An Object-Dependent and Context Constraints-Aware Access Control Approach Based on RBAC

Xiaoli Ren, Lu Liu and Chenggong Lv

School of Economics & Management, Beihang University, Beijing 100083, P.R. China
rxl@sem.buaa.edu.cn liulu@buaa.edu.cn lcgong@gmail.com

Abstract. The universal adoption of the Internet requires a fine grained access control in the sharing of sensitive resources. However, existing access control mechanisms are inflexible and do not help in alleviating the management task of administrating users' access to resources based on security policies. In this paper, we propose an approach to implement fine-grained access control based on RBAC while considering specific context constraints. The approach is object-dependent and policy-enforced through binding policies to particular object. In the policies, context constraints are incorporated to support separation of duties (SoD). Furthermore, the implement of the approach is described in detail and an application to meet specific access control requirements of comprehensive knowledge management system in an aviation enterprise is presented.

Keywords: *Role-based access control, Security policy, Context constraints, SoD, Object-dependent*

1. INTRODUCTION

The evolution of software and hardware technologies for sharing resources is progressing at a high pace. This poses high demands on access control services that are deployed in interconnected and interactive environments. Users of a system must be authenticated to be legitimate users, and must only be permitted to retrieve and modify data in the ways that are authorized by an access control policy specification. In the first place, fine-grained access control is required and in particular, such services often need to consider the context constraints to enforce fine-grained access control policies, which rely on context information such as the particular sensitive resource's attributes and run-time information, such as time, location, process state, access history and so on. Therefore permissions and permission assignment often depend on context constraints.

Role-based access control (RBAC) has emerged as a widely accepted alternative to classical discretionary and mandatory access controls. Several models of RBAC have been published and several commercial implementations are available [1]. As a result, explicit safety specification is often supported. For example, RBAC models include constraints for safety expression, and there has been a significant amount of work on constraint expression in access control models [2]. In work related to ours, Adam et al. introduced a sophisticated authorization model that was specifically designed to

meet access control requirements of digital libraries [3]. In particular their model allows for the consideration of additional user and object attributes aside from unique identifiers. Role templates as proposed by Giuri and Iglio can be used to consider certain types of context information when defining roles and permissions.

One possible way to deal with dynamically changing context is to modify permissions assignment according to the changes in the environment. Another way is to define conditional permissions that consider certain context conditions in access control decisions, and thus permissions are context-aware to a certain degree [4]. In either way it is sensible to adapt existing access control models and technologies. Thus we think that an access control mechanism supporting object-dependent access control with context constraints should be based on well known models and techniques, and should offer a path from "traditional" to context constraint-aware access control policies.

In order to support fine-grained access control, we propose an object-dependent approach to focus on a particular object. In our approach, the object can be a document, a user, or anything particular that need to be managed. Through a process of so called "binding policies" to the object, we could achieve the goal to assure the particular resource security.

Moreover, considering the dynamic attributes of object and interconnected and interactive environments, a central idea is to support constraints on almost all parts of an RBAC model (e.g. permissions, roles, and assignment relations) to achieve much higher flexibility. Our approach support both static and dynamic separation of duties to allow the complicated properties of object and interconnected and interactive environments, so that the complexity of security management can be largely eased [5].

The rest of this paper is organized as follows. We first present preliminary definitions and then introduce the model which focuses on particular object and the characteristics of the model. Second, we propose the mechanism for implementing object-dependent Access Control through binding polices to objects and considering complex context constraints. Third, the process of binding policy to object and considering constraints are discussed in more detail. Finally, an application to meet specific access control requirements of comprehensive knowledge management system in an aviation enterprise is presented to prove the effectiveness.

2. PRELIMINARY NOTIONS, MODE AND ENGINEERING PROCESS

2.l Preliminary Notions

In this section, we briefly describe some essential concepts that are relevant in our approach and the relationships between these entities.

User: user refers to people who interface with the computer system.

Role: embody a collection of permissions within an organizational setup.

Subject: computer process acting on behalf of a user is referred to as a subject.

Object: an object can be any resource accessible on a computer system, including files, peripherals such as printers, databases, and fine-grained entities such as individual fields in database records [1].

Operation: operation is an active process invoked by a subject.

Context constraint specifies that certain context attributes must meet certain conditions in order to permit a specific operation, i.e. actual values of one or more contextual attributes for predefined conditions.

Policy: policy refers to some combination of role, operation and constraint and specifies role can do what operations under what constraints.

Rule: each policy can be reduced to a set of such triple [role, operation, constraints] which is referred as a rule.

Bind: the process of assigning policies to a particular object.

Roles are granted to users and policies are bind to objects. The relationship is illustrated in Fig. 9.

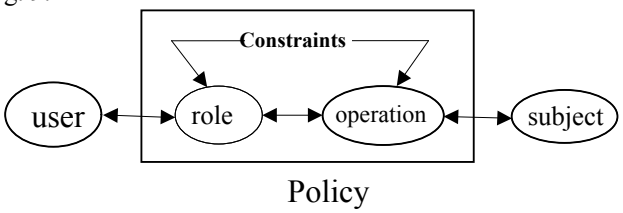

Figure 1. Relationship Between the Entities

2.2 Model

Properties of core RBAC ensure that user is authorized by comparing user's roles with authorized roles in the performance of object access check.

Property 2.2.1 Role authorization: A role R can be allow to perform an operation OP on an object O under context constraint Con only if there exits identical rule [R, OP, Con] which is reduced from the policies that are banded to O.

Property 2.2.2 User authorization: A User U can perform an operation OP on object O only if U has authorized role R.

Property 2.2.3 Operation checked: When an operation is performed, it must be checked to see if it is permitted.

2.3 Description of the Engineering Process

To attain the most fine-grained and flexible access control, we focus on each particular sensitive object that needs to be managed. Because different policies are required by different objects, we assign policies for each object through a process of binding policy to object.

When a subject attempts to perform an operation (e.g., read or write) on an object, the validate monitor (VM) must perform a check, comparing the attributes of the subject with that of the object.

In addition, the VM, with respect to some security policy, must control the specific checks that are made and all modifications to the context constraint and policies, as illustrated in Fig. 2.

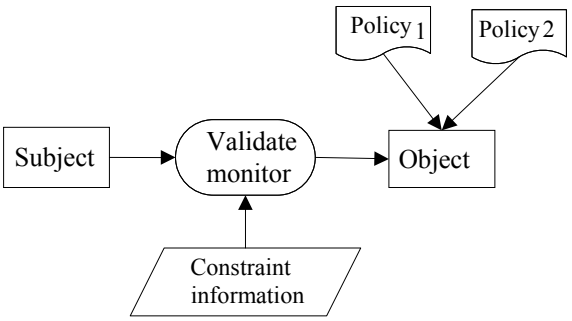

Figure 2. Engineering Process

3. CHARACTERISTIC: POLICY-ENFORCED AND CONTEXT CONSTRAINTS-AWARE

3.l Easing Policy-enforced Management Through Template

RBAC is used to alleviate users and resources administration [2]. In RBAC, each role has a set of permissions for operating on resources. Although a user may play several business roles, during a particular period of time, the user's privileges must comply with the security. Existing access control mechanisms are inflexible and do not help in alleviating the management task of administrating users access to resources based on permissions. Because the access control permission usually changes over a system life cycle, policy which is defined above, offers a key benefit through its ability to modify security policies enforced on a role's permissions. By introducing security policies into a role's permissions in RBAC, policy-enforced access control makes the administration of user's access to resources less of a burden to system administrators [6].

Through binding security policy to object, the complexity of security and trust management can be largely eased. Since every sensitive object needs access control, there will be many policies. However, these policies have many common functions. For example, in order to prevent the uploading of sensitive files, each organization unit has a role censor to examine the contents. Though each unit censor is a unique role, the operations that he can perform to the files in his charge are universal. This gives a way to ease the management of policies through policy template. Firstly, the

template defines comment operations that roles can do. Secondly, when it comes to a specific organization unit, the template role can be substituted by specify roles [7].

Here is an example. In template,

<kbs:Context rdf: about="$Template $/commonuser">.

In policy,

<kbs:Context rdf:about="$Unit A$/commonuser">

Several languages have been proposed to describe policies. The main drawback with them is that they are either too complicated for a non-programmer to use or too specific to describe more general policies [6]. There is a problem in designing a language that is both simple and expressive. An ideal solution is XML language which has the following characteristics:

- It is easy to use so that writing a policy is as easy as describing what it is.
- It is flexible so that various constraints can be described.
- It has good programmability.
- It supports writing of pre-set policy templates, which are more succinct than instantiated policies.
- It is capable of refinement since policy development is a process of converting a high-level policy specification to a machine understandable code.
- It has a unified mechanism. Policies, credentials, and trust relationships can be represented and processed in a unified manner, rather than being dealt with separately.

3.2 Context Constraint Aware to Support SoD

Most computer security mechanisms do not easily accommodate SoD rules. Although RBAC can make authorization decisions based on a user's role, it is not sufficiently powerful to enforce rules that are dependent on run-time parameters, in order to take into account the context information and thus being able to adapt itself based on the current conditions [4]. Thus, context has to be incorporated in the access control mechanism, in order to extend the traditional RBAC model to gain many advantages from its context-aware capability. A typical authorization policy in RBAC is represented as "User U in role R has permission P". A context-aware access control mechanism is represented as "User U in role R who fulfills constraint C has permission P" [8].

A context constraint is defined through the following terms: context attribute, context function, context facts and context condition [2]:

A context attribute represents a certain property of the environment whose actual value might change dynamically (like time, date, or session-data for example), or which varies for different instances of the same abstract entity (e.g. location, ownership, birthday, or nationality).Thus, context attributes are a means to make context information explicit.

A context function is a mechanism to obtain the current value of a specific context attribute (i.e. to explicitly capture context information).

A context condition is predefined constant or another context attributes of the same domain in policies.

First, we need to define the context attribute, then, we collect context facts through context function, and finally, context facts will be compared with the context conditions to see if the conditions are satisfied.

Such constraints may include object properties such as lifestate as well as run-time conditions.

3.3 A Policy Example

To better illustrate the characteristics of the approach, an example is given below:

```
<acr: allow>
<acr: group>
<acr: operation    acr:name="content:list"/>
<acr: operation    acr: name="content:view"/>
</acr:group>
 <acr:group>
  <acr:role rdf: resource="urn:role-001" />
   <acr:role rdf: resource="urn:role-002" />
<acr:role rdf:resource="urn:role-003" />
</acr:group>
<acr:group>
    <acr:constraint>
  <acr:condition acr:name="location" acr:value="intranet"/>
<acr:condition          acr:name="security-level"….>
</acr:constraint>
</acr:group>
 </acr:allow>
```

In the policy file, whether the relationship of constraint conditions should be optional or full meet depends on the design mechanism. It provides flexibility and extensibility.

4. APPLICATION

The approach is proved to be effective when it is applied to a Knowledge Management System in an aviation enterprise.

The fine-grained access control is required and the complicated environment is incorporated in the aviation enterprise. To solve the security problem and to enhance resources share, we develop the object-dependent, policy-enforced, context constraints-aware approach to ensure the security of sensitive resources.

The application is based on java platform. We use tomcat as server, Berkeley DB as database and XML language to describe policy.

The whole process is performed as follows:

1. Initialize system memory by uploading policy and role template.
2. Assemble template to specific policies.
3. Bind necessary policies to a particular object.

4. When a user performs an operation on an object, the validate monitor will make a check. First, the validate monitor will find out policies that are banded to the object, and reduce the policies to a set of rules.
5. Then the validate monitor will need to collect context constraint information, and figure out under the context constraint facts, which roles are authorized to perform the operation according to the rules.

Finally, the validate monitor will compare the roles authorized with the roles that the user is granted to see if matched roles exit. If the two role sets has an intersection, the user can perform the operation, else the request will be denied.

The process is described as in Fig.3.

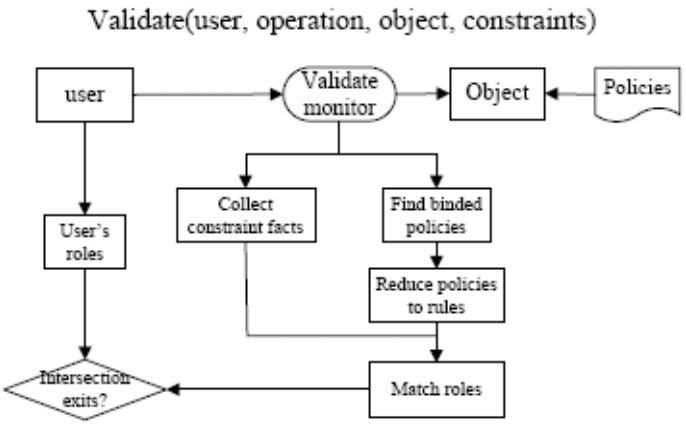

Figure 3. Implement Process

During the application in the aviation knowledge management system, the approach is proved to be effective to support fine-grained access control.

5. CONCLUSIONS AND FUTURE WORK

This paper proposes an approach to implement fine-grained RBAC model considering the specific context constraint. This approach has the following characteristics.

- It is object-dependent. To support fine-grained access control, we consider particular object through binding policies to it.
- It is policy-enforced. Policy-enforced access control makes the administration of users' access to resources less of a burden to system administrators.
- It is context constraints-aware. Context has been incorporated in the access control mechanism, in order to extend the traditional RBAC model to gain many advantages from its context-aware capability.

In our application, the approach is proved to be effective to support fine-grained access control and accommodate flexible policy.

Further research topics include: the effort to ease policy management; if we use template, there is a requirement to manage templates; To better support SoD, we must consider more static constraints and incorporate them to our approach.

ACKNOWLEDGEMENTS

Funding for this research was supported by the National Natural Science Foundation of China under Grant No.70671007 and the PhD Program Foundation of Education Ministry of China under Contract No. 20040006023.

REFERENCES

1. D.F. Ferraiolo, D.R. Kuhn, and R. Chandramouli, Role-Based Access Control (Artech House: 2003).
2. G.J. Ahn and R. Sandhu, Role-based Authorization Constraints Specification, *ACM Transactions on Information and System Security*. Volume 3, Number 4, (2000).
3. N.R. Adam, V. Atluri, E. Bertino, and E. Ferrari, A Content-Based Authorization Model for Digital Libraries, *IEEE Transactions on Knowledge and Data Engineering*. Volume 14, Number 2, (2002).
4. K. Alghathbar, *An Approach to Engineer and Enforce Context Constraints in an RBAC Environment* (SACMAT, 2003).
5. V. Kapsalisa and L. Hadellisb, A dynamic context-aware access control architecture for e-services, *computers & security*. Volume 25, pp.507-521, (2006).
6. A. Lin and R. Brown, The application of security policy to role-based access control and the common data security architecture, *Computer Communications*. Volume 23, pp.1584-1593, (2000).
7. L. Giuri and P. Iglio, Role Templates for Content-Based Access Control. in *Proc. of the ACM Workshop on Role-Based Access Control* (1997).
8. K. Alghathbar, Validating the enforcement of access control policies and separation of duty principle in requirement engineering, *Information and Software Technology*. Volume 49, pp.142-157, (2007).