软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2024,35(3):1357−1376 [doi: 10.13328/j.cnki.jos.006809]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

混合云环境下面向数据生命周期的自适应访问控制*

林 莉[1,2], 毛新雅[1,2], 储振兴[1,2], 解晓宇[1,2]

[1](北京工业大学 信息学部 计算机学院, 北京 100124)
[2](可信计算北京市重点实验室 (北京工业大学), 北京 100124)
通信作者: 林莉, E-mail: linli_2009@bjut.edu.cn

摘　要: 混合云模式下企业业务应用和数据经常跨云流转迁移, 面对多样复杂的云服务环境, 当前大多数混合云应用仅以主体为中心制定数据的访问控制策略并通过人工调整策略, 无法满足数据在全生命周期不同阶段时的细粒度动态访问控制需求. 为此, 提出一种混合云环境下面向数据生命周期的自适应访问控制方法 AHCAC. 该方法首先采用基于关键属性的策略描述思想去统一混合云下数据全生命周期的异构策略, 尤其引入"阶段"属性显式标识数据的生命周期状态, 为实现面向数据生命周期的细粒度访问控制提供基础; 其次针对数据生命周期同阶段策略具有相似性和一致性的特点, 定义策略距离, 引入基于策略距离的层次聚类算法实现数据生命周期各阶段对应访问控制策略的构建; 最后通过关键属性匹配实现当数据所处阶段变化时, 触发策略评估引擎上数据对应阶段策略的自适应调整和加载, 最终实现面向数据生命周期的自适应访问控制. 在 OpenStack 和开源策略评估引擎 Balana 上通过实验验证了所提方法的有效性和可行性.

关键词: 混合云; 数据生命周期; 访问控制; 层次聚类; 策略自适应调整
中图法分类号: TP311

中文引用格式: 林莉, 毛新雅, 储振兴, 解晓宇. 混合云环境下面向数据生命周期的自适应访问控制. 软件学报, 2024, 35(3): 1357–1376. http://www.jos.org.cn/1000-9825/6809.htm
英文引用格式: Lin L, Mao XY, Chu ZX, Xie XY. Adaptive Access Control Oriented to Data Life Cycle in Hybrid Cloud Environment. Ruan Jian Xue Bao/Journal of Software, 2024, 35(3): 1357–1376 (in Chinese). http://www.jos.org.cn/1000-9825/6809.htm

Adaptive Access Control Oriented to Data Life Cycle in Hybrid Cloud Environment

LIN Li[1,2], MAO Xin-Ya[1,2], CHU Zhen-Xing[1,2], XIE Xiao-Yu[1,2]

[1](College of Computer Science, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China)
[2](Beijing Key Laboratory of Trusted Computing (Beijing University of Technology), Beijing 100124, China)

Abstract: In a hybrid cloud environment, enterprise business applications and data are often transferred across different cloud services. For complex and diversified cloud service environments, most hybrid cloud applications adopt access control policies made around only access subjects and adjust the policies manually, which cannot meet the fine-grained dynamic access control requirements at different stages of the data life cycle. This study proposes AHCAC, an adaptive access control method oriented to data life cycle in a hybrid cloud environment. Firstly, the the policy description idea based on key attributes are employed to unify the heterogeneous policies of the full life cycle of data under the hybrid cloud. Especially, the "stage" attribute is introduced to explicitly identify the life-cycle state of data, which is the basis for achieving fine-grained access control oriented to data life cycle. Secondly, in view of the similarity and consistency of access control policy with the same life-cycle stage, the policy distance is defined, and a hierarchical clustering algorithm based on the policy distance is proposed to construct the corresponding data access control policy in each life-cycle stage. Finally, when the life-cycle stage of data is changed, the adaptation and loading of policies of corresponding data stages in the policy evaluation are triggered through key attribute matching, which realizes the adaptive access control oriented to the data life cycle. This study also conducts experiments to verify the effectiveness and feasibility of the proposed method on OpenStack and open-source policy evaluation engine Balana.

* 基金项目: 国家自然科学基金 (61502017)
收稿时间: 2021-05-14; 修改时间: 2022-06-11, 2022-07-26; 采用时间: 2022-10-11; jos 在线出版时间: 2023-04-26
CNKI 网络首发时间: 2023-04-27

Key words: hybrid cloud; data life cycle; access control; hierarchical clustering; adaptive policy adjustment

1 引 言

目前越来越多的企业选择混合云作为主要部署模式, 企业数据依据用途会被部署到不同云端[1]. 数据在整个生命周期依据业务需求流动于多云之间, 由不同的云平台对其进行管理, 故混合云环境下跨云数据在全生命周期中各阶段所面对的用户和所处环境是复杂且多变的. 同时, 数据的生命周期由业务场景决定, 各阶段的安全需求会随业务场景的变化而改变. 因此, 如何在混合云环境下有效保障数据全生命周期安全成为云计算发展和应用的重要问题之一.

访问控制技术一直是解决数据安全与隐私保护问题的有效方法之一, 然而当前云环境所应用的访问控制方法无论是传统访问控制模型[2−5], 或是以传统访问控制模型为基础的优化方法[6−9], 亦或是基于风险评估或信任评估的动态访问控制方法[10−15], 主要都是以主体为中心设定访问控制策略[16]. 由于混合云环境下跨云业务数据的安全需求和其所处生命周期阶段息息相关, 并且不同生命周期阶段数据所处云端和所面临用户行为可能不同, 因此, 以主体为中心进行访问控制的现有方法无法满足数据在复杂混合云环境下不同生命周期阶段具有的细粒度、多样化的动态安全需求.

在混合云环境下实现以数据为主的自适应访问控制需要解决以下 3 个难点: 1) 混合云环境下不同云服务商采用不同的策略语言, 描述方式多样, 如何在将异构策略轻量化统一描述的同时发现数据不同阶段策略间的关联性不是易事. 2) 当前策略构建方式多以主体为中心, 无法满足对数据细粒度保护的安全需求, 如何结合数据生命周期特性细粒度构建策略是一个难点. 3) 当处于不同生命周期阶段的数据在不同云平台间流动时, 如何在策略评估引擎上自适应加载对应阶段的访问控制策略也是一个难点.

为此, 本文提出一种混合云环境下面向数据生命周期的自适应访问控制方法 (adaptive hybrid cloud access control, AHCAC). 该方法首先借鉴基于属性的策略描述思想统一处理混合云环境下数据全生命周期的异构策略, 尤其引入“阶段”属性显式标识数据的生命周期状态, 为实现基于数据生命周期的细粒度访问控制提供基础; 其次定义策略距离, 引入基于策略距离的层次聚类算法实现数据生命周期各阶段对应访问控制策略的构建; 最后通过关键属性匹配实现当数据所处阶段变化时, 触发策略评估引擎上对应阶段策略的自适应调整和加载, 最终实现面向数据生命周期的自适应访问控制.

与现有工作相比, 本文提出方法 AHCAC 的创新点如下.

(1) 针对当前混合云环境下存在的策略描述方式各异和未支持面向数据生命周期的细粒度控制等问题, 基于关键属性统一简化描述异构策略, 同时显式引入“阶段”属性用于标识数据所处生命周期阶段, 以便后续构建策略及通过阶段关联触发不同状态下策略自适应调整.

(2) 针对数据生命周期同阶段策略具有的相似性和一致性特点, 定义策略距离计算方式, 引入基于策略距离的层次聚类算法挖掘数据的生命周期阶段, 完成阶段与访问控制策略之间的指派, 实现面向数据生命周期的细粒度策略构建.

(3) 针对现有策略评估引擎需要自适应策略的问题, 给出面向数据生命周期的访问控制策略自适应加载方案, 通过在混合云平台内部各子网网关处监测数据生命周期状态, 结合数据的“阶段”属性, 采用关键属性匹配方式触发对应阶段访问控制策略集的自适应覆盖.

(4) 已在 OpenStack 和开源策略评估引擎 Balana 上对提出方法进行实验验证. 通过与采用欧氏距离和余弦距离的层次聚类算法比较聚类效果来证明策略构建的准确性; 通过在 Balana 上比较本文提出方法部署前后的性能差异来证明策略自适应调整的效果; 通过与主流的 RBAC 进行性能评估来证明本文提出方法能够有效提升混合云平台的访问控制效率.

2 相关工作

下面从策略描述语言、云计算访问控制和策略构建 3 个方面进行相关现状介绍.

访问控制策略描述语言是实现访问控制的重要基础. 近年来, 已有较多的策略描述语言被提出, 比如国际标准组织 OASIS 提出的 XACML (extensible access control markup language)[17]、SPL (security policy language)[18]等. 同时, 一些云服务提供商也提出自己的策略语言, 例如, Amazon Web services (AWS) 服务采用自定义的访问控制策略语言. 此外, 当前混合云环境多采用松散式访问控制方法, 不同云平台采用不同的访问控制策略描述方法, 策略语言的不统一使其在语义和语法上易产生歧义, 存在一定的数据安全隐患.

传统相对静态的访问控制技术无法直接应用于复杂动态的云计算环境. 例如, 基于角色的访问控制 RBAC[4,5]依赖于角色属性, 动态性较差, 需要采用增加虚拟角色的方式来满足云环境下数据的动态访问控制需求, 这会极大增加角色管理的难度和成本, 因此 RBAC 方法无法适用于复杂动态的混合云环境. 为此, 国内外学者基于传统访问控制技术提出一些云计算访问控制方法[19]. 例如, Weng 等人[6]提出了基于 BLP 和 BIBA 模型的云计算访问控制方法, 通过综合角色、时间和环境等信息来实施访问控制, 侧重保密性策略和完整性策略, 主要适用于等级性强、权限较固定的云平台, 而不适用于混合云中因数据变化、主体行为变化而导致权限动态变化的应用场景. Zhu 等人[7]提出了一种基于 UCON 的云计算访问控制方法, 基于角色、安全级别等安全属性以及义务和条件决定是否授予云用户访问权限, 然而云环境的开放性使得用户属性易变, 这给云服务的安全授权管理及安全策略存储增加了难度, 因此该方法不适应于混合云平台.

也有一些学者提出了以主体为主的动态访问控制方法, 如李凤华等人[8]等提出一种基于行为的云计算访问控制方法, 将角色、时态和环境这 3 种属性定义为主体行为, 基于主体行为进行授权. 杨宏宇等人[9]提出一种 Hadoop 云平台动态访问控制方法, 先对 Hadoop 平台合法用户行为进行分类, 再通过监控用户操作行为进行评估, 进而基于评估值对用户实行动态访问控制. Chen 等人[10]提出基于 ABAC 的动态风险访问控制方法, 利用数据流滑窗技术计算访问请求者当前的风险值, 通过与风险阈值比较来实现动态访问控制. Wang 等人[11]提出一种基于服务级协议和隐私感知的动态云服务信任评估方法, 采用信任模型, 通过动态计算用户的信任值来避免恶意用户的访问. 范运东等人[12]基于用户行为定义综合信任值, 基于综合信任值动态确定用户信任等级, 根据信任等级完成授权. 张凯等人[13]基于用户行为证据计算其信任等级, 根据该等级激活对应角色以及访问权限. 丁红发等人[14]提出基于演化博弈的隐私风险自适应访问控制模型, 该模型基于香农自信息设计动态访问数据集的隐私量化方法, 并基于其构造访问请求隐私风险函数和用户隐私风险函数, 给出多参与者博弈的访问控制演化稳定策略选取方法. Zhao 等人[15]提出混合云计算环境中具有动态信任的基于属性访问控制方法, 在 ABAC 基础上增加了信任属性, 通过计算访问用户的信任属性值实现动态授权. 然而动态访问控制方法不仅要关注授权的动态性, 更需要关注授权的效率. 当前混合云环境复杂, 存在较多业务交互场景, 同时云用户和数据数量庞大, 因此当数据场景频繁变化时, 以主体为中心的策略访问控制方法无法快速响应大批量用户的并发请求, 存在策略评估时间长, 评估效率低等问题.

策略构建和调整是实现访问控制的关键, 特别是针对混合云环境这样具有大规模用户的大型信息系统更是一个难点. 目前在这一方面研究较多的是自底向上的 RBAC 策略构建方法, 即角色挖掘[20], 该方法主要应用于已运行过访问控制的遗留系统[21,22]. Schlegelmilch 等人[23]提出了基于层次聚类的角色挖掘方法, 通过分析权限集之间的层次关系构建了角色树. Ene 等人[24]将角色挖掘转化为最小二部图覆盖问题, 并设计出基于图论的角色挖掘算法. 随着机器学习技术的发展, 针对角色挖掘出现了一些新的聚类方法. 例如, Choobdar 等人[25]提出通过采集节点当前位置及其历史行为数据进行集合式角色聚类. Ni 等人[26]提出一种基于监督式学习的角色自动化配置方法. 上述工作侧重构建以主体为主的 RBAC 策略, 相比数据各生命周期阶段, 其得出的角色粒度加粗, 无法有效满足混合云环境下跨云业务数据在不同生命周期阶段的细粒度动态安全需求.

为此, 本文提出混合云环境下面向数据生命周期的自适应访问控制方法. 该方法基于关键属性统一简化描述异构策略, 同时显式引入“阶段”属性用于标识数据所处生命周期阶段的方式, 以解决当前混合云环境下存在的策略描述方式各异和未支持面向数据生命周期的细粒度控制等问题; 同时, 基于数据生命周期同阶段策略具有的相似性和一致性特点, 通过定义策略距离计算方式, 引入基于策略距离的层次聚类算法挖掘数据的生命周期阶段, 完成阶段与策略之间的指派, 实现面向数据生命周期的细粒度策略构建, 以解决当前以主体为主的策略构建方法存在的动态性较弱、无法满足混合云环境下跨云业务数据细粒度安全保护需求的问题; 此外, 该方法给出面向数据

生命周期的策略自适应加载方案, 通过在混合云平台内部各子网网关处监测数据生命周期状态, 结合数据的"阶段"属性, 采用关键属性匹配方式触发对应阶段访问控制策略集的自适应覆盖, 以解决现有动态策略评估引擎存在的策略评估时间长, 评估效率低等问题.

3 问题描述

本文提出方法针对当前主流的混合云应用场景, 如图 1 所示, 主要涉及用户、网络环境、混合云平台 3 部分, 其中, 不同用户通过有线、无线等网络环境部署业务到混合云平台, 混合云平台包含公有云、私有云, 各云之间通过 VPN、专线等方式实现通信[1]. 当用户业务跨云交互时, 业务数据会流转于多个云平台, 即在数据生命周期 (这里指数据从创建到销毁的全过程) 的不同阶段, 数据状态及所处云平台环境可能不同, 因而数据访问控制策略应根据数据状态变化动态调整. 当不同用户访问云端数据时, 应根据数据当前阶段的策略来实现安全访问.

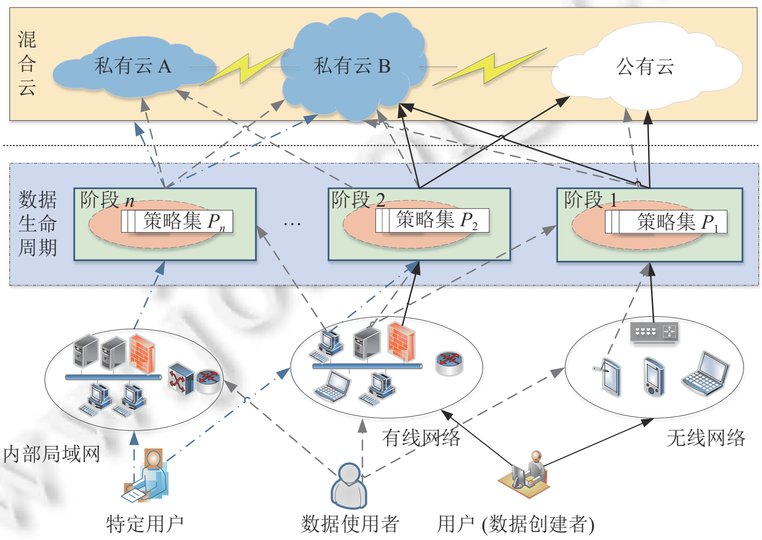


图 1 混合云环境下跨云业务数据流通场景

以广电行业某电视台的混合云系统为例, 其承载了媒资数据采集、节目制作、多平台播出等复杂融媒体业务. 从全局来看, 被不同业务处理的媒资数据具有典型生命周期特点, 可细分为创建、转码、编辑、审核、备播、发布、存储、删除等生命周期阶段. 数据在不同阶段间的流动方向由业务场景决定, 如已发布并处于存储阶段的媒资数据若需要在其他平台播出, 则该数据需要根据业务需求被重新转码、编辑、备播和发布, 即数据在生命周期中的流动方向并非单向. 同时, 电视台混合云系统包含新闻云、综合云等多个平台, 有私有云也有公有云, 媒资数据会依据业务需求灵活存储于不同云平台中, 如用于新闻节目的媒资数据可能被综合节目采用.

由于不同业务人员对不同阶段媒资数据的操作和所处网络环境可能不同, 故不同阶段的访问控制策略不一定相同. 仍以电视台混合云系统为例, 在创建阶段, PGC (professional generated content)、UGC (user-generated content) 用户在公有云环境创建媒资数据; 在转码阶段, 由转码人员在私有云环境下对上传后的媒资数据进行转码和制作; 在编辑阶段, 由编辑人员在私有云环境下对其编辑与修改; 在审核阶段, 由审核人员在私有云环境下对媒资数据进行审核判定; 在备播阶段, 媒资数据通常存储在云端待播, 由备播人员对其进行管理; 在发布阶段, 媒资数据转到业务系统进行播放, 由监测人员对其进行监测. 在删除阶段, 由管理人员定期删除在公有云存储的冗余媒资数据. 容易看出, 在上述整个业务交互流程中, 因媒资数据在不同阶段面临的用户、业务场景和网络环境不同, 所以其在不同阶段具备不同访问控制策略. 另外, 对数据生命周期的某一阶段, 因数据所处环境和用户访问行为基本相似, 故该阶段数据的安全保护需求通常相似, 因而该阶段数据访问控制策略也具备相似性.

此外, 在混合云环境下的跨云业务交互中, 由于数据在不同阶段所处的业务场景通常有关联, 所以数据在不同

阶段的访问控制策略也具备一定关联. 如在创建阶段, 用户可通过无线或有线网络访问公有云来创建制作媒资数据, 拥有对数据读取、修改和删除的权限. 当媒资数据流转至私有云环境下台内编辑专用服务器时会进入下一阶段, 此时将收回数据所有者的修改和删除权限, 仅允许台内编辑通过内网在工作时间读取和修改该媒资数据. 当媒资数据处于审核阶段时, 此时将收回台内编辑人员对其具有的读取和修改权限, 给审核人员赋予读取权限. 当媒资数据处于备播阶段时, 此时将收回审核人员具有的读取权限, 允许备播人员对其具有读取和删除权限. 当媒资数据处于发布阶段时, 将收回备播人员的权限, 给监测人员指派读取和删除的权限. 当媒资数据处于删除阶段时, 仅允许台内特定工作人员通过内网对其进行删除. 当媒资数据存储至数据库服务器后, 仅允许台内媒资管理人员通过内网在工作时间对其读取和删除. 即数据访问控制策略应随数据生命周期阶段变化而变化.

因此, 本文基于数据生命周期的阶段特性提出混合云环境下的自适应访问控制方法, 其主要目标包括: 第一, 对数据全生命周期的策略进行统一轻量化描述, 显式考虑数据生命周期阶段, 将其定义为数据的重要属性之一. 第二, 数据生命周期阶段属性的定义需要满足业务需求和数据安全需求, 具有不同阶段属性的策略集不一定相同, 具有相同阶段属性的策略集具有相似性. 第三, 当数据所处生命周期阶段发生变化时, 策略评估引擎中对应的策略应随之自适应调整.

4 方法设计

本节首先给出方法架构, 接着介绍自适应访问控制过程, 最后详细介绍策略轻量化描述、策略构建和策略自适应调整 3 个模块.

4.1 总体架构

图 2 为本文提出方法 AHCAC 的架构图, AHCAC 基于当前主流的访问控制框架[3]构建, 主要部署在混合云平台的授权模块中.

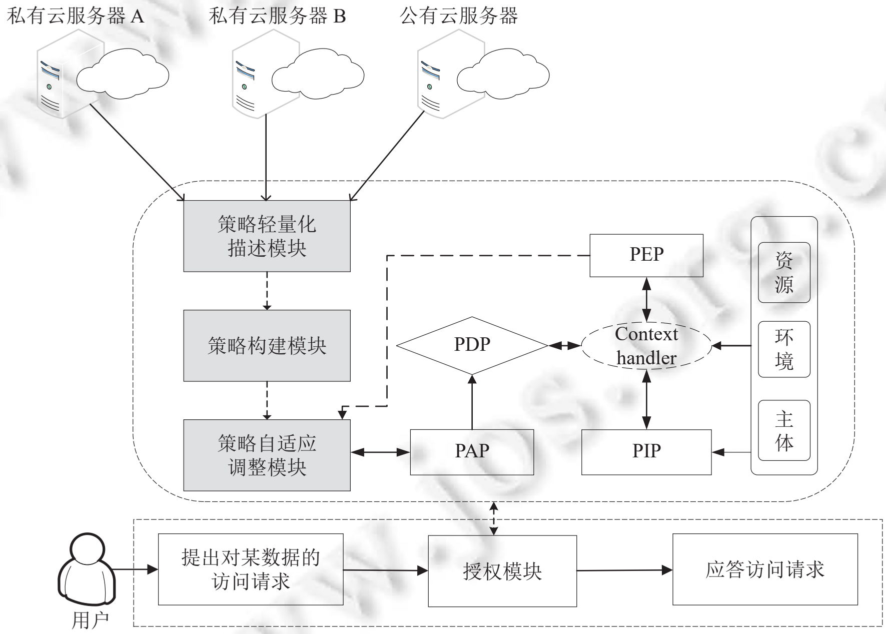


图 2 AHCAC 架构

策略管理点 (policy administration point, PAP) 负责制定策略集. 策略执行点 (policy enforcement point, PEP) 负责接收用户对数据的授权请求, 并根据策略决策点 (policy decision point, PDP) 反馈的决策结果执行相应的操作. 策略信息点 (policy information point, PIP) 负责获取如主体、资源和环境等相关属性信息. PDP 负责基于 PAP 中

策略以及 PIP 提供的相关属性信息进行访问控制决策. 上下文处理器负责解析和转发各模块间的消息. 策略轻量化描述模块负责对数据全生命周期的访问控制策略集进行统一格式化处理. 策略构建模块基于上一模块处理好的策略集, 负责为数据生命周期各阶段构建出对应的策略集. 策略自适应调整模块在前两个模块工作的基础上, 负责基于数据生命周期状态自适应调整对应的阶段策略集.

AHCAC 方法的具体工作流程如下: 首先, 策略轻量化描述模块对混合云环境下各云平台提供的异构访问控制策略进行统一描述. 因为访问控制策略本质规定了某环境下某主体对某资源可进行何种操作, 故该模块先使用静态代码规范工具对异构策略进行预处理, 然后基于资源属性进行初步分类, 接着采用关键词遍历方式提取异构策略的主体属性、操作属性、环境属性和效用属性等关键属性, 基于关键属性统一轻量化描述异构策略. 同时, 考虑到数据在不同生命周期阶段安全保护需求不同, 引入“阶段”属性显式标识数据的生命周期状态. 然后, 将策略轻量化描述模块处理好的数据策略集传送给策略构建模块. 该模块首先基于统一描述策略中的关键属性计算出各策略间的距离. 接着, 基于该距离通过层次聚类算法生成数据的生命周期阶段, 并完成阶段与权限的指派, 从而构建出数据生命周期各阶段对应的访问控制策略集. 当生命周期阶段明确后, 由资源管理者根据业务场景和实际安全需求定义阶段属性标识. 最后, 将融入阶段属性标识的策略集覆盖存入策略数据库中. 接着, 考虑到数据的安全需求及用户的访问行为跟数据所处阶段紧密相关, 因此仅在 PAP 中存储数据当前所处阶段的策略集. 当部署在各云平台子网网关的探针检测到数据发生流动时, 会调用策略自适应调整模块, 通过基于关键属性匹配的方式决定是否调整 PAP 中的策略集. 当用户发起云服务访问请求时, 对应的云服务 API (application programming interface) 将连接至授权服务器并发送该请求. 授权服务器接收请求并依据当前数据所处阶段的访问控制策略进行验证, 然后反馈授权结果. API 得到响应后执行相应操作.

4.2 策略轻量化描述

混合云环境下跨云业务交互的数据访问控制策略由各云平台提供, 具备复杂多样的特点. 而现有细粒度访问控制策略语言大多基于 XACML 语言描述, 格式复杂, 不便于大规模存储, 因此, 需要在不改变策略语义的前提下, 对数据全生命周期的异构策略进行轻量化处理. 考虑到访问控制策略本质是描述某环境下某主体对某资源可以进行何种操作, 以及各云服务访问控制策略均包含多种通用的属性类型, 故本文提取主体属性、资源属性、操作属性、环境属性和效用属性等关键属性, 基于关键属性对策略进行轻量化描述. 另外, 由于基于属性策略描述本身具备可扩展性, 故本文提出方法应能支持未来混合云环境下更复杂业务应用的策略描述场景.

如图 3 所示, 本文提出方法首先收集各云服务组件提供的策略集, 并统一存储在策略数据库中, 然后使用如 CheckStyle、Jtest 等静态代码分析工具对策略进行预处理, 接着遍历各策略集提取出资源列表. 然后基于资源列表, 通过编写脚本文件将关键属性从各策略中提取出来, 并存入 JSON 格式的策略树对应的节点上, 叶子节点为策略属性值, 非叶子节点为策略组成结构. 同时由于不同生命周期阶段的数据安全需求不同, 因此本文依据业务场景, 显式添加“阶段”属性, 以便后续构建各阶段策略及关联触发策略自适应调整. 在策略描述时, 针对某种属性类型无对应属性值的情况, 将使用空值进行填充. 通过这种方式, 可将策略进行统一轻量化描述. 下面给出策略描述相关定义.

定义 1. 访问控制策略形式化描述为:

$$Policy=(SA,OA,OPA,EA,EF,StageMark),$$

其中, SA (subject attribute) 表示主体属性, OA (object attribute) 表示客体属性, OPA (operation attribute) 表示操作属性, 如读/写等, EA (environment attribute) 表示环境属性, EF (effect) 表示效用属性, 可取允许 (permit) 和拒绝 (deny) 两个值, $StageMark$ 表示生命周期阶段属性.

在混合云环境下, 数据在整个生命周期中会根据不同的业务应用场景和环境处于若干个不同阶段, 即数据具有不同阶段的细粒度安全需求, 因而本文显式引入“阶段”属性, 帮助实现基于阶段的策略自适应调整, “阶段”属性定义如下.

定义 2. 生命周期阶段属性 ($StageMark$) 由如下 4 个相关属性组成, 记作:

$$StageMark=(RE\_SRC,RE\_DET,CC,S\_SA),$$

其中, RE_SRC (resource environment attribute source) 指当前阶段的数据环境属性; RE_DST (resource environment attribute destination) 指目标阶段的数据环境属性; CC (change condition) 指数据可变更阶段的条件; S_SA (subject change stage attribute) 指当前阶段中可手动更改数据所处阶段的主体属性, 用于限制能够手动修改当前 PAP 存储策略集的主体权限. 不失去一般性, 当某属性缺失时, 则将值置为 $\varnothing$.

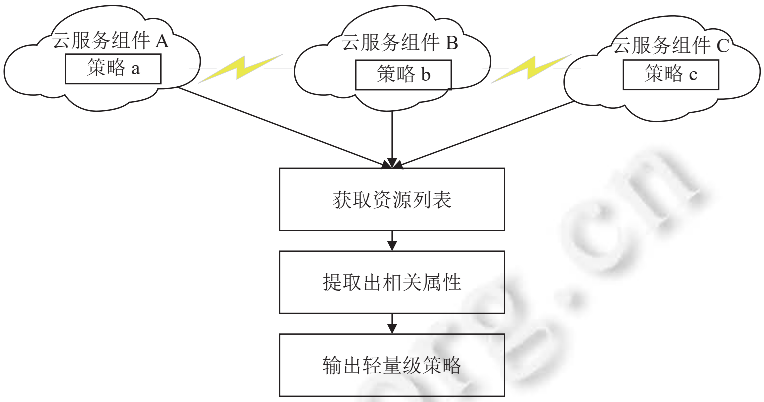


图 3　策略轻量化描述过程

表 1 给出了某数据生命周期阶段 1 和阶段 2 的属性描述示例.

表 1　生命周期阶段属性示例

阶段	示例
1	*StageMark*: {*RE_SRC*: {IP: 192.168.2.2}, *RE_DST*: {IP: 192.168.2.175}, *CC*: {Time: [9:00, 12:00]}, *S_SA*: $\varnothing$ }
2	*StageMark*: {*RE_SRC*:{IP: 192.168.2.175}, *RE_DST*: {IP: 192.168.2.180}, *CC*: {Time: [9:00, 22:00]}, *S_SA*: {IP: [192.168.1.1, 192.168.1.254]}

基于上述定义, 采用算法 1 对混合云环境下各云服务存储的异构策略集进行统一轻量化处理, 本文提出方法采用 ALFA 格式[27]存储策略. ALFA 是一种用于描述访问控制策略的伪代码语言, 其继承了 XACML 的结构和思想, 既支持细粒度描述, 可读性和扩展性较好, 又更为轻量化, 能够有效描述异构策略.

算法 1. 异构策略轻量化描述算法.

输入: 异构策略集 PS;
输出: 轻量格式化后的策略集 PN.

1. 初始化 PS, 并输出 $ResourceList$
2. **FOR** *each* $resource \in ResourceList$ **DO**
3. 　$resourceAttList = getAttributeValue(PS)$
4. 　**FOR** *each* $resourceAtt \in resourceAttList$ **DO**
5. 　　$SA = getSubject(resourceAtt)$
6. 　　$OA = resource$
7. 　　$OPA = getOperation(resourceAtt)$
8. 　　$EA = getEnviroment(resourceAtt)$
9. 　　$EF = getEffect(resourceAtt)$
10. 　　根据业务场景, 提取 $StageMark$
11. 　　$rule = create(SA, OA, OPA, EA, EF, StageMark)$

```
        policy[resource] = addRule(rule)
    END FOR
    PN = addPolicy(policy[resource])
    采用 ALFA 结构统一描述 PN
END FOR
```

下面, 通过实例说明算法 1 的作用. 表 2 为不同云服务商提供的访问控制策略样例, 3 条策略具有不同格式. 表 3 为上述各异构策略采用定义 1 的统一形式化描述, 由于当前各策略还没有对其进行阶段定义, 故此时阶段属性为空. 各策略执行算法 1 的最终结果如表 4 所示.

表 2 不同云服务商的访问控制策略样例

策略	样例

1	{
	Version: 2021-10-27,
	Statement:[{
	Action: ec2: Download,
	Effect: Allow,
	Resource: video,
	}]}

2	{
	computer:get_all:
	project_name: TENANT1
	and user_name: USER1
	and TENANT: %(project_name)s
	and VM1: %(instance_name)s
	}

3	<?xml version="1.0" encoding="utf-8"?> <Policy PolicyId="P0-1" RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides" xmlns="urn:oasis:names:tc:xacml:1.0:policy:schema:os"> <Target> <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"> <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">Role0</AttributeValue> </Subject> <Resource> <ResourceMatch MatchId="urn: oasis: names: tc:xacm:l 1.0: function:anyURI-equal"/> <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resources-id"/> <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">192.168.1.107/compute/v2.1/limits</AttributeValue> </Resource> <Action> <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"> <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"/> <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">GET</AttributeValue> </ActionMatch> </Action> </Target> <Rule Effect="Permit" RuleId="Rule0-1"> <Environment> <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:enviroment-id"> <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">None</AttributeValue> </Attribute> </Environment> </Rule> </Policy>

表 3 提取关键属性后的策略形式化描述

策略	主体属性	客体属性	操作属性	环境属性	效用属性	阶段属性
1	Name: USER1	Video	Download	∅	permit	∅
2	Name: USER1Domain: TENANT1	TENAN1/servers/VM1	Get	∅	permit	∅
3	Role: Role0	192.168.1.107/compute/v2.1/limits	Get	∅	permit	∅

表 4 采用算法 1 统一轻量化描述后的策略格式

策略	格式

1	policy example1 {
	target clause OA == "Video"
	apply first-applicable
	rule eg1 {
	target clause SA.name =="USER1"
	and OPA == "Download"
	permit }}

2	policy example2 {
	target clause OA ==
	"TENAN1/servers/VM1"
	apply first-applicable
	rule eg2 {
	target clause SA.Name == "USER1"
	and SA.Domain == "TENANT1"
	and OPA == "Get"
	permit}}

3	policy example3 {
	target clause objectType == "192.168.1.107/compute/v2.1/limits"
	apply deny-overrides
	rule eg3 {
	target clause SA.Role =="Role0"
	and OPA == "Get"
	permit}}

4.3 策略构建

基于第 4.2 节统一处理得到的策略集, AHCAC 策略构建模块负责求解数据生命周期各阶段对应的访问控制策略, 基本思想是层次聚类. 根据数据生命周期同阶段策略具有的相似性和一致性特点, 定义策略距离, 将每条策略看作一个聚类簇, 先计算出每个簇之间的距离, 然后将距离最近的聚类簇进行合并, 不断重复上述过程, 直到达到最佳聚类簇个数为止, 最终产出的聚类簇结果即为生命周期阶段. 最后为各阶段指派访问控制策略.

4.3.1 策略距离计算

策略距离是层次聚类的基础, 策略距离本质是策略中各属性间的距离. 因为属性具有字符型、数值型和范畴型等不同类型, 因此为了保证属性距离的准确性, 本文提出方法根据不同属性类型来分类计算.

另外, 由于现有策略距离计算方法不支持不同类型的策略属性, 借鉴我们之前工作[28]提出的相似度计算思想, 本文如公式 (1) 所示计算策略间的距离:

$$Distance(P_1,P_2)=\alpha_{\text{Subject}}Distance_{\text{Subject}}(P_1,P_2)+\beta_{\text{Operation}}Distance_{\text{Operation}}(P_1,P_2)+\mu_{\text{Effect}}Distance_{\text{Effect}}(P_1,P_2) \tag{1}$$

其中, $Distance(P_1,P_2)$ 表示两个策略 P_1 和 P_2 间的距离, 它由 P_1 和 P_2 的主体属性距离、操作属性距离和效用属性距离综合计算得到. 需要指出的是, 这里环境属性作为主体属性和操作属性距离的影响因子, 不单独计算, 另外, 本文主要针对数据生命周期下各阶段的策略, 数据资源都相同, 故数据资源距离也不计入计算. 具体地, 针对上述 3 种策略属性类型, 我们给出属性距离的计算方式.

(1) 字符型属性

字符型的策略属性通常由单一字符串组成, 因此这类属性通过字符匹配来计算其距离, 如公式 (2) 所示:

$$Distance_{\text{Character}}(V_1,V_2)=\begin{cases}1, & V_1\neq V_2\\ 0, & V_1=V_2\end{cases} \tag{2}$$

其中, $Distance_{\text{Character}}(V_1,V_2)$ 表示字符型属性 V_1 和 V_2 之间的距离, V_i 表示第 i 条 (i=1, 2) 策略的属性. 当策略中对应的字符型属性字符匹配相同时, 该距离为 0, 否则距离为 1.

(2) 数值型属性

数值型属性可以利用数值间的差集和并集来计算距离, 具体如公式 (3) 所示:

$$Distance_{\text{Number}}(V_1,V_2)=1-\frac{Num(V_1\cap V_2)}{Num(V_1\cup V_2)} \tag{3}$$

其中, $Distance_{\text{Number}}(V_1,V_2)$ 表示数值型属性 V_1 和 V_2 之间的距离, 交集越大, 距离越小.

(3) 范畴型属性

范畴型属性是指具有层次关系的属性, 如角色属性等, 本文方法统一用树结构描述属性间的层次关系, 范畴型属性距离如公式 (4) 计算:

$$Distance_{\text{Level}}(V_1,V_2)=\frac{Dis(V_1,V_2)}{Max(Dis)} \tag{4}$$

其中, $Distance_{\text{Level}}(V_1,V_2)$ 表示范畴型属性 V_1 和 V_2 之间的距离, $Dis(V_1,V_2)$ 表示在层次树中两属性之间的最短距离, $Max(Dis)$ 表示层次树中的最长距离.

这样, 当主体属性是角色时, 属于范畴型属性, 当主体属性是用户 ID 时, 属于字符型属性, 主体属性距离根据实际应用情况从公式 (2) 和公式 (4) 中选择对应公式进行计算; 效用属性距离使用公式 (2) 计算, 操作属性和环境属性要根据实际应用情况从公式 (2)–公式 (4) 中选择对应公式进行计算.

属性权重代表该属性在整个策略中的重要程度, 公式 (1) 中各属性权重的计算公式如下:

$$\alpha_{\text{Subject}}=\frac{N_S+N_{ES}}{N_P} \tag{5}$$

$$\beta_{\text{Operation}}=\frac{N_O+N_{EO}}{N_P} \tag{6}$$

$$\mu_{\text{Effect}}=\frac{N_E}{N_P} \tag{7}$$

其中, N_P 表示策略属性个数, 主体属性权重由全部属性中主体属性出现的频次 N_S 和环境属性中描述主体属性次数 N_{ES} 决定, 如公式 (5) 计算. 操作属性权重的计算类似主体属性权重, 如公式 (6) 计算. 效用属性权重为效用属性值占全部属性个数的比重, 如公式 (7) 计算.

如表 5 所示, 以策略 P_1 , P_2 为例, 运用公式 (1) 计算其策略距离. 首先计算属性距离, P_1 策略中的 *effect* 属性为 deny, P_2 策略中的 *effect* 属性为 deny, 根据公式 (2) 可计算出该属性距离为 1. P_1 策略中的 *action* 属性为 upload, P_2 策略中的 *action* 属性为 download, 根据公式 (2) 可计算出该属性距离为 0. P_1 策略中的 *environment* 属性为 19:00–22:00, P_2 策略中的 *environment* 属性为 20:00–24:00, 则两者的交集 20:00<time<22:00, 并集为 19:00<time<24:00, 根据公式 (3) 经过计算可得到该属性距离为 3/5. P_1 策略中的 *subject* 属性为 editor, P_2 策略中的 *subject* 属性为 technical support, 两者之间的层次关系如图 4 所示.

根据公式 (4) 可计算出该属性距离为 1/2. 因此, 可得出:

$$\begin{cases}Distance_{\text{Subject}}(P_1,P_2)=1/2\\ Distance_{\text{Operation}}(P_1,P_2)=3/5\\ Distance_{\text{Effect}}(P_1,P_2)=1\end{cases}.$$

其次计算属性权重, 其中 P_1 策略中 N_P 为 4, N_S 为 1, N_O 为 1, N_E 为 1, *environment* 属性用于描述操作行为, 故 N_{EO} 为 1, 因此 P_1 策略的属性权重分别为 $\alpha_{\text{Subject}\_P_1}=1/4$, $\beta_{\text{Operation}\_P_1}=2/4$, $\mu_{\text{Effect}\_P_1}=1/4$. P_2 策略中 N_P 为 5, N_S 为 2, N_O 为 1, N_E 为 1, *environment* 属性同样用于描述操作行为, 故 N_{EO} 为 1, 因此 P_2 策略的属性权重分别为 $\alpha_{\text{Subject}\_P_2}=2/5$, $\beta_{\text{Operation}\_P_2}=2/5$, $\mu_{\text{Effect}\_P_2}=1/5$. P_1 和 P_2 之间的属性权重分别为:

$$\begin{cases} \alpha_{\text{Subject}} = (\alpha_{\text{Subject\_}P_1} + \alpha_{\text{Subject\_}P_2})/2 = 13/40 \\ \beta_{\text{Operation}} = (\beta_{\text{Operation\_}P_1} + \beta_{\text{Operation\_}P_2})/2 = 18/40 \\ \mu_{\text{Effect}} = (\mu_{\text{Effect\_}P_1} + \mu_{\text{Effect\_}P_2})/2 = 9/40 \end{cases}.$$

由此, 根据以上数值, 基于公式 (1) 可计算出 P_1 策略和 P_2 策略之间的距离为 0.657 5.

表 5　用于说明策略距离计算的策略实例

策略	实例

P_1	{
	Subject: Editor,
	Action: Upload,
	Enviornment: time: [19:00, 22:00],
	Effect: Deny
	}

P_2	{
	Version: 2021-10-27,
	PolicyId: P_2
	Statement: [
	{
	Subject: Technical support, Editor
	Action: ec2: Download,
	Effect: Deny,
	Environment: [20:00, 24:00]
	}
	]
	}

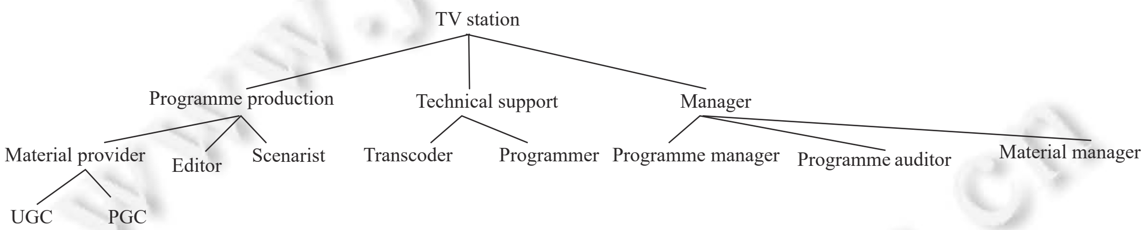


图 4　主体层次关系

策略距离计算算法如算法 2 所示.

算法 2. 策略间距离计算算法.

输入: 两条策略 P_1 和 P_2;

输出: 策略间距离 *Distance*.

```
FOR 策略 P1 的主体属性 P1.SA1 和策略 P2 的主体属性 P2.SA2 DO
    IF P1.SA1 ≠ P2.SA2 THEN
        判断主体属性的类型
        计算距离 Distance_Subject
END FOR
FOR 策略 P1 的操作属性 P1.OPA1 和策略 P2 的操作属性 P2.OPA2 DO
    IF P1.OPA1 ≠ P2.OPA2 THEN
        计算距离 Distance_Operation
END FOR
```

FOR 策略 P_1 和 P_2 的环境属性 **DO**
    判断环境属性的类型
    判断描述主体 or 操作
    计算距离 $Distance_{\text{Environment}}$
END FOR
计算 P_1 和 P_2 效用属性距离 $Distance_{\text{Effect}}$
采用公式 (1) 计算 $Distance(P_1,P_2)$

4.3.2 阶段划分及阶段策略生成

基于公式 (1) 可以计算出同一资源全生命周期内各策略间的距离, 将其通过二维矩阵存储起来, 建立距离表. 首先将每一条策略都看作是一类, 将其看作是坐标轴上的一点, 策略间的差异作为各点之间的距离, 根据各个点之间不同的平均距离进行聚类, 聚类后生成一层新阶段, 如此循环直至生成最后的层次聚类树. 聚类过程示例如图 5 所示.

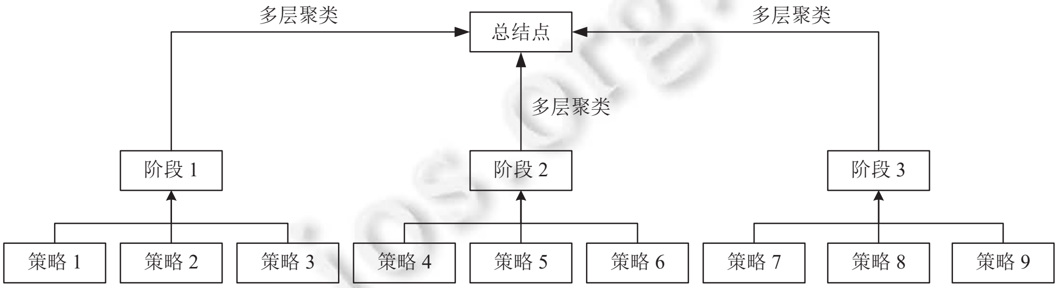


图 5 AHCAC 层次聚类过程示例

为了保证客观性, 在聚类之前, 聚类个数根据调整兰德系数 (adjusted Rand index, *ARI*)[29]算出. *ARI* 系数由 Hubert 等人于 1985 年提出, 是经典聚类效果评价指标之一, 因 *ARI* 指标不受所选聚类算法影响, 因此本文选择 *ARI* 指数作为确定最佳聚类簇数的标准, 具体计算公式如 (8) 所示:

$$ARI=\frac{\sum_{ij}\binom{n_{ij}}{2}-\left[\sum_i\binom{n_i}{2}\sum_j\binom{n_j}{2}\right]\Big/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{n_i}{2}+\sum_j\binom{n_j}{2}\right]-\left[\sum_i\binom{n_i}{2}\sum_j\binom{n_j}{2}\right]\Big/\binom{n}{2}} \tag{8}$$

其中, $\sum_{ij}\binom{n_{ij}}{2}$ 表示聚类后实际输出的簇数. $\frac{1}{2}\left[\sum_i\binom{n_i}{2}+\sum_j\binom{n_j}{2}\right]$ 指最大簇数, 表示策略聚类最完美的簇数, $\left[\sum_i\binom{n_i}{2}\sum_j\binom{n_j}{2}\right]\Big/\binom{n}{2}$ 指期望簇数, 表示策略矩阵中通过将行配对方法数量与列配对方法数量相乘, 然后除以总配对方法数量所求得的平均策略聚类簇数. 即 *ARI* 为策略实际聚类后的簇数与理想状态下的最大聚类簇数各自减去期望策略聚类簇数后的比值. 当比例越接近于 1, 则表明聚类后的簇数越合理.

本文将设定一个虚拟且拥有所有策略属性的点设定为原点, 在每次聚类前, 首先将本层阶段策略与原点之间的距离和阶段策略之间的距离导入至 *ARI* 指数计算函数中, 然后选择使 *ARI* 指数最大的聚类簇数作为本轮聚类簇数的输入值, 以保证聚类效果最佳. 聚类的迭代次数由 *ARI* 指数决定, 若所获取的聚类簇数与上一轮簇数相同, 则表明当前聚类效果最佳, 停止迭代聚类并输出聚类结果. 最后采用树来存储阶段聚类结果. 阶段策略层次聚类算法如算法 3 所示.

算法 3. 阶段策略层次聚类.

输入: 未聚类的策略集 P;
输出: 阶段生成树 $Stage\_T$.

WHILE 层次聚类循环
  进行阶段与权限指派
  调用策略距离计算函数
  将距离传入 *ARI* 函数, 得到最佳聚类簇数 *num*
    IF *num* 不等于上一轮簇数 **THEN**
      将距离和 *num* 传入层次聚类函数 *AgglomerativeClustering*
      执行下一轮聚类
    ELSE
      输出阶段生成树 *Stage_T*
END WHILE

4.3.3 阶段与权限的指派

在每轮层次聚类后, 会进行阶段与权限的指派. 每轮层次聚类后, 会保留策略 ID 和其在每一层对应的阶段标识. 当生成最后的阶段树后, 各阶段对应的所有权限就来自具备对应阶段标识的去重策略的集合. 因为涉及多轮聚类, 所以本文规定同一策略拥有阶段标识越大, 代表其聚类轮次越靠后. 具体指派算法如算法 4 所示. 当策略集按照数据生命周期阶段进行划分后, 资源管理者进一步根据业务场景和实际安全需求为数据制定各个阶段的策略, 如可通过提取 IP 地址等数据网络环境属性、匹配数据可变更阶段的时间区间条件属性等自主补充定义 2 所描述的生命周期阶段属性 *StageMark* .

算法 4. 阶段-权限指派算法.

输入: 阶段集 *S*、策略集 *P* 和各策略对应的阶段标识 *S_Mark*;
输出: 各阶段拥有的权限 *S_P*.

创建阶段权限数组 *S_P*
FOR 循环匹配所有阶段
  IF 某策略 P_i 阶段标识 S_Mark_i 属于阶段 S_i **THEN**
    IF P_i 不属于阶段 S_i **THEN**
      将 P_i 加入至阶段权限数组 $S\_P[S_i]$

4.4 策略自适应调整

第 4.3 节已将策略集按照数据生命周期阶段进行划分, 并由资源管理者根据业务场景来定义策略集中的生命周期阶段属性, 得到数据各个阶段的策略集. 因为在混合云环境下, 策略集往往数量比较庞大, 因此为了提高策略评估效率, 我们在 PAP 中仅存储数据当前阶段的策略集, 并且基于数据状态自适应加载 PAP 中的策略集, 通过优化策略数量和规模来提高评估效率. 基于数据状态的访问控制策略自适应加载时序图如图 6 所示.

具体地, 在数据创建初期, PAP 默认加载该数据第 1 阶段的策略集. 同时在各云平台子网网关处部署探针, 这里的探针是指基于特征属性监测数据状态的工具, 比如 NetFlow 生成与采集管理工具 Fprobe、数据流收集和分析工具集 Flow-tools 等. 当所标记的数据包流经所部署的关键点时, 探针将提取数据包中的关键属性 (如源 IP 地址、目的 IP 地址、源端口和目的端口等), 并发送数据阶段调整请求给 PEP, PEP 将该请求发送至上下文处理器, 上下文处理器解析请求并提取数据的环境属性 (如网络环境等)、条件属性等阶段属性后, 先将其与 PAP 现阶段策略集的阶段属性即 *StageMark* 中的 *RE_SRC*, *RE_DET*, *CC* 和 *S_SA* 等进行匹配, 若一致则无需更新加载. 若有任一项属性不匹配, 则进行第 2 步匹配, 将相关属性与存储在策略数据库中数据各阶段策略集中的阶段属性进行匹配. 因为阶段属性具有唯一性, 所以若有匹配成功的策略集, 则将该策略集加载至 PAP 中, 并删除前一阶段策略集. 若不存在匹配的策略集, 则 PAP 中策略集保持不变, 记录匹配不成功次数, 以备后续调整更新策略库.

与各策略集中阶段属性进行匹配的相关属性如定义 2 所述. 基于数据状态的访问控制策略自适应加载的核心匹配算法详见算法 5.

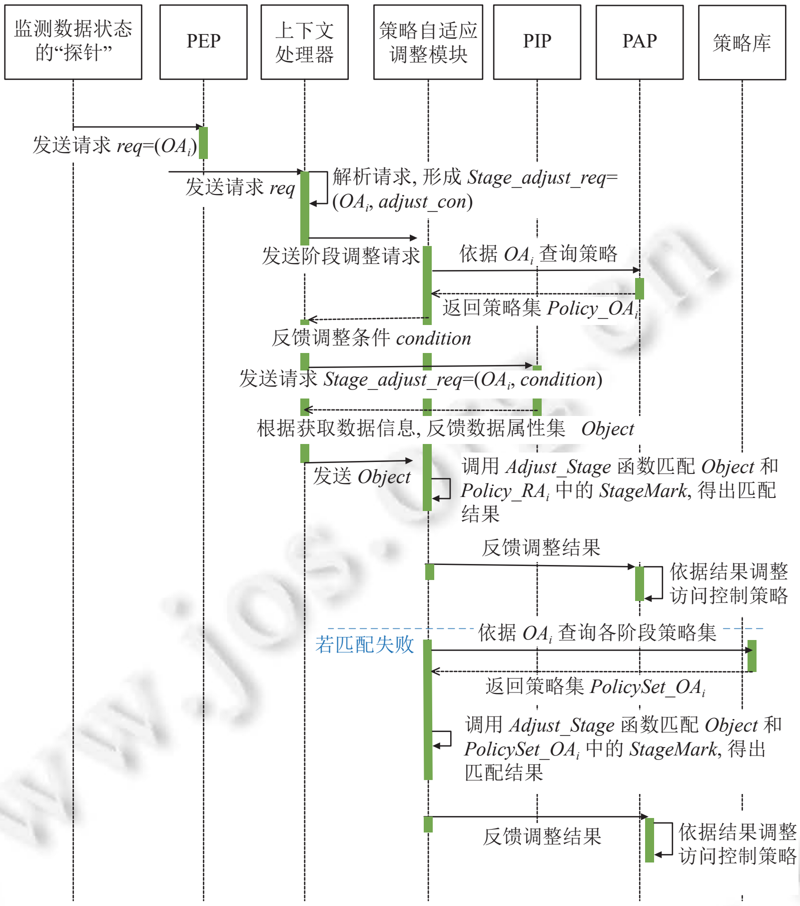


图 6 基于数据状态的访问控制策略自适应加载时序图

算法 5. 生命周期阶段调整算法.

输入: 数据 object 的 $\{RE\_SRC_0, RE\_DST_0, CC_0, S\_SA_0\}$, 策略集 $Policy\_RA_i$ 的属性之一 $StageMark$;
输出: 匹配结果 True/False.

IF 数据生命周期阶段需调整 **THEN**
    调用 $Adjust\_Stage$ 函数
    IF $RE\_SRC_0 \in RESRC_i, RE\_DST_0 \in REDST_i, CC_0 \in CC_i, S\_SA_0 \in S\_SA$ **THEN**
        返回 True
    ELSE
        返回 False

5 实验评估

为了验证 AHCAC 的有效性, 本文参考广电行业某电视台真实融合媒体混合云系统架构, 基于 OpenStack 构建实验环境进行实验评估.

5.1 实验环境

本文基于 OpenStack 模拟了混合云平台, 并基于开源策略评估引擎 Balana 部署了 AHCAC 原型, 用以替换原授权功能, 执行过程如图 7 所示.

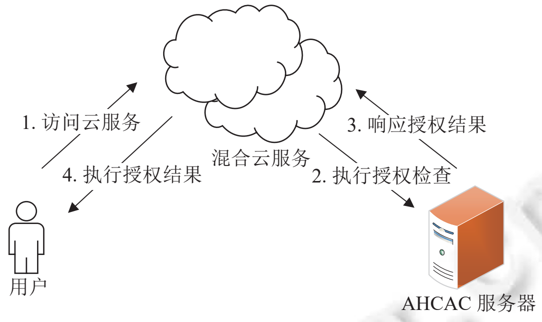


图 7 混合云环境下 AHCAC 执行访问控制的过程

该平台统一集中化管理混合云环境的访问控制策略, 用户在访问云服务之前, 需要首先通过管理平台的授权验证. 因为开源策略评估引擎 Balana 是由 SunXACML 发展而来, 所以其具有 XACML 的基础模块, 因此本方法基于 Balana 进行部署. 如图 8 所示, 以 Keystone 为例, 我们在 OpenStack 的各相关部件中部署了 PEP 和 AHCAC 客户端, PEP 通过 API 获取必要信息并构建相关请求, 然后将请求通过 AHCAC 客户端发送至 AHCAC 服务器. AHCAC 服务器中部署了提出方法的其他部分, 在接收到请求后, 会执行相应的授权工作. 本文实验环境主要包括 3 台物理主机, 其中一台作为管理服务器节点, 另外两台主机分别作为私有云服务器节点和公有云服务器节点.

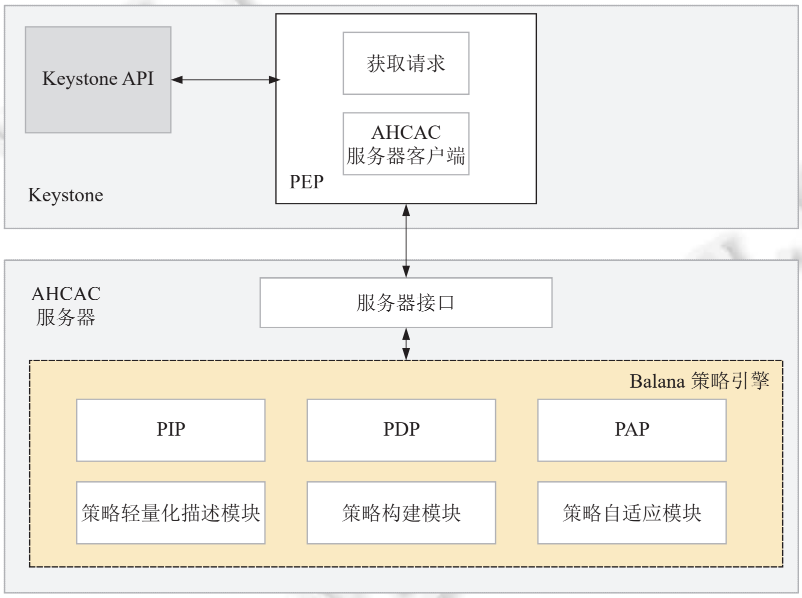


图 8 AHCAC 在 OpenStack 中的部署

本文以某电视台融合媒体混合云系统内部实际部署的访问控制策略为参考选定实验的访问控制策略集, 同时从策略集中提取出广告节目单、节目素材和节目视频 3 类不同安全级别数据资源的策略作为聚类对象. 在电视台融合媒体混合云系统的实际应用中, 广告节目单通常与直播安全紧密相关, 因而安全级别最高; 节目视频通常是指经过精编、技审过的录播视频, 安全级别较高; 节目素材通常是指通过互联网抓取、PGC/UGC 接收、地方台\记者站回传、4G/5G 回传、信号收录等多种方式不同途径采集的媒资数据, 安全级别相对广告节目单和节目视频是

最低的. 如表 6 所示, 这 3 类数据所拥有的生命周期阶段是不同的. 此外, 实验策略集中涉及的电视台主体层次关系如图 4 所示. 这些主体对数据的可能操作主要为创建、查看、添加、修改和删除等.

表 6 数据集

数据资源	策略数量	业务阶段
广告节目单	480	创建、编辑、审核、执行、归档、质控、删除等
节目素材	756	汇聚、编辑、审核、编目、归档、质控、删除等
节目视频	1 417	创建、编辑、审核、备播、转码、发布、归档、质控、删除等

5.2 策略构建层次聚类效果

数据生命周期阶段及各阶段对应的访问控制策略是 AHCAC 方法的重要基础, 只有当层次聚类出的阶段是细粒度合理且同时符合数据所应用的业务场景时, 才能达到 AHCAC 的目标, 因而能否基于策略特点, 正确的将数据生命周期阶段划分出来尤为重要, 而策略间距离计算的准确度是影响聚类效果的关键因素, 因此, 为了验证 AHCAC 中策略构建的正确性, 我们着重通过实验比较了 AHCAC 中基于策略距离的层次聚类算法与传统采用欧氏距离和余弦距离[30]的层次聚类算法在策略构建上的聚类效果, 即比较了生命周期阶段数量的准确性以及不同阶段内策略和阶段间策略的平均相似度. 其中, 欧氏距离和余弦距离是通过将文本与向量建立映射关系来计算文本距离. 由于策略是文本, 策略属性数值化后均可以映射为向量中的元, 因此可以通过这两种方法计算策略间的距离, 策略属性数值化的过程可见我们之前的工作[28].

如图 9 所示, 横轴表示选取的广告节目单、节目素材和节目视频 3 类不同数据资源的策略集, 纵轴表示采用不同距离计算方案层次聚类出的阶段数量. 实验结果表明, 与其他距离计算方案相比, 在策略数量相对较小的情况下, AHCAC 中的策略距离计算方法与欧氏距离聚类出的结果相当, 但当策略数量增加时, 可以看出 AHCAC 中的策略距离计算更为精准, 可以对数据的生命周期阶段进行更有效的划分, 也较符合实际的业务阶段数.

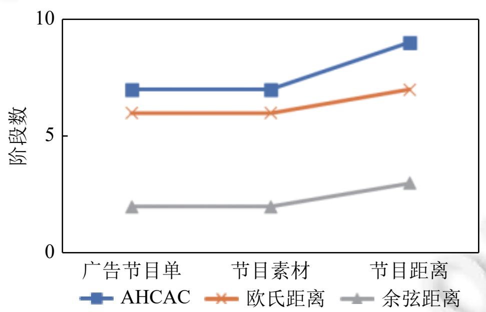


图 9 基于 AHCAC 策略距离、欧氏距离和余弦距离层次聚类的阶段数比较

进一步, 我们通过计算阶段内策略相似度和阶段间策略相似度来对比上述不同距离计算方案层次聚类的精度, 其中, 策略相似度计算延用我们之前工作[28]给出的计算方式. 实验结果如后文图 10 和图 11 所示, 横轴表示不同的策略距离计算方法, 纵轴表示策略相似度. 实验结果表明, 相比欧氏距离、余弦距离相比, 本文提出 AHCAC 中的策略距离计算方法所得出的阶段内策略相似度较高, 阶段间策略相似度较低, 因此, 本文提出的基于策略距离的层次聚类方法精准度更高.

5.3 策略自适应调整效果

下面根据第 5.2 节产出的策略集随机生成访问请求, 在开源策略评估引擎 Balana 上进行实验, 以访问请求处理时间为性能指标, 比较本文提出的根据数据所处阶段自适应调整访问控制策略方法和统一检索访问控制策略方法的策略评估效率. 实验选取节目视频创建、编辑、审核 3 个阶段的策略集作为测试用例, 即在实验过程中数据生命周期状态为这 3 个阶段, 其中创建阶段包含了 100 条策略, 编辑阶段包含 200 条策略、审核阶段包含 300 条

策略. 针对 3 个阶段, 分别随机生成由 200 个访问请求构成的 3 组请求, 每组请求将随机重复发送 50 次. 实验中统计了 Balana 在部署本文提出策略自适应调整方案前后接收请求到反馈出正确响应结果的总处理时间. 访问请求总处理时间的计算过程如公式 (9) 所示:

$$总处理时间 = \text{PEP}接收请求的时间 + \text{PDP}匹配策略时间 + \text{PDP}反馈决策结果时间 \tag{9}$$

为了减小误差, 避免离群数据的干扰, 以上实验步骤重复了 3 次, 最终取 3 次处理时间的平均值为对比数据. 如图 12 所示, 横轴表示创建、编辑和审核 3 个阶段, 纵轴为平均处理时间, 实验结果表明, 在部署本文提出的策略自适应调整方案以后, Balana 对访问请求的平均处理时间大大降低. 原因是策略评估引擎基于 PAP 进行检索评估, PAP 中存储的策略集量级将影响评估效率. 统一检索方法是在 PAP 中存储全部策略集, 并对其采用逐层遍历的方式进行检索, 当数据所处阶段变化时, 需被检索的策略集规模将会增加, 那么对总处理时间自然也将增加. 本文提出方法是在 PAP 中存储当前阶段对应的策略集, 且该策略集提前经过轻量化处理, 故当数据所处阶段发生变化时, 需被检索的策略集规模变化不大, 这加快了策略检索速度, 进而保证能更快响应当前访问请求. 因此, 本文方法可以有效提高策略评估效率.

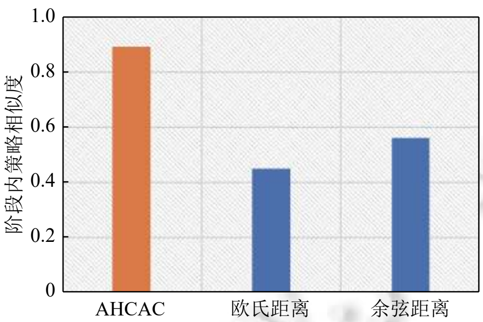


图 10 基于 AHCAC 策略距离、欧氏距离和余弦距离层次聚类的阶段内策略相似度比较

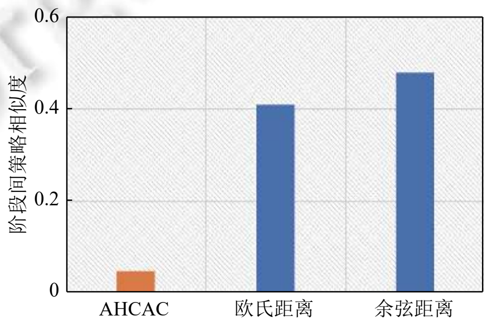


图 11 基于 AHCAC 策略距离、欧氏距离和余弦距离层次聚类的阶段间策略相似度比较

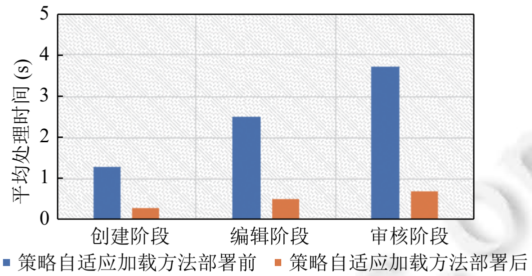


图 12 部署策略自适应调整方法前后的性能比较

5.4 与 RBAC 的时间性能对比

为了测试提出方法 AHCAC 的性能, 我们与现有主流的 RBAC 方法进行比较, 比较两个方法对不同量级用户并发访问的平均处理时间, 其中平均处理时间的计算方法同公式 (9), 同样为了减小误差, 避免离群数据的干扰, 以上实验步骤我们重复了 3 次, 最终的对比数据为 3 次时间的平均值.

本实验以节目视频策略集作为测试数据, 其中创建阶段包含了 500 条策略, 编辑阶段包含 1 000 条策略, 审核阶段包含 2 000 条策略. 实验中进行了 Test1 和 Test2 两组测试: Test1 是节目视频数据生命周期阶段从创建阶段变更为编辑阶段, 此时数据从公有云的存储节点传输至私有云的存储节点; Test2 是节目视频数据生命阶段从编辑阶段变为审核阶段, 此时数据在私有云中两个实例间进行传输. 实验结果如图 13 所示, 横轴为并发访问量, 纵轴为平均处理时间, 容易看出, 由于 AHCAC 基于生命周期阶段优化了策略集, 加快了策略检索速度, 从而降低了策略

评估的复杂度, 所以随着并发访问量的增加, 可以看出, 相比 RBAC 方法, AHCAC 方法的平均处理时间更少, 响应效率更高. 因此, AHCAC 方法更适合解决具有大规模用户的混合云环境下的跨云数据安全访问控制问题.

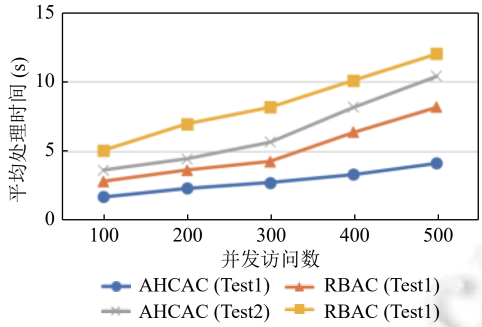


图 13 AHCAC 和 RBAC 针对不同量级访问请求的平均处理时间比较

6 结论与未来工作

本文重点讨论混合云环境下数据在不同生命周期阶段下的细粒度访问控制问题. 我们提出了一种基于数据生命周期的自适应访问控制方法, 首先基于关键属性特征统一描述数据全生命周期的异构策略, 并引入“阶段”属性显式标识数据的生命周期状态. 然后提出基于层次聚类的访问控制策略构建方法, 求解出数据生命周期各阶段对应的访问控制策略. 最后提出面向数据生命周期阶段的访问控制策略自适应调整方法, 通过关键属性匹配实现当数据所处阶段变化时, 自适应加载对应阶段的访问控制策略集. 实验结果证明, 提出方法的策略分类方法准确率较高, 符合数据全生命周期各阶段的细粒度安全控制需求. 同时, 当数据所处阶段动态变化时, 提出方法能自适应调整数据的访问控制策略, 提高了访问控制效率.

当前本文提出方法基于混合云环境下不同云平台提供的异构策略去统一轻量级描述和构建数据在不同生命周期阶段下的细粒度访问控制策略, 如果某一云平台未提供访问控制策略, 则需要主动采集该平台的海量访问控制日志并对其进行策略挖掘, 未来我们将进一步探索高效的策略挖掘方法.

References:

[1] OpenSource Cloud Alliance for Industry, OSCAR. Hybrid cloud white paper (2019). 2019 (in Chinese). http://www.caict.ac.cn/kxyj/qwfb/bps/201907/P020190704511581594525.pdf

[2] Graham GS, Denning PJ. Protection: Principles and practice. In: Proc. of the 1972 Spring Joint Computer Conf. Atlantic: ACM, 1972. 417–429. [doi: 10.1145/1478873.1478928]

[3] Bell DE, La Padula LJ. Secure computer system: Unified exposition and multics interpretation. Technical Report, MTR-2997, Bedford: The MITRE Corporation, 1976.

[4] Sandhu RS, Coyne EJ, Feinstein HL, Youman CE. Role-based access control models. Computer, 1996, 29(2): 38–47. [doi: 10.1109/2.485845]

[5] Sandhu R, Bhamidipati V, Munawer Q. The ARBAC97 model for role-based administration of roles. ACM Trans. on Information and System Security, 1999, 2(1): 105–135. [doi: 10.1145/300830.300839]

[6] Weng CL, Luo Y, Li ML, Lu XD. A BLP-based access control mechanism for the virtual machine system. In: Proc. of the 9th Int’l Conf. for Young Computer Scientists. Hunan: IEEE, 2008. 2278–2282. [doi: 10.1109/ICYCS.2008.503]

[7] Zhu JL, Wen QY. SaaS access control research based on UCON. In: Proc. of the 4th Int’l Conf. on Digital Home. Guangzhou: IEEE, 2012. 331–334. [doi: 10.1109/ICDH.2012.50]

[8] Li FH, Xiong JB. Access Control Technology for Complex Network Environment. Beijing: Posts & Telecom Press, 2019. 106–152 (in Chinese).

[9] Yang HY, Meng LX. Hadoop cloud platform user dynamic access control model. Journal on Communications, 2017, 38(9): 9–17 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2017142]

[10] Chen AG, Xing HW, She K, Duan GD. A dynamic risk-based access control model for cloud computing. In: Proc. of the 2016 IEEE Int’l

Conf. on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom). Atlanta: IEEE, 2016. 579–584. [doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.90]

[11] Wang YB, Wen JH, Zhou W, Luo FJ. A novel dynamic cloud service trust evaluation model in cloud computing. In: Proc. of the 17th IEEE Int'l Conf. on Trust, Security and Privacy in Computing and Communications (TrustCom), the 12th IEEE Int'l Conf. on Big Data Science and Engineering (TrustCom/BigDataSE). New York: IEEE, 2018. 10–15. [doi: 10.1109/TrustCom/BigDataSE.2018.00012]

[12] Fan YD, Wu XP, Shi X. Research on access control model based on the trust value assessment of cloud computing. Netinfo Security, 2016(7): 71–77 (in Chinese with English abstract). [doi: 10.3969/j.issn.1671-1122.2016.07.011]

[13] Zhang K, Pan XZ. Access control model based on trust of users' behavior in cloud computing. Journal of Computer Applications, 2014, 34(4): 1051–1054 (in Chinese with English abstract). [doi: 10.11772/j.issn.1001-9081.2014.04.1051]

[14] Ding HF, Peng CG, Tian YL, Xiang SW. Privacy risk adaptive access control model via evolutionary game. Journal on Communications, 2019, 40(12): 9–20 (in Chinese with English abstract). [doi: 10.11959/j.issn.1000-436x.2019240]

[15] Zhao ZY, Sun L. Attribute-based access control with dynamic trust in a hybrid cloud computing environment. In: Proc. of the 2017 Int'l Conf. on Cryptography, Security and Privacy. Wuhan: ACM, 2017. 112–118. [doi: 10.1145/3058060.3058062]

[16] Luo Y, Shen QN, Wu ZH. A novel access control policy specification language and its permission classification method. Chinese Journal of Computers, 2018, 41(6): 1189–1206 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2018.01189]

[17] OASIS. extensible access control markup language (XACML) version 3.0. 2013. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[18] Ribeiro C, Zúquete A, Ferreira P, Guedes P. SPL: An access control language for security policies and complex constraints. In: Proc. of the 2001 Network and Distributed System Security Symp. San Diego: NDSS, 2001. 89–107.

[19] Wang YD, Yang JH, Xu C, Ling X, Yang Y. Survey on access control technologies for cloud computing. Ruan Jian Xue Bao/Journal of Software, 2015, 26(5): 1129–1150 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4820.htm [doi: 10.13328/j.cnki.jos.004820]

[20] Huang C. The construction of role based access control policy [Ph.D. Thesis]. Hangzhou: Zhejiang University, 2010 (in Chinese with English abstract).

[21] Kuhlmann M, Shohat D, Schimpf G. Role mining-revealing business roles for security administration using data mining technology. In: Proc. of the 8th ACM Symp. on Access Control Models and Technologies. Como: ACM, 2003. 179–186. [doi: 10.1145/775412.775435]

[22] Li NH, Byun JW, Bertino E. A critique of the ANSI standard on role-based access control. IEEE Security & Privacy, 2007, 5(6): 41–49. [doi: 10.1109/MSP.2007.158]

[23] Schlegelmilch J, Steffens U. Role mining with ORCA. In: Proc. of the 10th ACM Symp. on Access Control Models and Technologies. Stockholm: ACM, 2005. 168–176. [doi: 10.1145/1063979.1064008]

[24] Ene A, Horne W, Milosavljevic N, Rao P, Schreiber R, Tarjan RE. Fast exact and heuristic methods for role minimization problems. In: Proc. of the 13th ACM Symp. on Access Control Models and Technologies. Estes Park: ACM, 2008. 1–10. [doi: 10.1145/1377836.1377838]

[25] Choobdar S, Ribeiro P, Silva F. Evolutionary role mining in complex networks by ensemble clustering. In: Proc. of the 2017 Symp. on Applied Computing. Marrakech: ACM, 2017. 1053–1060. [doi: 10.1145/3019612.3019815]

[26] Ni Q, Lobo J, Calo S, Rohatgi P, Bertino E. Automating role-based provisioning by learning from examples. In: Proc. of the 14th ACM Symp. on Access Control Models and Technologies. Stresa: ACM, 2009. 75–84. [doi: 10.1145/1542207.1542222]

[27] Giambiagi P, Nair SK, Brossard D. Abbreviated language for authorization version 1.0. 2015. https://www.oasis-open.org/committees/download.php/55228/alfa-for-xacml-v1.0-wd01.doc

[28] Lin L, Hu J, Mao XY, Zhang JB. Saphena: An approach for analyzing similarity of heterogeneous policies in cloud environment. In: Proc. of the 3rd Int'l Conf. on Cyber Security and Cloud Computing (CSCloud). Beijing: IEEE, 2016. 36–41. [doi: 10.1109/CSCloud.2016.13]

[29] Hubert L, Arabie P. Comparing partitions. Journal of Classification, 1985, 2(1): 193–218. [doi: 10.1007/BF01908075]

[30] Wang J, Xi TW, Zhao KQ. Research on evaluation problem based on improved K-means algorithm. Journal of Chongqing Normal University (Natural Science), 2020, 37(1): 99–106 (in Chinese with English abstract). [doi: 10.11721/cqnuj20200106]

附中文参考文献:

[1] 云计算开源产业联盟. 混合云白皮书(2019年). 2019. http://www.caict.ac.cn/kxyj/qwfb/bps/201907/P020190704511581594525.pdf

[8] 李凤华, 熊金波. 复杂网络环境下访问控制技术. 北京: 人民邮电出版社, 2019. 106–152.

[9] 杨宏宇, 孟令现. Hadoop云平台用户动态访问控制模型. 通信学报, 2017, 38(9): 9–17. [doi: 10.11959/j.issn.1000-436x.2017142]

[12] 范运东, 吴晓平, 石雄. 基于信任值评估的云计算访问控制模型研究. 信息网络安全, 2016(7): 71–77. [doi: 10.3969/j.issn.1671-1122.2016.07.011]

[13] 张凯, 潘晓中. 云计算下基于用户行为信任的访问控制模型. 计算机应用, 2014, 34(4): 1051–1054. [doi: 10.11772/j.issn.1001-9081.2014.04.1051]

[14] 丁红发, 彭长根, 田有亮, 向淑文. 基于演化博弈的隐私风险自适应访问控制模型. 通信学报, 2019, 40(12): 9–20. [doi: 10.11959/j.issn.1000-436x.2019240]

[16] 罗杨, 沈晴霓, 吴中海. 一种新的访问控制策略描述语言及其权限划分方法. 计算机学报, 2018, 41(6): 1189–1206. [doi: 10.11897/SP.J.1016.2018.01189]

[19] 王于丁, 杨家海, 徐聪, 凌晓, 杨洋. 云计算访问控制技术研究综述. 软件学报, 2015, 26(5): 1129–1150. http://www.jos.org.cn/1000-9825/4820.htm [doi: 10.13328/j.cnki.jos.004820]

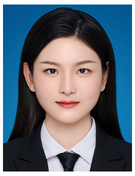
[20] 黄超. 基于角色的访问控制策略构建方法研究 [博士学位论文]. 杭州: 浙江大学, 2010.

[30] 王娟, 席天为, 赵克全. 基于改进K-means算法的评价问题研究. 重庆师范大学学报(自然科学版), 2020, 37(1): 99–106. [doi: 10.11721/cqnuj20200106]

林莉(1979－), 女, 博士, 副教授, CCF 专业会员, 主要研究领域为信息安全, 网络计算.

储振兴(1996－), 男, 硕士生, 主要研究领域为区块链, 访问控制.

毛新雅(1993－), 女, 硕士生, 主要研究领域为混合云安全, 访问控制.



解晓宇(1996－), 女, 硕士生, 主要研究领域为微服务, 访问控制.