research Project of China (973) (2014CB340600); National High-Tech R&D Program of China (863) (2015AA016002); National Natural Science Foundation of China (91118003, 61173138, 61272452, 61332019)

收稿时间: 2015-08-12; 修改时间: 2015-10-09; 采用时间: 2015-12-05; jos 在线出版时间: 2016-01-21

CNKI 网络优先出版: 2016-01-22 11:01:38, <http://www.cnki.net/kcms/detail/11.2560.TP.20160122.1101.010.html>

the interactive process of internal components in cloud, the cloud service is depicted as the interactions between users and cloud, and interactions among entities inside cloud. Finally, the model analysis tool, Kronos is used to analyze system state variations in service provision from multiple perspectives including availability, reliability and security. The analysis results demonstrate that the presented model can find not only known trusty problems but also unknown risks, which indicates the model is effective, and can provide theoretical support for trusted cloud construction.

Key words: cloud computing; trust property; platform modeling; labelled transition system; Kronos

云计算具有资源高度集中、动态可扩展、外包服务等特点,使得用户丧失了对数据及资源的可见性与可控性,信任问题凸显.根据埃森哲的调研报告显示,用户对云服务缺乏信任是拒绝使用云服务的重要因素^[1].云平台作为云服务的重要载体,如何对云平台安全展开分析与评价,使用户能够信任云服务提供商以及其所提供的云平台,是云计算推广和普及的关键因素之一.

学术界对于信任与可信的含义存在不同的理解,国际标准化组织与国际电子技术委员会以及 TCG 均从行为的可预测性或预期性来定义可信^[2,3];IEEE CS 可信计算与容错技术委员会(IEEE Computer Society Technical Committee on Dependable Computing and Fault Tolerance)从可信赖角度来描述可信性,强调系统的可靠性与可用性^[4];文献[5]提出了可信 \approx 安全+可靠的学术观点,提出可信性必须综合考虑系统的可靠性、可用性、主体行为与信息的安全性.上述定义与描述中,文献[3]的定义难以进行定性与定量的分析,文献[4]偏向于系统可靠性描述,而文献[5]综合考虑了可用、可靠与安全多方面的影响.结合近年来云服务出现的服务故障事故,这其中既有安全事故的发生,例如 CSDN 云数据中心隐私泄露事件以及 sony 网络受云平台攻击事件,还包括一些服务设计缺陷而导致的可靠性事故,例如亚马逊云服务大面积中断事故,因此我们认为:文献[5]的学术观点较能符合云平台的特点,只有对可用属性、可靠属性和安全属性等多项属性进行综合的分析研究,才能对云平台可信性进行更准确的评价.

目前已经有了一些对云环境进行建模分析的方法,并发现了部分安全问题^[6,7],这些研究为进一步实现云平台的安全增强方案提供了理论指导与现实基础.但总结来看,现有的对于云平台的建模与分析方法存在以下问题和挑战:

- 目前,云平台可信性的定义以及其所包含的内容尚不完善的,这就导致了目前的分析方法大多缺乏明确的分析目的、适用范围与分析依据;
- 现有的大部分分析方法仅关注安全性分析,且在云环境中提及的安全性更多是数据与隐私安全^[8-11],不能分析发现因平台设计或执行缺陷导致的故障.根据文献[5]中对可信的描述,对安全性的分析仅仅是可信性的部分体现,不能充分让用户信任云平台;
- 目前,对于云环境的建模方法,通常以进程代数^[6]、图论^[7]或贝叶斯网络^[11]为基础,建立的大多是对云环境的静态描述模型,不能体现平台提供服务时的状态变化过程,不适用于对可信性中,可用与可靠等动态属性的分析.

针对上述问题,本文在结合云平台特点的基础上,首先定义云环境的可信性,并明确其包括的具体内容,以明确本文的分析依据、范围以及目的;随后提出一种云平台建模方法,以标记变迁系统(LTS)为语义描述工具,模型建立过程依赖于云平台的组件,以便完整反映用户使用云服务时组件的交互过程以及系统的状态变迁过程;最后,通过模型分析检测工具对平台可用性、环境稳定性以及组件的安全约束能力等可信性属性进行定性和定量的验证,进而实现在实际部署时做出相应的安全应对,尽可能减少云平台故障发生的可能,提高云环境的可信性.

本文的主要贡献在于较为完善地明确了云平台可信性分析内容与评判标准,建立了对平台可信性部分动态属性进行统一综合描述分析的模型,提出的建模方法能够兼顾可用性、可靠性以及安全执行能力等多方面的分析需求,解决了对于可信性中部分属性形式化分析困难的问题.

本文第 1 节分析国内外相关工作.第 2 节给出云平台的基本架构与硬件逻辑构成,并对云平台进行抽象描述,还对用户基础服务种类进行抽象划分.第 3 节首先给出可信性定义,并详细说明本文模型的分析依据、适用

范围以及目的;随后,利用 LTS 相关理论构建语义模型,构建云平台的运行状态模型,并针对其状态空间过于复杂的问题进行约减.第 4 节结合第 3.1 节的定义提出分析思路,采用模型分析检测工具 Kronos 进行可信性分析.第 5 节总结全文,结合第 4 节分析得出的隐患,讨论下一步的研究方向.

1 相关工作

在云环境的可信性分析研究方面,文献[6]利用形式化方法对云环境以及构成云的组件进行形式化描述,从云环境的可用性和安全性方面给出了严格的数学定义和证明;在此基础上,还给出了云环境扩展、限制以及更新的严格定义,为进一步分析奠定了基础.但是文中建立的模型是一种静态描述模型,不适用于对一个动态系统进行可靠性分析.

文献[12]提出一种云计算模式下的运行环境可信性验证方法,通过引入可信第三方,对云平台的可信性进行远程验证和审计,能够分析发现针对内存和文件系统的威胁.但是内存和文件系统仅是可信性中安全属性的一小部分,方法不能完全反映服务的可信性.

在服务与其依赖平台关联的分析与验证方面,现有的研究大多集中于 Web 与物联网,其方法都是以进程代数、自动机或 Petri 网为理论基础构建服务提供时的平台运行模型,并利用模型验证工具对模型进行进一步分析与验证^[13-18].在这些研究中,状态机模型一般是将服务过程映射为对应的形式化描述,特别适用于强调服务提供时平台组件间的交互行为与关系,在动态反映建模对象的行为和过程方面具有优势^[16-18].

文献[16]提出了 Web 服务属性分析和验证模型,通过将服务组合进程转换为自动机,并进一步将自动机解释为能被模型检测工具 SPIN 所接受的输入语言来验证的相关属性;文献[17]通过扩展自动机的逻辑表达式,加强了对于服务行为的描述,并能够形式化描述服务的消息序列,最终实现服务行为的准确匹配;文献[18]专注于如何对物联网服务进行建模与分析,通过引入时间自动机,增加了对于时间属性的刻画,将物联网服务行为建模为其与相关平台实体的交互过程,解决了连续性建模与验证的问题.

上述研究都将状态机理论用于 Web 或物联网的建模与验证,而 Web、物联网与云计算的本质都是用户通过网络来使用系统功能,其性质是可类比的,因此,状态机理论也适用于对于云平台的建模与验证.

与相关研究工作相比,本文更关注于如何结合云环境的特点建立对云平台进行动态的模型描述,以便对平台的可信性进行分析与验证.

2 云平台的基础逻辑构成

云平台是一个构造复杂的系统,外部包括数量庞大的用户,内部包含海量的控制、计算与存储实体以及软件,这些实体之间彼此互相关联,依存度极高,对如此之多且关联紧密的实体进行建模是困难且不现实的,因此,首先有必要对实体按照逻辑功能进行划分,并对其进行进一步的抽象.

2.1 云平台的逻辑构成

现有的云平台按其透明度又分为开源和不开源两种.不开源的云环境往往不公开其内部细节^[19],我们无法得知其部署方式与内部结构,因此本文不做讨论.因此,本文的研究重点是以开源项目为基础来进行构建和部署的云平台,例如惠普云以及 At&T 的 Cloud Architect^[20,21].开源云平台包括了基础硬件、虚拟化相关软件、基础设施层管理软件(Eucalyptus, OpenStack 以及 OpenNebula)^[22-24]以及用户等主要组件,其基础框架与运行过程基本类似,如图 1 所示.

从逻辑上看,云平台的基础框架主要包括 4 个主要部分,每一部分都由若干最小功能组件构成:

- (1) 接入用户:既包括普通用户(简称租户),又包括云提供商的管理员用户(简称管理员),他们是云平台的管理者和使用者,通过管理中心接入云中,是使用和管理系统资源和功能的主体;
- (2) 基础设施层管理中心:既是云对外服务的接口,也是云资源的统一管理、监控和调度中心.标准的云管理中心又包括接入服务器、调度服务器、元数据库以及监控服务器.其中,监控服务器负责监控云环

境的运行状态并记录日志,不是提供的云服务的必备组件,因此,在进行建模分析时可忽略这一模块;

- (3) 计算资源中心:由逻辑上可管理的物理计算资源组成,是用户管理和使用的虚拟机的宿主机,同时被平台调用.云资源计算中心又包括计算集群控制节点以及控制节点下属的若干台计算节点,在计算节点上,安装虚拟机监控器、虚拟机以及其他应用软件;
- (4) 存储资源中心:由逻辑上可管理的物理存储资源组成,这里的存储资源是指用于存储虚拟机模板和用户虚拟机镜像的统一管理存储节点,不用于存储一般性用户数据.一个云资源存储中心同样包括存储集群控制节点以及控制节点下属的若干台存储节点.存储节点上存放的是虚拟机镜像与虚拟机模板.

上述架构既包括了外部用户,又包括了内部控制、计算与存储组件,基本可以很好地刻画云平台运行时的参与主体与客体,具有较好的完备性.

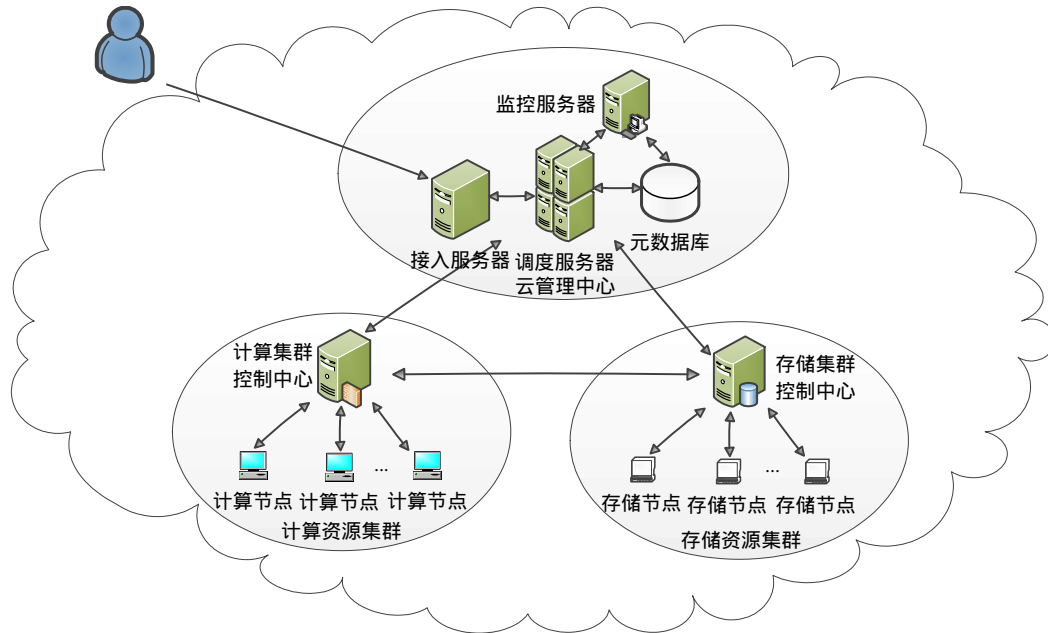


Fig 1 Infrastructure of cloud platform

图 1 云平台基础框架

2.2 云平台的抽象描述

基于第 2.1 节中描述的基本框架与逻辑构成,我们将云平台做如下抽象定义:

定义 1. 云平台是三元组 (S, Ob, Op) , 其中, S 代表系统主体集, Ob 代表系统客体集, Op 代表用户服务功能集.

定义 2. S 为系统主体集合, 系统主体包括所有管理员用户与普通用户, 为了方便表述, 下文中将系统主体统称为用户, 普通用户统称为租户, 记为 $S = Admins \cup Users$.

定义 3. Ob 为系统客体集合, 系统客体为云平台内部的所有组件, 记为

$$Ob = MC(\text{manage center}) \cup CRC(\text{Compter Resource Center}) \cup SRC(\text{Storage Resource Center}).$$

其中,

- $MC = LS(\text{login server}) \cup CS(\text{control server}) \cup mDB(\text{meta database});$
- $CRC = CN(\text{computer node}) \cup CCC(\text{computer cluster controller}) \cup VMMs(\text{virtual machine monitors}) \cup VMs(\text{virtual machines});$
- $SRC = SN(\text{storage node}) \cup SCC(\text{storage cluster controller}) \cup VMimas(\text{virtual machine images}) \cup Vmmous(\text{virtual machine moulds}).$

定义 4. Op 是用户服务功能集,包括了系统主体使用云平台的一些基本操作,记为

$$Op=\{Login,Logout,VMapply,VMuse,VMrelease,Ima-Mou-manage\}.$$

- $Login,Logout$:用户登入与登出云的操作,一般通过主体与接入服务器交互完成;
- $VMapply:Users$ 申请资源操作,一般表现为租户申请创建一个新的 VM 实例;
- $VMuse:Users$ 使用资源操作,使用过程包括虚拟机实例的连接使用、启动、暂停、恢复和关闭.由于虚拟机迁移是云的内部操作,不由租户控制和操作,因此在本文中不作考虑;
- $VMrelease:Users$ 释放资源操作,一般表现为 $Users$ 释放销毁一个 VM;
- $Ima-Mou-manage$:用户对于虚拟机镜像或者虚拟机模板的管理操作,包括上传、注册、激活发布与删除.其中, $Users$ 有权对镜像进行管理,而 $Admins$ 负责对于虚拟机模板进行管理.

集合 Op 包括了从虚拟机模板创建一直到资源释放的全部动作,是一个用户使用资源的完整生命周期,因此具有较好的概括性与完备性.

3 云平台的形式化建模

3.1 云平台的可信性定义

根据云平台的特点,结合 IEEE CS 可信计算与容错技术委员会的观点、文献[5]的内容以及我们自己对于可信的理解,我们认为:可信云平台除了提供安全的云计算服务、明确向用户保证隐私和数据的安全外,还必须详细规定平台在可用性、可靠性等方面的属性.因此我们认为,可信性应该包括 3 个子属性,分别是安全性、可用性以及可靠性.

我们进一步参考云计算安全联盟(cloud security alliance,简称 CSA)发布的《云安全指南》^[25]、全国信息安全标准化技术委员会(TC260)制定的安全标准^[26-28]以及云计算发展与政策论坛(3CPP)制定的《可信云服务认证评估方法》和《可信云服务认证评估操作办法》^[29,30],从易于理论分析和可信性评估的角度出发,对安全性、可用性以及可靠性的具体内容给出了详细定义与描述(定义 5),并以此作为本文提出模型的分析依据.

定义 5. 可信性= $\{安全性,可用性,可靠性\}$,其中,安全性主要包括密码机制、网络安全、租户数据安全、安全隔离、认证与访问控制策略以及策略执行等,可用性主要包括服务可用与组件负载等,可靠性主要包括健壮性、可控性与故障等级等.

对于定义 5 中各项属性的含义与解释见表 1.

Table 1 Credibility definition and description

表 1 可信性定义及其属性描述

可信属性	子属性	子属性内容
安全性	密码机制	密码算法实现、密码使用模式、密钥长度、密钥管理机制、签名机制
	网络安全	网络传输安全、网络入侵检测、防火墙、内容过滤
	租户数据安全	数据恢复策略、数据重建、数据存储加密、数据传输加密、数据使用加密、数据删除、数据泄露
	安全隔离	存储隔离、网络隔离、租户隔离、资源隔离
	认证与访问控制	身份管理、认证服务、授权服务、控制策略
	策略执行	存放策略组件安全、执行组件约束能力
可用性	服务可用	基础服务功能、特殊权限功能
	组件负载	组件负载情况
可靠性	健壮性	死锁检测、性能瓶颈、容错能力
	可控性	系统状态切换、控制组件的管控能力
	故障等级	故障严重程度划分

根据我们的理解给出的可信性中有众多属性,仅仅用一个模型就想分析全部内容是困难且不现实的,因此,首先要明确本文建立模型主要针对上述哪些属性进行分析,以便确定分析范围.

在云平台中,对于安全性的大部分属性分析已经较为完善:密码机制是属于密码学的范畴,而网络安全、租

户数据安全、安全隔离、认证与访问控制策略均可通过安全协议和访问安全策略并结合管理手段来保障.如何设计这些协议与策略不是本文的研究范围,且对于现有安全协议与安全策略的验证,利用 π 演算、BAN 逻辑等形式化工具也有了较为成熟的建模验证方法^[31,32],并能够通过这些方法发现一些安全问题,因此,这些属性的分析不是本文研究的重点.

本文着重解决针对云平台的可用性、可靠性以及部分安全属性的建模与分析困难问题,具体包括:策略执行、服务可用、组件负载、健壮性、可控性、故障等级等,这些属性均是云平台在提供服务过程中不同组件的动态执行表现,是云平台可信性的重要组成部分,目前尚缺乏能够对它们进行统一分析的模型与方法.通过本文提出的方法分析,可以解决目前只能分析云平台部分可信属性的问题,能够为进一步进行可信云评估并构建可信云环境提供重要理论补充.

3.2 建模目的与思路

根据第 3.1 节中的定义与描述,我们提出本文建模的目的:通过建立模型对云平台可信性中的部分属性进行分析,发现潜在的可信性问题,提出一些解决建议,为后续的云平台可信评估以及进一步构建可信云提供了理论支撑.建模的内容和思路包括:

首先,服务的可用性是用户的基本需求之一,如果云平台提供的某些服务由于设计缺陷不可用,那么一切安全也就无从谈起.因此,实现服务可信的基本前提之一就是保证云环境提供的所有服务都是可用的.在对云平台的具体分析中,我们首先需要明确所有用户都能使用的基础功能以及不同权限用户能够使用特殊功能;在此基础上,对用户与云环境之间的交互过程进行建模,通过建立模型来分析功能的可用性以及不同组件的负载情况.

其次,在安全协议和策略的保障下,用户隐私数据在一定程度上得到了保障,但是用户不仅希望使用的服务是安全的,还希望在任何时刻都能不间断的按需使用服务,这就依赖于系统的可靠性.云平台是一个构造复杂的系统,内部包含海量的控制、计算与存储实体以及大量的系统与软件,用户的任意服务请求都会引发云内部组件的交互,因此,我们还需要对参与用户服务的组件进行建模,并进一步将其整合为系统整体状态变迁模型,进而分析云平台在服务提供中可能存在的性能瓶颈与死锁隐患,并给出容错性建议.此外,我们还需要分析系统组件切换与状态转换过程中的潜在威胁,并根据故障组件的重要程度对可能出现的故障提供等级划分基础.

最后,服务的安全性还和安全策略能否严格正确执行有关.在配置了完善的安全策略之后,这些策略不能被正确的执行,或是相关配置文件本身容易被攻击者篡改,都会对导致信息泄露而引发服务故障.在云平台中,安全策略配置存放在相关组件中并由它们负责执行,因此,我们还需要对这些存放、执行安全策略组件的约束能力进行分析.

3.3 建模目标与约束

基于第 3.1 节和第 3.2 节中的内容,我们提出模型建立的目标:建立用户与平台之间以及平台内部组件的动态交互模型,并从可用、可靠以及安全约束能力这 3 个方面分析其对外提供服务时可能存在的可信性问题.针对以上目标,我们还为本模型设定以下约束.

- 模型是一个抽象的交互模型,刻画了用户服务引发的云内部组件交互过程,但是不对具体的实现细节进行建模;
- 模型必须具有整体性,即,能够反映出云平台逻辑组件之间的关系,不能仅仅只是对云平台的部分组件进行多次独立的描述;
- 模型必须是动态的,能够反映出系统的动态变化,以体现服务的动态性;
- 模型不包括对云平台存在的安全协议和策略本身进行建模,而主要体现执行这些策略和协议的组件的安全约束能力.

3.4 云平台运行状态变迁模型

标记变迁系统是能够用于形式化描述并发系统的可观察行为,成为系统说明与深入研究的可靠工具^[33-35].此外,LTS 还能够作为许多进形式化描述语言的语义模型,比如 CCS,CSP 等.因此,我们采用 LTS 相关理论刻画

建立模型.

3.4.1 LTS 相关理论定义

一个标记变迁系统由状态以及带有标记的变迁动作组成,对于任意一个给定的模型,给出原子动作和状态变迁规则,就可以描述出对应的变迁系统.

LTS 的相关语法如下所示:

符号	定义
L	系统所有可观察的动作集
L^*	L 上的字符串集
τ	系统不可观察的内部动作
l_1, l_2, l_3	集合 L 的元素
ε	空序列
L_τ	$L \cup \{\tau\}$
μ, μ_1, μ_2, \dots	集合 L_τ 的元素
s, s', s_1, s_2, \dots	集合 S 的元素
$s \rightarrow s'$	$s \xrightarrow{\tau} s'$
$s \xrightarrow{\mu_1 \mu_2 \dots \mu_n} s'$	存在 $n \in \mathbb{N}$, 满足 $s \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} s_2 \xrightarrow{\dots} s_n \xrightarrow{\mu_n} s'$
$s \Rightarrow s'$	存在 $s \xrightarrow{\tau^n} s', n \geq 0$
$s \xRightarrow{\mu} s'$	存在 s_1, s_2 , 满足 $s \xRightarrow{\mu} s_1 \xrightarrow{\mu} s_2 \xRightarrow{\mu} s'$
$s = \mu \Rightarrow$	存在 s' 使得 $s = \mu \Rightarrow s'$
$s = \mu \nRightarrow$	$s = \mu \Rightarrow$ 的否定

下面给出标记变迁系统的操作语义定义.

定义 6. 标记变迁系统是四元组 (S, L, \rightarrow, s_0) : S 是非空有限状态集合; L 是有限标号(动作)集合, 它表示了系统所有的可观察动作, 而外部不可观察的内部动作记为特殊标记 τ ; \rightarrow 是 S 上的二元关系, 其中的元素可记为 $s \xrightarrow{\mu} s', s, s' \in S, \mu \in L \cup \{\tau\}$, 表示系统在执行动作 μ 后, 由状态 s 变迁到 s' ; s_0 是系统的初始状态, 且 $s_0 \in S$.

标记变迁系统中, 还存在一些重要性质, 例如迹的相关概念与性质、状态收敛的判定、LTS 树的判定等, 下面分别给出定义.

定义 7. 迹(traces)是 LTS 中的重要概念, 它表示系统可观察动作所组成的一个有限序列. L 上所有迹的集合用 L^* 表示. 迹相关的重要性质有:

- (1) $Trace(s) = \{ \alpha \in L^* | s \xRightarrow{\alpha} \}$;
- (2) $con(\alpha_1, \alpha_2) = \alpha_1 \cdot \alpha_2, \alpha_1, \alpha_2 \in L^*$;
- (3) $len(\alpha) = |\alpha|, \alpha \in L^*$;
- (4) $after(s, \alpha) = \{ s' \in S | s \xRightarrow{\alpha} s' \}$;
- (5) $sub(s, s') = \{ s' \in S | \exists \alpha \in L^* : s \xRightarrow{\alpha} s' \}$.

定义 7 中: 性质(1)给出了迹的定义; 性质(2)定义了迹的连接表示符号; 性质(3)定义了迹的长度表示方法; 性质(4)描述了状态改变过程, 即状态 s 经过动作 α 后到达状态 s' ; 性质(5)描述了动作迹存在性的表示方法, 即, L 中存在一条由状态 s 到状态 s' 的动作迹 α .

定义 8. $s \Downarrow$ 表示状态 s 收敛, $s, s' \in S$, 则有:

- (1) if $s \Downarrow, s \Downarrow \varepsilon$;
- (2) if $s \Downarrow$ and $s \xRightarrow{\mu} s', s \Downarrow \mu$;
- (3) if $s \xrightarrow{\mu} s'$, then $s \Downarrow, s' \Downarrow$.

定义 8(1)说明: 如果状态 s 在不可见动作集上收敛, 可得出其在空序列上也是收敛的; 定义 8(2)说明: 如果状态 s 在空序列的上收敛, 且能够经过一个动作迹到达另一确定状态, 则 s 收敛于动作 μ ; 定义 8(3)常用于检测标记变迁系统中是否存在不可观察的无限动作序列.

定义 9. if $s' \in LTS(s'), s \in LTS(s); s' \text{ tr } s \stackrel{\text{def}}{=} trace(s') \subseteq trace(s)$.

定义 9 描述了子树判定准则,即:如果存在 $LTS(s')$ 是 $LTS(s)$ 的子树,则两者应满足二元关系 \sqsubseteq .

3.4.2 云平台组件建模

我们首先对第 2.2 节中定义的主要客体分别建模,每一个组件的 LTS 状态图都可以看做是构成云平台整体状态图的一个组成部分.

根据云平台结构的特点,我们将集合 Ob 中 MC , CRC 和 SRC 三者之间的通过程、 MC 内部的客体交互过程以及 CCC 与 SCC 的控制管理过程定义为系统可观察动作,而 CRC 与 CN 、 SRC 与 SN 的交互过程以及各个逻辑组件内部处理过程为系统不可观察的内部动作.

根据第 3.4.1 节中描述的 LTS 语法与语义,我们对重要的组件建模过程如下.

(1) LS 建模

登录服务器负责对云环境主体的登录请求进行验证,验证当前是 $Admins$ 还是 $Users$ 登录,并分别赋予不同权限,若口令错误,则返回等待状态.其过程的 LTS 语义表述为

$wait - error \rightarrow wait;$
 $wait - Users \rightarrow User_Login;$
 $wait - Admin \rightarrow Admin_Login.$

LS 的 LTS 模型 $LTS(S_{LS})$ 如图 2 所示.

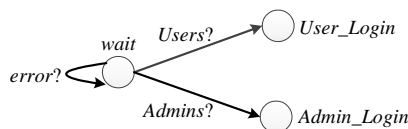


Fig.2 LTS model of LS

图 2 登录服务器的 LTS 模型

(2) CS 建模

调度服务器负责接收从登录服务器传来的用户指令,并进行查询元数据库处理.如果查询正确,则转发给相应组件进行下一步处理;否则返回初始状.同时,调度服务器还负责接受计算集群和存储集群发来的操作信息,并进行相应的元数据库更新操作.其 LTS 语义表述为

$wait - receive \rightarrow judge\ and\ process;$
 $judge\ and\ process - sent\ from\ LS \rightarrow wait\ result;$
 $judge\ and\ process - sent\ from\ CCC \rightarrow process;$
 $judge\ and\ process - sent\ from\ SCC \rightarrow process;$
 $process - update \rightarrow wait;$
 $wait\ result - receive \rightarrow judge\ and\ compute;$
 $judge\ and\ compute - error \rightarrow wait;$
 $judge\ and\ compute - correct \rightarrow next\ state.$

CS 的 LTS 模型 $LTS(S_{CS})$ 如图 3 所示.

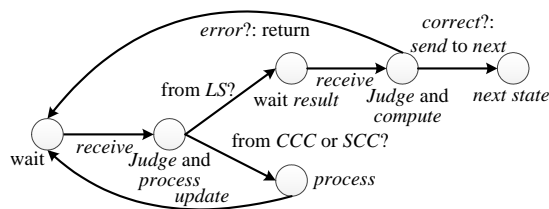


Fig.3 LTS model of CS

图 3 调度服务器的 LTS 模型

(3) *mDB* 建模

元数据库仅仅接收调度服务器发送的指令,并进行相应处理后返回结果,处理过程是一个不可见的内部动作.其 LTS 语义表述为

$$\begin{aligned} &wait - receive \rightarrow process; \\ &process - sent \rightarrow wait. \end{aligned}$$

mDB 的 LTS 模型 $LTS(S_{mDB})$ 如图 4 所示.

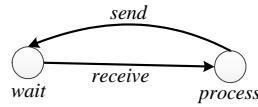
Fig.4 LTS model of *mDB*

图 4 元数据库的 LTS 模型

(4) *CCC* 建模

计算集群控制节点是 *CRC* 内部对外交互通信的唯一组件,它负责接收 *CS* 发来的指令,并在集群内部做相应处理后,返回结果.在某些情况下,*CCC* 会返回再次执行指令,例如在释放资源时发现 *VM* 未关闭,则需要返回先关闭 *VM*.其 LTS 语义表述为

$$\begin{aligned} &wait - receive \rightarrow process; \\ &process - send \rightarrow wait\ result; \\ &wait\ result - receive \rightarrow judge\ and\ compute; \\ &judge\ and\ compute - release\ VM \rightarrow process; \\ &judge\ and\ compute - send \rightarrow wait. \end{aligned}$$

CCC 的 LTS 模型 $LTS(S_{CCC})$ 如图 5 所示.

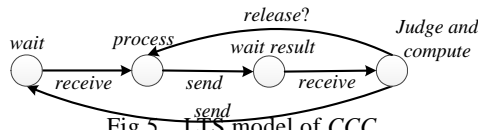
Fig.5 LTS model of *CCC*

图 5 计算集群控制节点的 LTS 模型

(5) *SCC* 建模

存储集群控制节点是 *SRC* 内部对外交互通信的唯一组件,它与 *CCC* 具有类似功能.不同的是,在某些指令控制下,它还能向 *CCC* 发送信息.其 LTS 语义表述为

$$\begin{aligned} &wait - receive \rightarrow judge\ and\ process; \\ &judge\ and\ process - send\ to\ CS \rightarrow wait; \\ &judge\ and\ process - send\ to\ CCC \rightarrow next\ state. \end{aligned}$$

SCC 的 LTS 模型 $LTS(S_{SCC})$ 如图 6 所示.

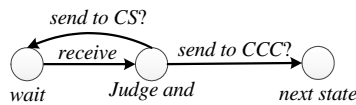
Fig.6 LTS model of *SCC*

图 6 存储集群控制节点的 LTS 模型

$$\text{Min} \left\{ \sum_{\substack{k=m+1=n+2 \\ k,m,n \in B}} b_k + b_m + b_n \right\} = \text{Min} |LTS(s)|.$$

证明:根据图 2~图 6,有 $A=\{1,3,2,4,2\}$,因此:

$$\text{Min} \left\{ \sum_{\substack{k=m+1=n+2 \\ k,m,n \in B}} b_k + b_m + b_n \right\} = \{b_1 + b_2 + b_3\} = 6.$$

而 $\text{Min}|LTS(s)|=|\alpha_1 \alpha_3 \alpha_{14} \alpha_{15} \alpha_{16} \alpha_{20}|=6$,因此定理 2 得证.

定理 2 说明从初始状态到结束状态,最少情况下组件的参与运行情况,即:最少需要 LS,CS 以及 mDB 的协同,这也是与云平台的整体变迁过程和实际的运行过程是相符合的.定理 2 也说明:从组件参与过程的角度,建立的模型具备正确性.

图中, s_{14}, s_{15}, s_{16} 以及 s_{34}, s_{35}, s_{36} 这 6 个状态是系统的关键状态, s_{14} 和 s_{34} 是调度服务器接收到用户发送来的操作指令, s_{15} 和 s_{35} 是调度服务器根据需要查询元数据, s_{16} 和 s_{36} 是元数据库返回有效查询结果给调度服务器,这 3 个状态也是用户发送大多数操作指令必须经过的状态.状态 s_0 是系统的起点,是用户登陆之前的状态,但 s_{19}, s_{29} 和 s_{41} 不是整个系统终结状态,而是指一条用户操作请求的完成.

图 2 描述了在外部用户输入不同指令时,内部组件交互引起的不同系统状态共计 41 种, $LTS(s)$ 中的可观察动作集合 $L=\{a_i|0 \leq i \leq 48\}$, $S \times S$ 上的二元关系共 70 种.如此复杂的可观察动作和状态不利于进一步对系统安全进行分析,因此有必要对图 2 进行简化.

3.4.4 状态图约减

对于云平台的主体与客体而言, LS 仅验证用户身份后转发一次用户操作请求,而不对操作请求做任何处理; CS 在处理 LS 发送数据时,命令权限已经由接入服务器验证, CS 仅执行操作;对于 SCC 和 CCC 而言,都是直接执行由 CS 发送的指令,并与自身集群内部的 SN 和 SN 进行交互;并且,不论是以何种用户身份登录,一个操作指令的终止状态要么是更新元数据库,要么是建立了一个远程连接以供用户使用.根据以上特点,状态图约减规则可以总结为:

- M1:状态 $s_3 \sim s_{13}$ 以及状态 $s_{30} \sim s_{33}$ 是可以去除的;
- M2:状态 s_{14} 与 s_{34} , s_{15} 与 s_{35} , s_{16} 与 s_{36} 这 3 组状态是可以合并的;
- M3:M2 中合并的 3 组状态必须被保留;
- M4: $LTS(s)$ 的终止状态 s_{29} 和 s_{41} 是可以合并的;
- M5: s_0 是 $LTS(s)$ 的起始状态, $LTS(s)$ 的结束状态是 s_{19} 以及 P4 中合并的状态.

按照上述简化规则约减后的状态图如图 8 所示.

如图所示,经过简化后 $LTS(s')$ 中,状态约减为 11 种,可观察动作集合 $L'=\{a_i|0 \leq i \leq 17\}$, $S \times S$ 上的二元关系仅剩余 18 种.为了说明 $LTS(s)$ 到 $LTS(s')$ 约减过程的正确性,我们还需要进行必要的数学证明.

定理 3. 如果 $LTS(s')$ 是 $LTS(s)$ 经过正确的动作约减而成,则满足 $s' \xrightarrow{tr} s, s' \in LTS(s'), s' \in LTS(s)$.

证明:对于 $s' \in LTS(s'), s' \in LTS(s)$, 根据定义 9, 要证明 $s' \xrightarrow{tr} s$, 只需证明 $trace(s') \subseteq trace(s)$.

又根据定义 7(1), $Trace(s)=\{\alpha \in L^* | s=\alpha \Rightarrow\}$, 因此,定理 3 的证明转为证明 $\{\alpha_{s'}\} \subseteq \{\alpha_s\}$.

$\{\alpha_{s'}\} \in L_{s'}^*, \{\alpha_s\} \in L_s^*$, 而根据定理条件,有 $L_{s'}^* \in L_s^*$, 因此有 $\{\alpha_{s'}\} \subseteq \{\alpha_s\}$. 证毕.

定理 3 说明: $LTS(s')$ 是 $LTS(s)$ 经过正确的动作约减而成,且经过约减的 $LTS(s')$ 中的动作序列是原始状态图 $LTS(s)$ 中动作序列的子集,说明了动作序列约减过程的正确性.

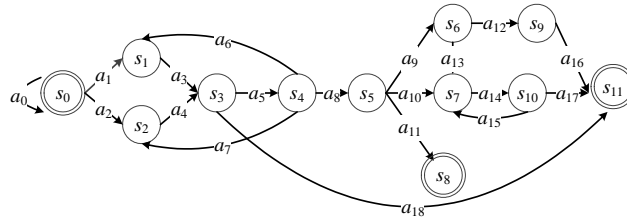


Fig.8 LTS model after reduce

图 8 约减后的模型图 $LTS(s')$

4 模型分析与验证

4.1 验证内容分析

结合第 3.1 节中对可信属性的分解描述,我们提出以下针对建立的模型的验证分析内容,并根据分析结果给出一定的建议.

1) 可用性.该属性的子属性是服务可用性与组件负载,其中:服务可用性描述的是各项服务是否存在并且能被正确执行的属性,主要关注系统状态转变,表示在控制策略范围内,系统主体期望的服务是否可用,在对应状态图中可以表述为 $LTS(s')$ 中的任何一个状态是否都由初始状态 s_0 可达;组件负载主要体现服务的易用性,可以根据 $LTS(s)$ 以及 $LTS(s')$ 中的路径经过的节点情况,给出不同组件的负载的量化数据.

2) 可靠性.该属性又包括健壮性、可控性与故障等级划分.下面分别进行说明.

- (1) 健壮性,我们不仅希望系统主体的期望能够正确执行,也希望云环境不会因为误操作或攻击而陷入错误的状态,因此,我们针对系统死锁和性能瓶颈进行分析,并在此基础上给出一些容错性建议.例如,大多数服务都会经过查询元数据这一步骤,因此,我们希望系统能够识别有效的服务请求与恶意的攻击指令,保证系统不会因为执行攻击者精心构建的大量无效元数据库查询请求而陷入死锁状态,尽量减少拒绝服务类攻击的可能;
- (2) 可控性,表示云环境能够执行正确的选择,即:系统处于某一状态时,能够受控制的进入另一正确状态.我们将对系统状态切换、控制组件的管控能力进行分析,例如,当元数据库查询结果错误时,能够返回 User 等待状态;
- (3) 故障等级,结合可用性中的组件负载情况,可以分析故障等级的严重程度,负载较高组件故障时,往往意味着严重故障出现.

3) 安全约束能力.该属性描述的是执行访问控制策略的组件的参与能力的属性.在云平台中,一般由 LS 来负责赋予用户角色权限,由 CS 来执行主要访问策略,是云中的关键组件.因此,需要验证系统中所有规定的从初始状态到结束状态的路径都经过关键节点,即:系统中不存在越权操作、有绕过关键组件控制的路径存在.

4.2 模型验证

我们使用模型检测工具 Kronos 来进行检验分析,Kronos 是一种用于实时系统的分析验证工具,它可以使用计数的或者带有标记的程序搜索带自动机,支持各种过程代数标记的模型验证^[36].标记变迁系统由一组带有标记的变迁动作构成,且在本文的模型中,状态转换从时序上可以看做是无时间约束的顺序转换,因此,也可以使用 Kronos 来对本文构建的状态图进行验证分析^[37].

我们采用的实验环境是 Intel i5-2450,4G 内存,系统采用 Ubuntu 12.10,Kronos 版本号为 2.52.

4.2.1 参数设定

对模型进行分析前,我们首先要对状态节点进行分析并进行参数设定.在 $LTS(s')$ 中所经历涉及的状态节点均包含以下 3 个状态:(1) 等待触发事件($Idle_i$);(2) 获取并同步传输消息(T_i, ST_i);(3) 获取并异步传输消息(T_i, AT_i).其中,状态转换由消息队列的阻塞程度决定.因此,我们对每一个状态节点均使用 3 个参数记录事件触发

的次数, T_i 记录当前节点交互的任务, ST_i 记录同步消息传输次数, AT_i 记录异步消息传输触发次数. 在本文的模型中, 我们可以假定因消息阻塞而导致的异步传输延时 $TTL \approx 0$.

同时, 我们使用一个栈来存储一个状态节点所处理的消息. 具体过程是: 当有消息到达节点 s_i 时, 先进入消息处理缓冲队列, 根据当前节点空闲状态判断异步传输或同步传输; 消息处理完毕后, 将节点 s_i 压入当前任务的栈 $Trace$ 中. 节点 s_i 的行为被描述为一个自动机 $Station_i$.

对于整个 $LTS(s')$, 我们将其描述成一个积自动机 $LOGIN = protocol || Station_1 || \dots || Station_i$. 为验证方便, 实验中将具有类似属性的节点描述为同类节点, Kronos 中节点和交互过程分别用文件 $station.tg$ 和 $protocol.tg$ 表示. 由于节点状态以及交互过程的复杂性, 导致验证的输入冗长, 本文中仅给出部分描述以作说明.

以节点 s_1 为例, 在 $station.tg$ 中部分描述如下:

```
#states 3
#trans 6
#clocks 0
#sync IN OUT

state: 0
invar: TRUE
trans:
TRUE  $\Rightarrow$  IN; reset{}; goto 1
TRUE  $\Rightarrow$  OUT; reset{}; goto 2
```

交互过程在 $protocol.tg$ 中的部分描述如下:

```
state: 1
prop: R_0_S
invar: TRUE
trans:
TRUE  $\Rightarrow$  HEAD_0; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_1; reset{}; goto 2
TRUE  $\Rightarrow$  ADD_0; reset{}; goto 0
TRUE  $\Rightarrow$  ADD_1; reset{}; goto 2
TRUE  $\Rightarrow$  ADD_01; reset{}; goto 2
TRUE  $\Rightarrow$  OUT; reset{}; goto 2
TRUE  $\Rightarrow$  ADD_0 OUT; reset{}; goto 0
TRUE  $\Rightarrow$  HEAD_0 ADD_0; reset{}; goto 1
TRUE  $\Rightarrow$  HEAD_0 ADD_1; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_0 ADD_01; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_0 OUT; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_0 ADD_0 OUT; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_1 ADD_0; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_1 ADD_1; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_1 ADD_01; reset{}; goto 0
TRUE  $\Rightarrow$  HEAD_1 OUT; reset{}; goto 2
TRUE  $\Rightarrow$  HEAD_1 ADD_0 OUT; reset{}; goto 2
```

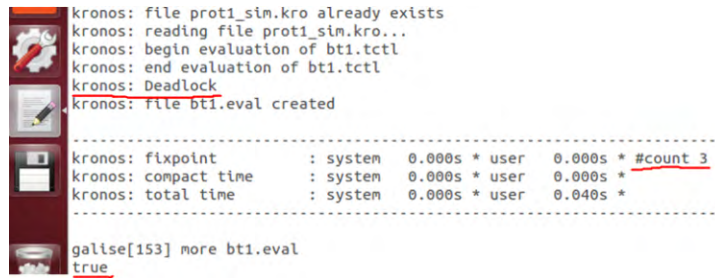
4.2.2 模型分析

篇幅所限, 在此仅给出 2 个验证实例进行详细说明.

首先, 我们不会因为无效的数据库查询请求而发生拒绝服务攻击. 如果图中不存在闭环, 就能很大程度上减少上述攻击发生的可能. 因此, 上述攻击性质在图中表示为不存在死锁.

结合第 4.1 节的设定, 我们定义经过状态节点 s_i 次数 $Count_i = ST_i + AT_i$, 这一验证结论如果存在死锁, 则在 Kronos 中应该表达为 $Count_i \leq 3$, 该性质描述存于文件 $bt1.tctl$ 中.

通过 Kronos 分析得到的结果部分截图如图 9 所示.



```

kronos: file prot1_sim.kro already exists
kronos: reading file prot1_sim.kro...
kronos: begin evaluation of bt1.tctl
kronos: end evaluation of bt1.tctl
kronos: Deadlock
kronos: file bt1.eval created

-----
kronos: fixpoint      : system  0.000s * user  0.000s * #count 3
kronos: compact time  : system  0.000s * user  0.000s *
kronos: total time    : system  0.000s * user  0.040s *
-----

galise[153] more bt1.eval
true

```

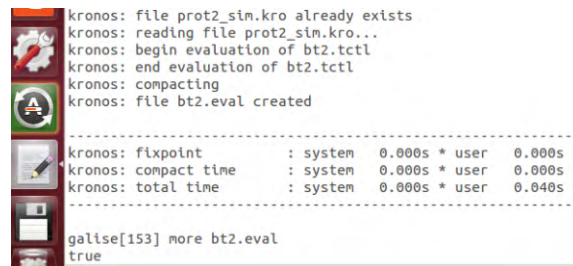
Fig.9 Analysis result 1

图 9 分析结果图 1

运行结果表明:图中存在发生死锁的动作迹 $\alpha_3\alpha_5\alpha_6$ 和 $\alpha_4\alpha_5\alpha_7$,表明攻击者可持续构造错误的查询指令,消耗CS与mDB的资源.验证结果说明了CS,mDB这两个系统组件由于有安全隐患而更易受到拒绝服务类攻击.在实际的部署中,我们就可以在这些组件中设置控制策略.比如:对可能引起类似攻击的服务,增加使用间隔阈值、发现大量类似服务请求时随机丢弃等方式来应对.

其次,我们希望验证在正确的访问控制策略下,用户使用VM的诉求能否达成,即,验证状态 s_8 是否可达.这条性质属于服务可用性的验证.

根据第4.2.1节中的描述,我们假设每个站点获得的消息不为空,则该性质在Kronos中表达为栈Trace中必存在记录 $s_1 \rightarrow s_8$.通过Kronos分析得到的结果部分截图如图10所示.



```

kronos: file prot2_sim.kro already exists
kronos: reading file prot2_sim.kro...
kronos: begin evaluation of bt2.tctl
kronos: end evaluation of bt2.tctl
kronos: compacting
kronos: file bt2.eval created

-----
kronos: fixpoint      : system  0.000s * user  0.000s *
kronos: compact time  : system  0.000s * user  0.000s *
kronos: total time    : system  0.000s * user  0.040s *
-----

galise[153] more bt2.eval
true

```

Fig.10 Analysis result 2

图 10 分析结果图 2

运行轨迹说明:云平台的各个组件在正常的访问控制策略下能够满足用户提出的服务请求,攻击者不能绕过权限来越级使用某些不属于自身权限范围内的功能.

4.3 验证结论

基于状态图 $LTS(s')$ 与模型检测工具Kronos,我们对第4.1节中提出的分析内容分别进行了验证,得到验证结果如下:

(1) 满足可用性.我们利用Kronos对用户服务功能集OP中的功能进行了逐条验证,每一个功能所代表的状态节点都是可达的,说明云平台提供给用户的相关功能都是可用的.但是在可用性分析中我们也发现:用户不但可以使用自己拥有使用权限的功能,还可能绕过访问控制权限越级使用某些非法功能,例如,攻击者利用漏洞实现使用User身份登陆后访问其他User的数据域信息等.这就需要在安全约束分析中进一步展开分析.

(2) 不满足可靠性.在健壮性方面, $LTS(s')$ 中存在4个环,与之相关的组件有LS,CS,mDB以及CCC.验证结果说明:潜在攻击者可以构造合理命令,使得系统陷入死循环,从而消耗系统资源.此外,运行轨迹显示:从初始状态到终止状态的所有运行轨迹中,有较多的运动轨迹经过LS以及mDB,说明调度服务器的处理能力与元数据查询能力是系统瓶颈资源,一旦这两个系统组件发生故障,容易造成系统的崩溃.我们还可以根据运行轨迹经过的组

件情况来判定组件在服务提供中的重要程度,为故障等级划分提供依据.在可控性方面,系统主要控制客体能够按照合理的访问控制策略满足云平台使用主体的诉求,即,具有控制功能的组件,包括接入服务器、调度服务器以及计算、存储集群控制节点能够按照预设的策略来控制系统的状态切换,说明具有较好的可控性.

相关组件具有安全约束能力.在状态图 $LTS(s')$ 中, s_3, s_4, s_5 是接入服务器和调度服务器参与工作的状态转换节点,图中所有动作迹都经过了这 3 个状态节点,说明了接入服务器和调度服务器具有关键的约束功能,配置的访问控制策略都能够发挥作用.但是这不保证接入服务器和调度服务器本身的安全,攻击者仍然能对关键组件本身发起攻击,篡改一些关键安全配置.因此在实际部署中,接入服务器和调度服务器是安全防护的重点关注对象,需要使用密码和安全协议等技术对其进行保护.

此外,我们还将本文的贡献和同类研究进行了对比,对比对象是文献[6,7,38],这 3 个研究都是用不同方法对云环境的部分或是整体进行建模,并对其中某些属性进行分析,对比结果见表 2.对比结果显示:本文提出的方法相对比较全面,能够对平台整体的各个不同属性进行综合分析,具有一定的优越性.

Table 2 Comparison of the results

表 2 研究结果对比

	本文研究	文献[6]	文献[7]	文献[38]
分析对象	云平台整体	虚拟机层	虚拟机监控以及虚拟机	云平台整体
云可信性定义	√	×	×	×
云环境建模	√	√	√	√
可用性分析	√	×	×	×
可靠性分析	√	×	√	√
约束分析	√	√	×	×
是否发现隐患	√	×	√	×

通过 Kronos 对 $LTS(s')$ 进行验证,我们可以发现:现有的大多数云平台存在一定的可信性问题,一般来说,服务满足了服务可用性,相关组件也具有较好的控制和安全约束能力;但是不具备较好的容错能力,且易受到资源消耗类的攻击.此外,对于分析得出的重要组件的保护也是保障服务可信的重点.在实际部署中,我们就可以针对以上安全隐患做出应对,例如对安全策略配置文件进行加密,增加关键组件的热备份服务器,对于某些服务设置使用间隔阈值,随机丢弃可能无效的数据包等.但是这些解决方案不是本文要讨论的重点,在此就不一一列举.我们将在下一步研究中对于这些存在的可信性问题展开更深入的研究.

5 结束语

由于商业云故障频发,企业与普通用户仍然对其应用场景的服务是否安全可靠存在很大疑虑,这也不利于云计算的进一步推广与发展.因此,对云平台的可信性进行分析验证是非常有必要的.

针对目前的模型不适用于对服务可信性中的部分动态属性进行有效分析的问题,本文利用 LTS 理论,提出一种新的模型建立方法:在抽象出云平台的逻辑组件基础上,从用户行为引发的组件交互过程出发,对关键组件进行建模,并对这些相对独立的单个模型进行整合,从动态的角度刻画出云平台在服务提供时,云环境的整体运行状态变迁图;同时,针对状态空间过于复杂问题进行了简化,解决了云平台可信性建模与分析困难的问题.以 Kronos 为工具对状态图进行检测时,除了发现云平台对外提供服务时一些已知的可靠性隐患外,还发现了存在一些未知的可靠性问题,如容错能力的具体量化数据等.

我们下一步工作将结合本文发现的问题,提出在实际部署时的解决措施,解决诸如绕过安全组件的旁路攻击、针对 LS 和 mDB 的资源消耗类攻击等;此外,还将基于路径对云平台关键组件的负载能力进行量化,并基于本文的分析结果构建故障等级评价体系,为可信云测评提供基础.

References:

- [1] Allen E, Peng LY, Lin RH, Jenney G. China's Pragmatic Path to Cloud Computing. 2011.

- [2] Common Criteria Project Sponsoring Organisations. Common criteria for information technology security evaluation. ISO/IEC Int'l Standard (IS), 15408 1-3, Version 2.1.
- [3] Trusted Computing Group (TCG). TCPA Main Specification. Version 1.1b.
- [4] IEEE Computer Society Technical Committee on Dependable Computing and Fault Tolerance. <http://www.dependability.org/>
- [5] Shen CX, Zhang HG, Wang HM, Wang J, Zhao B, Yan F, Yu FJ, Zhang LQ, Xu MD. Reserches on trusted computing and its developments. Science China: Information Sciences, 2010,50(3):405–433. [doi: 10.1007/s11432-010-0069-x]
- [6] Benzina H. Towards Designing Secure Virtualized Systems. In: Proc. of the Digital Information and Communication Technology and it's Applications (DICTAP). 2012. [doi: 10.1109/DICTAP.2012.6215385]
- [7] Zegzhda P, Zegzhda D, Nikolskiy A. Using graph theory for cloud system security modeling. Computer Network Security, 2012: 309–318. [doi: 10.1007/978-3-642-33704-8_26]
- [8] Khan I, Rehman H, Anwar Z. Design and deployment of a trusted eucalyptus cloud. In: Proc. of the 2011 IEEE Int'l Conf. on Cloud Computing (CLOUD). Washington, 2011. 380–387. [doi: 10.1109/CLOUD.2011.105]
- [9] Cui W, Li YF, Si XM. The protocol design of a eucalyptus-based infrastructure-as-a-service (IaaS) cloud framework. Journal of Electronics & Information Technology, 2012,34(7):1748–1754 (in Chinese with English abstract).
- [10] Weinman J. Axiomatic Cloud Theory. Working Paper. 2011.
- [11] Si GN, Ren YH, Xu J, Yang JF. A dependability evaluation model for internetware based on bayesian network. Journal of Computer Research and Development, 2012,49(5):1028–1038 (in Chinese with English abstract).
- [12] Liu CY, Lin J, Tang B. Dynamic trustworthiness verification mechanism for trusted cloud execution environment. Ruan Jian Xue Bao/Journal of Software, 2014,25(3):662–674 (in Chinese with English abstract). <http://www.jos.org.cn/1000-9825/4447.htm> [doi: 10.13328/j.cnki.jos.004447]
- [13] Zhu J, Guo CG, Wu QY. A Web services interaction behavior-environment model based on generalized stochastic Petri nets. Journal of Computer Research and Development, 2012,49(1):2450–2463 (in Chinese with English abstract).
- [14] Yi XC, Kochut KJ. A CP-nets-based design and verification framework for Web services composition. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS 2004). 2004. 756–760. [doi: 10.1109/ICWS.2004.1314810]
- [15] Hinz S, Schmidt K, Stahl C. Transforming BPEL to Petri nets. In: Proc. of the 3rd Int'l Conf. on Business Process Management (BPM 2005). Nancy, 2005. 220–235. [doi: 10.1007/11538394_15]
- [16] Fu X, Buhant T, Su J. Analysis of interacting BPEL Web services. In: Proc. of the 13th Int'l Conf. on the World Wide Web (WWW 2004). New York, 2004. 621–630. [doi: 10.1145/988672.988756]
- [17] Wombacher A, Fankhauser P, Mahleko P, Neuhold B. Matchmaking for business based on choreographies. Int'l Journal of Web Services Research, 2004,1(4):14–32.
- [18] Li LX, Jin Z, Li G. Modeling and verifying services of internet of things based on timed automata. Chinese Journal of Computers, 2011,34(8):1365–1377 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.01365]
- [19] Amazon elastic compute cloud (EC2). <http://aws.amazon.com/ec2/>
- [20] HP public cloud. <http://www.hpcloud.com/>
- [21] AT&T cloud architect. <http://cloudarchitect.att.com/Home/>
- [22] Nurmi D, Wolski R, Grzegorzczak C, Obertelli G, Soman S, Youseff L, Zagorodnov D. Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems. Technical Report, 2008-10, UCSB Computer Science, 2008.
- [23] The OpenStack community “OpenStack cloud software”. 2011. <http://www.openstack.org/>
- [24] The OpenNebula project “OpenNebula: The open source toolkit for cloud computing”. 2011. <http://opennebula.org/>
- [25] Security guidance for critical areas of focus in cloud computing. <http://www.cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf>
- [26] China Electronics Standardization Institute. Cloud Computing Standards Research. Beijing, 2010 (in Chinese with English abstract).
- [27] Wang HL, Yang C, Yang JJ. Information security and standard of cloud computing. Information Technology & Standardization, 2012. 16–19 (in Chinese with English abstract).
- [28] Wang HL, Luo FY, Yang JJ. Research on key criteria of cloud computing services security. Information Technology & Standardization, 2013,11:6–9 (in Chinese with English abstract).
- [29] Assessment method for trusted cloud service certification. <http://w.3cpp.org/lab/cloud/trustedcloudev/2014/0617/100.html>
- [30] Trial method of trusted cloud service certification. <http://www.3cpp.org/lab/cloud/trustedcloudev/2014/0617/100.html>

- [31] Milner R, Parrow J, Walker D. A calculus for mobile processes, part i. *Journal of Information and Communication*, 1992,100(1): 1–77.
- [32] Burrows M, Abadi M, Needham R. A logic of authentication. *ACM Trans. on Computer Systems*, 1990,8(1):18–36. [doi: 10.1145/77648.77649]
- [33] Keller RM. Formal verification of parallel programs. *Communications of the ACM*, 1976,19(7):371–384. [doi: 10.1145/360248.360251]
- [34] Plotkin G. A structural approach to operational semantics. Technical Report, DAIMI FN-19, Department of Computer Science Research Report DAIMI FN-19, Aarhus University, 1981.
- [35] Li MJ, Li ZJ, Chen HW. A survey of security protocol verification based on process algebra. *Journal of Computer Research and Development*, 2004,41(7):1093–1103 (in Chinese with English abstract).
- [36] Yovine S. Kronos: A verification tool for real-time system. *Int'l Journal on Software Tools for Technology Transfer*, 1997,1(12): 123–133. [doi: 10.1007/s100090050009]
- [37] Giannakopoulou D. Model checking for concurrent software architectures [Ph.D. Thesis]. Imperial College, University of London, 1999.
- [38] Cui H, Li Y, Chen J, Liu YJ. Methods with low complexity for evaluating cloud service reliability. In: *Proc. of the 16th Int'l Symp. on Wireless Personal Multimedia Communications (WPMC)*. IEEE, 2013. 1–5.

附中文参考文献:

- [9] 崔巍,李益发,斯雪明.基于 Eucalyptus 的基础设施即服务云框架协议设计. *电子与信息学报*,2012,34(7):1748–1754.
- [11] 司冠南,任宇涵,许静,杨巨峰.基于贝叶斯网络的网构软件可信性评估模型. *计算机研究与发展*,2012,49(5):1028–1038.
- [12] 刘川意,林杰,唐博.面向云计算模式运行环境的可信性动态验证机制. *软件学报*,2014,25(3):662–674. <http://www.jos.org.cn/1000-9825/4447.htm> [doi: 10.13328/j.cnki.jos.004447]
- [13] 朱俊,郭长国,吴泉源.基于广义随机 Petri 网的 Web 服务交互行为环境模型. *计算机研究与发展*,2012,49(1):2450–2463.
- [18] 李力行,金芝,李戈.基于时间自动机的物联网服务建模和验证. *计算机学报*,2011,49(8):1365–1377. [doi: 10.3724/SP.J.1016.2011.01365]
- [26] 中国电子技术标准化研究所.云计算标准研究报告.北京,2010.
- [27] 王惠莅,杨晨,杨建军.云计算安全和标准研究. *信息技术与标准化*,2012(005):16–19.
- [28] 王惠莅,罗锋盈,杨建军.云计算服务安全关键标准研究. *信息技术与标准化*,2013,11:6–9.
- [35] 李梦君,李舟军,陈火旺.基于进程代数安全协议验证的研究综述. *计算机研究与发展*,2004,41(7):1093–1103.



赵波(1972 -),男,湖北武汉人,博士,教授,博士生导师,主要研究领域为可信计算,信息安全.



向驥(1984 -),男,博士,主要研究领域为可信计算,云计算安全.



戴忠华(1978 -),男,博士,副研究员,博士生导师,主要研究领域为可信计算,信息安全.



陶威(1990 -),男,主要研究领域为可信计算,信息安全.