A Role and Activity Based Access Control Model for University Identity and Access Management System

Shuliang Zheng, Dongxing Jiang, Qixin Liu
Computer and Information Management Center of Tsinghua University
zhengsl03@gmail.com, jdx@cic.tsinghua.edu.cn, lqx@cic.tsinghua.edu.cn

Abstract

Although RBAC model has received broad support as a generalized access control solution, it has several innate limitations. We propose a Role and Activity Based Access Control model called R-ABAC, which extends the traditional RBAC model with the notion of participation, act and activity. An activity is an abstraction of the application environment, and organizes participations and acts as a digraph for expressing dependency of activity steps, as well as containing rich context information such as time, location and system status. An implementation of R-ABAC model has been applied to a university unified identity and access management system (UIAMS).

1. Introduction

Access control is an important technology for limiting users to access resources, and preventing the users from illegal access. Role-Based Access Control (RBAC) model [1] is an attractive security administration solution in many information systems.

University Identity and Access Management System, called UIAMS, is a unified identity administration, authentication and authorization system. UIAMS is specially used in university information environment with collaborative, distributed and heterogeneous features. RBAC is not suitable for UIAMS. Firstly, the increasingly expansion of role scale leads to heavier administration burden and slower access response. Secondly, more dynamic control requirements have been raised, and it is hard for RBAC to process any rich environment information. Thirdly, the coexistence of many heterogeneous systems makes it hard to implement a unified administration and authorization system via RBAC.

In order to overcome limitations of RBAC and solve these problems in UIAMS, we propose a Role and Activity Based Access Control (R-ABAC) model. R-ABAC extends the RBAC model with the notion of participation, act and activity. Participation is used to express the idea of “functional role” and is associated with act which is used to express operation or function. Activity organizes participation and act as a digraph for expressing dependency of activity steps. A user who sends an access request is assigned roles he or she owns, and then a proper activity context is loaded on the request. Next the user is granted participation via the mapping rules of role and participation. Finally, a judging result (“permit” or “deny”) is given through the context information.

The rest of this paper is organized as follows. Section 2 gives an overview of some related works and introduces the basic idea of R-ABAC design. In section 3, we formally present the core R-ABAC model. In section 4, an implement of R-ABAC model in UIAMS is introduced. Finally, we conclude the paper.

2. Related works and design thought

In the decades of the access control model research, various models are proposed for different requirements. Through a careful comparison and analysis, three research hotspots are found as follows:

First hotspot is to research how to abstract and model real world activities in a proper way. Many new application requirements need access control model not only to be able to judge whether a user has permissions to read or write some resources, but also to handle complex information activities, such as collaborative activities. So some access control models are built to reveal the real situation of specific domain, and a well abstracted model could be developed to a standard of its domain. Reference Information Model (RIM) of HL7 [2] is a successful example.

Second hotspot is to research how to improve the flexibility and scalability of dynamic authorization. For this purpose, just focusing on traditional access control model elements, such as subject, role, permission and object is not enough, and more expressive elements, such as policy and context are needed. Defining unified
policy format and policy control pattern is a big challenge, and XACML [3] is a successful instance of this work. On the other hand, context has been getting more and more attention. Many novel model elements have been proposed to describe the application context of the specific domain, such as team in TMAC [4], location in LBAC [5], trust in TCAC [6], etc.

Third hotspot is to research how to administrate identity and authorization conveniently and effectively. In relatively large organization, the usage of role shows considerable advantages. [7] demonstrates the necessity of role in most application environment with highly convenient administration requirement. Many novel access control models are built on the notion of role, such as T-RBAC [8], TCAC, etc. Based on a comprehensive consideration of the three spots above, R-ABAC model is designed to retain role as a core element, to introduce participation and act as main components of activity which can be seen as the access control context of R-ABAC, to model most university application system activities for UIAMS. So R-ABAC is conveniently administrative, dynamic, flexible and quite suitable for UIAMS.

3. R-ABAC model

R-ABAC integrates the elements of participation, act and activity into the traditional RBAC. Components of core R-ABAC are illustrated in Figure 1. We denote the set of users, roles, participations, acts, activity cells, control policies and role-participation maps with \(U, R, P, A, AC, CP, RP\). Activity cell is the basic cell of activity, that is, \(AC\) is the node of activity digraph. \(CP\) is an important component of R-ABAC for its function of providing judging rules to access control requests.

![Figure 1. R-ABAC model](Image)

Besides these basic components, R-ABAC also has a series of assignment components as follows:

- \(URA \subseteq U \times R\): many-to-many mapping relations between user and roles.
- \(RH \subseteq R \times R\): role hierarchy.
- \(RPA \subseteq R \times P\): many-to-many mapping relations between roles and participations, decided by \(RP\).
- \(PPA \subseteq P \times A\): one-to-one indirect relationships between participation and act, decided by \(ACPA\) and \(ACAA\) jointly.
- \(ACP \subseteq AC \times P\): one-to-one mapping relations between activity cells and participations.
- \(ACAA \subseteq AC \times A\): one-to-one mapping relations between activity cells and acts. \(ACAA\) and \(ACPA\) decide \(PPA\) jointly and indicate that one activity cell consists of one participation and one act.
- \(CPAC \subseteq CP \times AC\): one-to-many relations from control policy to activity cells.

3.1. Participation and act

Participation and act are not brand new concepts, already proposed in RIM of HL7. Not so complicated as the ones defined in RIM, participation and act in R-ABAC have simpler meanings. Generally speaking, act denotes the operation of application systems and participation denotes a qualification as the performer of related act. One act usually has several kinds of participation, and a participation also can perform more than one act. So the relationship between participation and act is conceptually many-to-many. However, in fact, participation and act do not have direct associations in R-ABAC, and activity cell is used to associate a participation instance with an act instance. An act instance contains not only the basic information of operation, but also some other constraint information, such as effective time, security location, etc. A participation instance also contains its effective time as well as basic qualification information.

Role and participation in R-ABAC have essential differences. Role is the relatively static and permanent identity of users, and participation denotes the highly dynamic identity which could be obtained by users when he or she is participating in an act. More precisely, a user has a chance to obtain a participation only if he or she has sent an access control request and has been associated with an activity cell, but the user owns roles in nature whether he or she send request or not. The typical participations in university information environment are teacher in class, student in class, assistant, accountant, etc, and the most typical roles in university are teacher, student, leader, servant, etc.

The component of participation helps to avoid the administration complexity and performance cost brought by the increasingly expansion of user scale and application amount. Participation, which is restricted by application systems, is independent of role, which is restricted by organization and users. So the change of organization and users do not affect participation, and the change of application systems do not affect role. If the cost brought by the former is \(c1\), and the cost
brought by the latter is c2, the total cost in R-ABAC is approximately c1+c2, but in RBAC the cost will be at the level of c1×c2. Moreover, in university information environment, many application systems have private permission control policies which are hard to be managed uniformly, and we can use R-ABAC to solve this problem naturally because of the decoupling attribute of role and participation. Application system can manage local participation component which is served as a remote component of UIAMS.

**Definition 1.** Let PT be the set of participation types. Let participation instance set \( P = \{ p_1, p_2, \ldots, p_n \} \), where \( p_i = \langle \text{type}, d, r, a_c, \text{sysId} \rangle \), \( p_i \in PT \). Besides, \( d \) is the effective duration of participation instance.

**Definition 2.** Let \( AT \) be the set of act types. Let act instance set \( A = \{ a_1, a_2, \ldots, a_n \} \), where \( a_i = \langle \text{type}, \text{sysId}, d, \text{location}, a_t, \text{type} \in AT \rangle \). And \( \text{sysId} \) is the numeric identity of the system on which the act is performed; \( d \) is the effective duration of \( a_i \); \( \text{location} \) is the security domain in which the act can be performed.

### 3.2. Activity

An activity is a digraph of activity cell instances. One activity cell instance denotes a step of activity, and the digraph describes the dependency between steps. In some circumstances, a step is not independent and relies on some other steps. If those steps are not all completed, this step will not be executed. As a result, dependency of steps can influence authorization and be treated as very important context information. The idea of activity derives from the consideration of real world activities and workflow system. Unlike some models which need rely on workflow system to execute authorization, R-ABAC designs activity as a much more light weight workflow management component. This design is quite suitable for university application environment, and brings simplicity and high efficiency.

**Definition 3.** Let activity cell set \( AC = \{ ac_1, ac_2, \ldots, ac_n \} \), where \( ac_i = \langle a_c, a, d, cp, isdep, iscompleted > \rangle \). Besides, \( isdep \in \{ \text{true}, \text{false} \} \) is a flag to show whether \( ac_i \) is dependent on others or not; \( iscompleted \in \{ \text{true}, \text{false} \} \) is a flag to show whether current act has been completed or not; \( d \) is the effective duration of activity cell. \( cp \) is the element of CP defined in Definition 7.

**Definition 4.** Let activity set \( AS = \{ a_1, a_2, \ldots, a_n \} \), where \( a_i = \langle \text{name}, G, d \rangle \), \( a_i \neq a_j \), \( i \neq j \), \( 1 \leq i, j \leq n \), that is activity name is unique in AS. Besides, \( G \) is the activity digraph of \( a_i \); \( d \) is the effective duration of \( a_i \).

### 3.3. Control policy

Control policy is not the core but a very important component of R-ABAC. Control policy is a kind of constraint component with configurable and runtime pluggable feathers, which is used to add dynamic constraints on activity cell. This component contains four elements: target, policy, rule and constraint. Target indicates the activity cell type which policy will react on. Each policy has a target and a set of rules with the same target. Each rule consists of several constraints. A constraint is an atomic expression that is used to restrict one attribute of activity cell.

**Definition 5.** Let target set \( T = \{ t_1, t_2, \ldots, t_n \} \), where \( t_i = \langle t, \text{clist} > \), \( t_i \in PT \), \( t_c \in AT \).

**Definition 6.** Let rule set \( R = \{ r_1, r_2, \ldots, r_n \} \), where \( r_i = \langle r, att > \), \( r_i, t_i \in T \), and \( r_i \text{clist} \) is a list of constraints. A constraint is defined as \( cons = \langle att, cop, evvalue > \), and \( att \) is the attribute of activity cell that constraint reacts on; \( cop \) is the constraint operation such as equal, not equal, greater, less, in, not in, etc; \( evvalue \) is the value argument of this expression.

**Definition 7.** Let policy set \( CP = \{ p_1, p_2, \ldots, p_n \} \), where \( p_i = \langle t, rs > \), \( p_i, t_i \in T \), \( p_i.rs \subseteq R \), and for any \( r \) in \( p_i.rs \), \( r.t = p_i.t \).

### 3.4. Access control

An access request is \( <s, \omega> \in SES \times A \), which means the user in session \( s \) wants to perform the act \( a \). The procedure of handling request in R-ABAC is:

1. Step 1. Find the proper activity \( A \) corresponding to \( a \). Test if \( a.d \) is in \( A.d \). If true, load \( A \) as context for this request and go to step 2. If false, stop and return “deny” for the request.

2. Step 2. Find the activity cell \( ac \) corresponding to \( a \) from \( A \). Test if this expression is true: \( ac.isdependent = \text{false or (ac.isdependent = true and all the cell that ac relies on is completed)} \). If true, go to step 3. If false, stop and return “deny”.

3. Step 3. Get the role information \( r \) of the user from \( s \). Test if \( ac.p \) can be assigned to \( r \) via RPA set. If true, go to step 4. If false, stop and return “deny”.

4. Step 4. Compare the attributes of \( ac \) from the request to the constraints of \( ac.cp \). If all the attribute values satisfy the constraints of \( ac.cp \), stop and return “permit” for the request, or return “deny”.

### 4. Implementation in UIAMS

R-ABAC is designed for various requirements of UIAMS, and suitable for other similar application environment. R-ABAC has been implemented as a core service component of UIAMS. The layered architecture of R-ABAC model system is illustrated in Figure 2.
Figure 2. Architecture of R-ABAC system

Data Base uses RDB to persist the basic data of model, including the data of user, role, participation, act and control policy.

Domain Layer contains several entities which are mapping objects of the tables in Data Base. These entities are implemented as EJB 3.0 entity beans and have one-to-one relation with the tables in Data Base.

EAO Layer, which is entity access object layer, provides interfaces and implementations of accessing entity beans of Domain Layer for upper layer.

Logic Layer contains the implementations of core business logic and algorithm of the basic components of R-ABAC system, such as access control, user, role, participation, act, activity, policy, etc.

Service Layer is the top layer of R-ABAC system and provides several core service components for UIAMS, including access control service, User and Role management service, activity management service and policy management service.

Common Utilities is a common utility set for all layers, including cache manager, xml parser, constraint expression parser, R-ABAC exceptions, etc.

6. Conclusion and future work

In this paper, we propose the R-ABAC model for UIAMS. R-ABAC extends the RBAC by introducing the notion of participation, act and activity. Access control bases on not only session information of user, but also rich context information built by activity. The administration is much more convenient because of the independency of role and participation, especially in a large organization. Dynamic constraint function provided by control policy component makes R-ABAC more flexible and scalable. And R-ABAC can naturally do the multi-grained access control, because it uses the instance of participation and act as basic control unit.

A lot of work remains to be done. We plan to express R-ABAC with the standard access control language such as XACML. Besides, many details of the model need to be systematized and formalized, including role inheritance, duty separation, delegation, etc, rather than just considered in implementation. And improving the efficiency of access control and management should be a constant work.

References


