Computers in Industry 65 (2014) 116–123

Contents lists available at ScienceDirect

Computers in Industry

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

Association-Based Active Access Control models with balanced scalability and flexibility

Zhai Zhi-nian[a,*], Lu Ya-hui[b], Zhang Ping-Jian[c], Chen Zhi-hao[d]

[a] School of Information and Electronic Engineering, Zhejiang University of Science and Technology, Hangzhou, China
[b] School of Computer and Software, Shenzhen University, Shenzhen, China
[c] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[d] Center for Software Engineering, University of Southern California, Los Angeles, United States

ARTICLE INFO

Article history:
Received 6 January 2012
Received in revised form 15 April 2013
Accepted 31 July 2013
Available online 17 September 2013

Keywords:
Access control
Workflow
Task
Role
Scalability
Flexibility

ABSTRACT

In existing Active Access Control (AAC) models, the scalability and flexibility of security policy specification should be well balanced, especially: (1) authorizations to plenty of tasks should be simplified; (2) team workflows should be enabled; (3) fine-grained constraints should be enforced. To address this issue, a family of Association-Based Active Access Control (ABAAC) models is proposed. In the minimal model $ABAAC_0$, users are assigned to roles while permissions are assigned to task-role associations. In a workflow case, to execute such an association some users assigned to its component role will be allocated. The association's assigned permissions can be performed by them during the task is running in the case. In $ABAAC_1$, a generalized association is employed to extract common authorizations from multiple associations. In $ABAAC_2$, a fine-grained separation of duty (*SoD*) is enforced among associations. In the maximal model $ABAAC_3$, all these features are integrated, and similar constraints can be specified more concisely. Using a software workflow, case validation is performed. Comparison with a representative association based AAC model and the most scalable AAC model so far indicates that: (1) enough scalability is achieved; (2) without decomposition of a task, different permissions can be authorized to multiple roles in it; (3) separation of more fine-grained duties than roles and tasks can be enforced.

1. Introduction

Business Process Management (BPM) is an important branch of Computer Supported Collaborative Work (CSCW). In a BPM system for some business goal there are complicated divisions of labor, thus a flexible and user-transparent protection should be performed in its access control [1]. On the other side, such a system is subject to enterprise-level application, and there are plenty of subjects, objects, permissions and constraints to administrate in a scalable and maintainable way. For corresponding access control models, both the flexibility of business collaboration and the scalability of security administration are definitely necessary.

In the 1990s, to improve the adaptability to business process in access control, Thomas and Atluri et al. proposed two critical features in their respective models: (1) to authorize permissions according to the requirements of a task [2]; (2) to synchronize the authorization-flow with the workflow [3]. That is to say, the permissions for a task would not be granted until the task is started, and once it is finished the granted permissions should be revoked. Knorr proved that by this way the data misuse possibilities will be reduced [4] and then the Principle of Least Privilege (PoLP) [5,6] will be better enforced. By these work a paradigm of Active Access Control (AAC) is initiated which is flexible in business context adaption. However there might be millions of tasks in large-scale business environment, it is a huge burden to assign each task with its candidate executors and required permissions.

In 2003, Oh and Park proposed Task-Role-Based Access Control (T-RBAC) [7], introducing the scalability of RBAC [8,9] into AAC. It comes with a 3-step (permission-task-role-user) authorization mechanism and Supervision-Role Hierarchy (S-RH). With the intermediary of a role, m^*n relationships between tasks and its candidate executors could be reduced into m task-role and n user-role relationships. With task inheritance in S-RH, the back-and-forth of task allocations between managers and their subordinates could be reduced. In most other 3-step models, such as literature [10,11], S-RH is replaced with the role hierarchy of RBAC to reduce task allocations between general superior–inferior roles. However, for permission assignment, there lack effective means in these

* Corresponding author. Tel.: +86 13456748762.
E-mail address: zhaizhinian@gmail.com (Z.-n. Zhai).

0166-3615/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.compind.2013.07.013

models. In a business process the required permissions of tasks are quite likely to intersect each other, and then repetitive authorizations will be raised. During the phase of maintenance, these authorizations must be repetitively modified or deleted too. Therefore the administrative burden of the 3-step mechanism is still huge. In 2011, Zhai and Lu proposed Scalability Enhanced Active-Passive-Integrated Access Control (SEA-PIAC) [12] model in which the task specialization, not the task hierarchy, is used to extract repetitive authorizations among tasks. Their approach is quite effective since the specialization relationship is extremely widespread and any authorization to a task will definitely be inherited by all its specialized tasks. By comparison, the task hierarchy is quite awkward in that. First, little authorization extracting power is provided with the concept of sub-task. For example in a software project [12], there are many programming tasks such as framework, component-1, ..., component-k programming. Each requires read-plan, read-requirement, read-design, read-code, execute-makefile and write-deploy permissions and then repetitions exist. Because of the diversity of these permissions, it is difficult to separate most of them with a few coarse-grained sub-tasks. If they are separated with many tiny sub-tasks such as plan reading, requirement reading, design reading, code reading and building, the repetitive permissions will just be replaced by almost as many repetitive sub-tasks. Obviously the programming tasks with their authorizations should be generalized, rather than decomposed. What about a super-task then? It is not quite suitable either, which will be analyzed in the next paragraph. Although in the enforced 3-step mechanism both the task allocation and permissions assignment can be well simplified, the collaboration of multiple roles with different permissions in a task, i.e. a team-task, is not supported. This feature has been introduced in the field of BPM [13] as far back as 2001. It is very elementary to improve the flexibility of team collaboration in workflows [14]. Besides, it is non-substitutable since some task with multiple roles, e.g. a conference, cannot be decomposed into multiple tasks each of which is executed by a single role. Actually a team-task is impossible to implement under the 3-step framework, because a task (with all its permissions) must be assigned to a role in a manner of All-or-Nothing. The fact that a task is assigned to multiple roles just means that each role is qualified to execute the whole task.

As another modeling approach, permissions are authorized to some association of task and role. Since a task may be associated to multiple roles and these associations may further be assigned different permissions, a team-task will be naturally enabled. In Atluri's model [15] subsequent to [3], a task may have multiple Authorization Templates (AT) each of which has a different role as its subject component. However, to assign k different permissions to a role in a task, k ATs (each of which associates a permission to the role) must be related to the task, and then $2k$ relationships must be specified manually. Not all these costs are necessary. In 2002, Wu and Sheth proposed a workflow authorization model [16] in which permissions are assigned to role-task pairs. To assign k permissions to a role in a task, only $k+1$ relationships, i.e. a role-task pair and k permission assignments to it, should be specified manually. To make some simplification, two implicit authorization rules are suggested in it, i.e. a permission assigned to a role r in performing a task t will propagate to: (1) all the roles which are superior to r; (2) all the sub-tasks which are included in t. However, a workflow has multiple cases to be processed and in any case which user to perform a task should be specified further. Before this resource allocation step in terms of workflow, a user u whose role is r cannot use a permission p in any instance of a task t even some role r' inferior to r is assigned p in t. In fact, once a user of r' or r is allocated to execute an instance of t as role r' only he will be authorized to use p in this instance and no other user will be authorized this permission by inheritance. Nevertheless, the rule (1) is useful in specifying candidate executors for a task before the instance-level resource allocation. As for the rule (2), an authorization inheritance mechanism does be given, but its applicable scope is quite limited. A real task might be decomposed in a unpredictable way, but this rule relies on too fixed a situation. For example, a system analysis task sa is undertaken by a role analyst a with a permission write–requirement (w–r). Along with the analysis progress, k sub-systems are identified and the requirement r is divided into k parts $r_1, r_2, \ldots, r_k$. Accordingly the task sa is decomposed into k sub-tasks $sa_1, sa_2, \ldots, sa_k$ where sa_i ($i = 1, \ldots, k$) should be undertaken by a with a permission w–r_i only. However, according to the rule (2), a is still permitted to write the whole r in sa_i. As another possibility, sa might be decomposed into three sub-tasks: investigation iv, draft df and review rw. If r is considered defective in rw performed by the project manager, it must be revised in df and once again reviewed. Obviously r could not be write by a during the process of rw. However, according to the rule (2), a is still permitted to write into r under review. Since a composing relationship is somewhat loose compared to a specialization relationship with rigorous logic, different tasks with diversiform authorizations may be assembled to achieve the goal of their super-task. It is difficult to deduce that the authorizations of a role (or a user) in a task will propagate to all its sub-tasks. Hereto the leftover issue in the last paragraph is also interpreted. In 2008, Qiu and Ma proposed a model [17] in which a task may have multiple authorization policies each of which associates some permissions and their authorized roles. As a result, in the same task n authorizations to m roles can be reduced to n permission-policy and m role-policy relationships. However, repetitive authorizations among different tasks cannot be handled. In 2009, Cao and Chen et al. proposed a policy-based authorization model for team-enabled workflows [14]. However, it does not concern the permission assignment since its main focus is the organization dimension of workflows. Furthermore in the above team-task enabled authorization models, the corresponding separation of duty (SoD) [5,18] are not specified. In fact, there is a composite duty to be constrained, which is determined by a task together with one of its participant roles. The most elementary requirement is that in the same workflow case once a user has executed a task t as a role r, he will be prevented to execute a task t' as a role r'. In the literature [15], constraints are specified based on potential authorizations each of which is a (user, object, privilege) tuple subject to a task in a case of the workflow. Since in a potential authorization the role of the user is not identified, the semantics of such separation of composite duties cannot be expressed. In the literature [14], a constraint policy will be specified using its SQL-like syntax. However, in its structure of "reject <resource> for <activity>", a user cannot be prohibited to execute a task as a specific role, and then the separation of composite duties cannot be specified.

In summary, the scalability and flexibility of security policy specification have not been well balanced in existing AAC models. To address this issue, a model family of Association-Based Active Access Control (ABAAC) is proposed in this paper. With task-role associations as the basis of authorization, different permissions can be authorized to multiple roles in the same task. By identifying exclusive associations, fine-grained SoD constraints can be enforced. A concise inheritance mechanism, the isa relationship on task-role associations, can be deduced based on the associations, the task hierarchy and role specialization relationships. By this mechanism, repetitive authorizations and constraints can be reduced effectively.

The rest of this paper is organized as follow. In Section 2, the ABAAC model family is formally defined. In Section 3, the main