The Journal of Systems and Software 107 (2015) 110–126

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A comprehensive modeling framework for role-based access control policies

Ameni Ben Fadhel[a,b,*], Domenico Bianculli[b], Lionel Briand[a,b]

[a] Faculty of Science, Technology and Communication, University of Luxembourg, 6, rue Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
[b] SnT Centre, University of Luxembourg, 4 rue Alphonse Weicker, L-2721, Luxembourg, Luxembourg

ARTICLE INFO

Article history:
Received 28 November 2014
Revised 22 April 2015
Accepted 11 May 2015
Available online 29 May 2015

Keywords:
Role-based access control
Modeling
Survey

ABSTRACT

Prohibiting unauthorized access to critical resources and data has become a major requirement for enterprises; access control (AC) mechanisms manage requests from users to access system resources. One of the most used AC paradigms is role-based access control (RBAC), in which access rights are determined based on the user's role.

Many different types of RBAC policies have been proposed in the literature, each one accompanied by the corresponding extension of the original RBAC model. However, there is no unified framework that can be used to define all these types of policies in a coherent way, using a common model.

In this paper we propose a model-driven engineering approach, based on UML and the Object Constraint Language (OCL), to enable the precise specification and verification of such policies. More specifically, we first present a taxonomy of the various types of RBAC policies proposed in the literature. We also propose the GemRBAC model, a generalized model for RBAC that includes all the entities required to define the classified policies. This model is a conceptual model that can also serve as data model to operationalize data collection and verification. Lastly, we formalize the classified policies as OCL constraints on the GemRBAC model.

1. Introduction

Prohibiting unauthorized access to critical resources and data has become a major requirement for enterprises. Access control (AC) mechanisms manage requests from users to access system resources; the access is granted or denied based on the authorization policies defined within the enterprise. Access control systems can be grouped into three categories (Sandhu and Samarati, 1996): *discretionary* (DAC), *mandatory* (MAC), and *role-based* (RBAC). In DAC, access rights are directly assigned to each user; moreover, a user is the only entity that can control the access to her own object(s), by assigning access rights to other users. In the second category, MAC, the access rights are determined according to mandated regulations stated by a central authority. In RBAC, access rights are determined based on the user's role, e.g., her job or function, as well as on the permissions assigned to each role. By decoupling users from permissions, RBAC simplifies the administration and the deployment of access control policies in large enterprises. In the rest of this paper, we focus on RBAC, since it has become the de facto standard for access control in enterprise systems (Ludwig et al., 2011).

The concept of role-based access control was initially proposed by Sandhu et al. (1996); later on, the various initial proposals of RBAC models were consolidated into a unified standard model for RBAC proposed by the NIST (Sandhu et al., 2000). The basic RBAC model is composed of: 1) entities, corresponding to users, roles, sessions, and permissions; 2) relations among these entities. A user is allowed to execute a set of permissions that corresponds to the role(s) assigned to her; in other words, a role maps each user to a set of permissions. A session maps each user to the set of her active role(s).

RBAC supports three security *principles*: least privilege, data abstraction, and separation of duty. The least privilege principle requires a user to be authorized to execute only the minimal set of permissions needed for a given task, as determined by her role. The data abstraction principle is satisfied by abstracting low-level operations (e.g., the *read* and *write* operations provided by the operating system) with high-level operations defined for each business object in the system (e.g., updating the list of employees). The separation of duty principle states that no user should be given sufficient permissions to misuse the system. Although these principles are supported by RBAC, they are not automatically enforced by a system

* Corresponding author at: SnT Centre, University of Luxembourg, 4 rue Alphonse Weicker, L-2721, Luxembourg, Luxembourg. Tel: +352 46 66 44 5675.
E-mail addresses: ameni.benfadhel@uni.lu (A. Ben Fadhel), domenico.bianculli@uni.lu (D. Bianculli), lionel.briand@uni.lu (L. Briand).

http://dx.doi.org/10.1016/j.jss.2015.05.015
0164-1212/© 2015 Elsevier Inc. All rights reserved.

implementing RBAC: additional authorization constraints, called also *policies*[1], have to be defined to restrict the user's access.

Various types of authorization constraints have been proposed in the literature. For instance, cardinality constraints represent a bound on the number of roles and sessions to which a user can be assigned. Prerequisite constraints are a precondition on user-role assignment, stating that a user can be assigned to a role only if the user is already a member of another role. Separation of duty constraints (SoD) define a mutual exclusion relation among roles, permissions, or users. Dually, binding of duty (BoD) constraints define a correlation among a set of operations that must be performed by the same user. Delegation constraints allow a user to temporarily transfer a set of permissions associated to her role to another user. Context constraints restrict a user from performing an action depending on her current location or on the time at which the action should happen.

Various extensions of the original RBAC96 model have been proposed to support these different types of constraints. However, there is no unified framework that can be used to define all these types of authorization constraints in a coherent way, using a common model. The lack of a unified framework makes difficult for practitioners to understand, select among, and implement the different types of policies proposed in the literature.

In this paper we survey and classify the various types of RBAC authorization constraints proposed in the literature, and describe the different facets that characterize each type of constraint. We also review the different extensions of the original RBAC model that have been proposed in the literature to support the various types of constraints. The main result of this review is that none of the proposed models can support all the constraints included in our classification. To address this limitation, we propose the GemRBAC model, a *Generalized Model for RBAC* that includes all the conceptual entities required to define the classified constraints. We then specify in an unambiguous manner all types of constraints to enable their operationalization. The specification follows a model-driven approach, based on UML and the Object Constraint Language (OCL): the classified constraints are formalized as constraints expressed with OCL on the GemRBAC model. This formalization brings three benefits: 1) it enables practitioners to select and make use of the various policies in a precise manner, based on the GemRBAC model; 2) it lays the ground for the practical verification of such policies, both at design time and at run time, based on UML modeling tools and OCL checkers (such as Eclipse OCL (Eclipse, 2014)); 3) it shows the expressiveness of the GemRBAC model, since it can accommodate all types of constraints included in our classification.

More specifically, the main contributions of this paper are: 1) a taxonomy, classifying the main RBAC policies proposed in the literature; 2) the GemRBAC model, which is a generalized model for RBAC that includes all the entities required to define the policies classified in the taxonomy; 3) the formalization, as OCL constraints on the GemRBAC model, of the RBAC policies included in the taxonomy; these constraints have been made publicly available, together with an Ecore (Eclipse, 2013) version of the GemRBAC model, at https://github.com/AmeniBF/GemRBAC-model.

The rest of the paper is organized as follows. Section 2 discusses the motivations of this work. Section 3 describes the original RBAC conceptual model. Section 4 presents a taxonomy of the various types of RBAC constraints proposed in the literature. Section 5 illustrates the various extensions to the original RBAC model. Section 6 introduces the GemRBAC model. Section 7 presents the specification of RBAC policies using OCL constraints defined on the GemRBAC model. Section 8 discusses, with an example, the application of the proposed model for the verification of RBAC policies. Section 9 discusses the related work while Section 10 concludes the paper and provides directions for future work.

[1] In the rest of this paper, we will use the terms "(authorization) constraints" and "policies" interchangeably.

2. Motivations

RBAC is an access control mechanism that defines rules for authorizations and access restrictions for each role within an organization. Such a policy is required to specify access rights according to an individual's job or function (i.e., her role); unlike traditional access control, rights are not assigned to a user according to her identity. RBAC is available in some, security-oriented variants of Unix-like operating systems, as well as in modern database management systems. These systems implement a subset of the NIST RBAC model (Sandhu et al., 2000), based on the initial proposal of Sandhu and Samarati (1996).

As we will see in the next sections, the original RBAC model supports a limited number of different types of authorization constraints, which cannot fulfill the expressiveness requirements that have emerged in the recent years in modern organizations. Examples of these new requirements are supporting delegation and revocation of permissions, and enabling access control policies based on the spatio-temporal context of users.

To fill this gap, researchers have proposed several extensions of the original RBAC model, to support the definition of new types of constraints (see Sections 4 and 5). Though this work opens new possibilities for applying RBAC in modern enterprise systems, it is not easy to exploit in its current form. Indeed, these types of constraints and their corresponding models are scattered across multiple sources, are defined using different formalisms, and sometimes the concepts are expressed in an ambiguous manner.

This situation is very impractical for practitioners who want to select the relevant types of policies to be implemented in their systems. Moreover, they are faced with several models, often partially overlapping with each other, but with slightly different semantic variations. Furthermore, to the best of our knowledge, there is no model that can express *all* the type of constraints that we have identified in our survey. Last, scattered and heterogenous models make it difficult for researchers to understand the state of the art in a coherent manner.

We contend there is clearly a need for organizing the various types of RBAC authorization constraints systematically, based on a unified framework. The goal would be to formalize these constraints in such a way as to enable and facilitate their operationalization. This is the reason for which we propose the GemRBAC model, as a unified RBAC model that captures *all* the types of constraints found in the literature. Furthermore, for each type of authorization constraint, we define its formalization using OCL. By using such a common, standardized, and well-supported language, we not only facilitate the precise understanding of such constraints but we also facilitate their operationalization through industry-strength tools.

3. The original RBAC conceptual model

The original RBAC conceptual model, proposed in 1996 by Sandhu et al. (1996), is composed of *users, roles, sessions*, and *permissions*; Fig. 1 illustrates the different components of this model and the relations between them. According to Sandhu et al., a role can be seen, at the same time, both as a collection of permissions and as a collection of users. A role can be assigned to one or more users via a *user-role assignment* relation. A *role-permission assignment* relation maps each role to one or more permissions. A session is a mapping of one user to a subset of the roles that have been assigned to her; this mapping *activates* the role(s) for a certain user. A permission allows a user to perform some operation(s) on some resource(s) of the system.

A role can be inherited using a *role hierarchy* relation, as shown in Fig. 2. A role can have one or more juniors (sub-roles) denoted by an arrow. For instance, r_2, r_3 and r_4 are juniors of r_1. In addition to its assigned permissions, r_2 inherits all permissions from its ancestor r_1.