Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2006 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

2006

An Administrative Model for Role-Based Access Control Using Hierarchical Namespace

Luning Xia
Chinese Academy of Science, halk@lois.cn

Jiwu Jing
Chinese Academy of Science, jing@is.ac.cn

An Administrative Model for Role-Based Access Control Using Hierarchical Namespace

Luning Xia
State Key Laboratory of Information Security, Chinese Academy of Science, Beijing, China
halk@lois.cn

Jiwu Jing
State Key Laboratory of Information Security, Chinese Academy of Science, Beijing, China
jing@is.ac.cn

Abstract

Access Control is an important mechanism of information security. Role-Based Access Control is a famous access control approach with good flexibility. RBAC96 and ARBAC97 are classical RBAC models. The ARBAC97 model facilitates decentralized administration of RBAC. However, ARBAC97 has some shortcomings in the case of being used in an organization with autonomous subsidiaries. The member of an administrative role can operate directly in the role range of a junior administrative role, which violates the autonomy of subsidiaries. We propose a new model named N-RBAC to overcome this weakness. In N-RBAC, roles are arranged according to a hierarchical namespace structure. Thus the role hierarchy is constructed in a local space instead of in a global space. The N-RBAC model does a better work in decentralized role administration in those organizations composed of autonomous subsidiaries.

Keywords: RBAC, RBAC96, ARBAC97, N-RBAC, Namespace

1. Introduction

Access control is a key mechanism to protect data from unauthorized access. Discretionary Access Control (DAC) and Mandatory Access Control (MAC) had been used for more than twenty years before the appearance of Role-Based Access Control (RBAC). With the development of computer science since 1980's, many new requirements of access control were brought forward, which couldn't be fulfilled under the framework of DAC or MAC. More extensibility and flexibility are needed than DAC and MAC can provide.
RBAC was proposed under this background. Essentially, RBAC is a mandatory access control model because it forbids delegating permissions to other users. However, the direction of information flow was not limited in RBAC. An intermediate element, the role, was introduced as a media to deliver the authorization information.
The original formal definition of RBAC was from (Ferraiolo et al. 1992). Ravi Sandhu and his Laboratory of Information Security Technique (LIST) of George Mason University proposed the famous RBAC96 model (Sandhu et al. 1996) in 1996. They divided traditional RBAC model into four conceptual models, and provided their formal definitions. Further in 1997, they proposed an administrative RBAC model named ARBAC97 to guide the decentralized role administration (Sandhu et al.. 1999). RBAC96 and ARBAC97 represented

the essence of role based access control well. They are both classical RBAC models. Most subsequent researches of RBAC were based on the two models.

2. Autonomy Problems In RBAC

The ARBAC97 model is more suitable to be applied in decentralized RBAC administration than previous models. However, it has some shortcomings if be used in an organization that composed of one or more autonomic branches.

For the convenience of description, let's consider a newspaper office named VERYNEWS. VERYNEWS comprises three channels: Society Channel, Entertainment Channel and Military Channel. Each channel is an autonomic branch of VERYNEWS and maintained by an independent editor team.

Now we apply RBAC96 and ARBAC97 to build the RBAC system of VERYNEWS. The regular role hierarchy of VERYNEWS is shown in Figure 1, and the administrative role hierarchy in Figure 2.

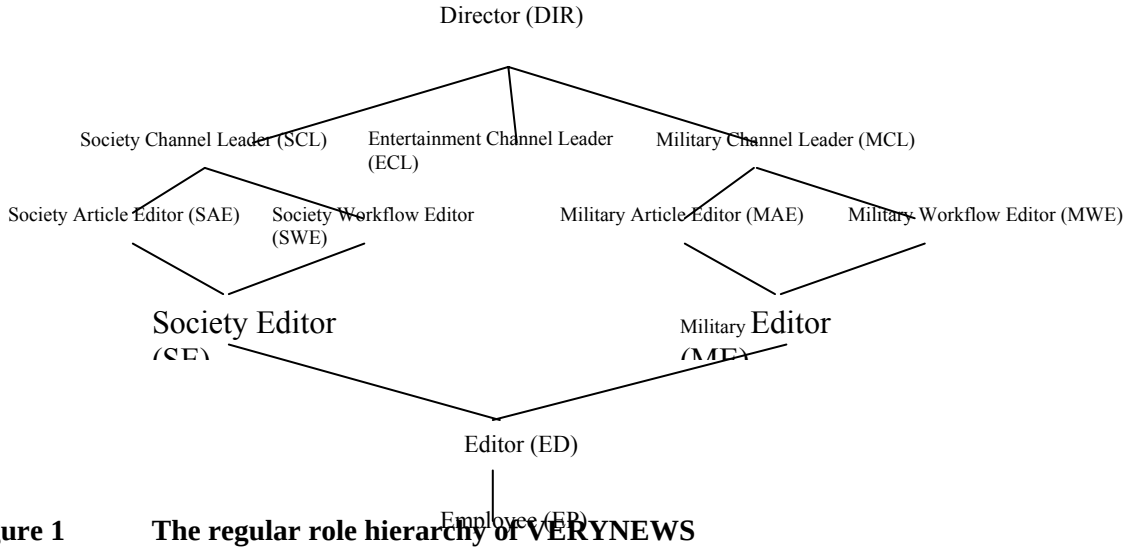

Figure 1 The regular role hierarchy of VERYNEWS

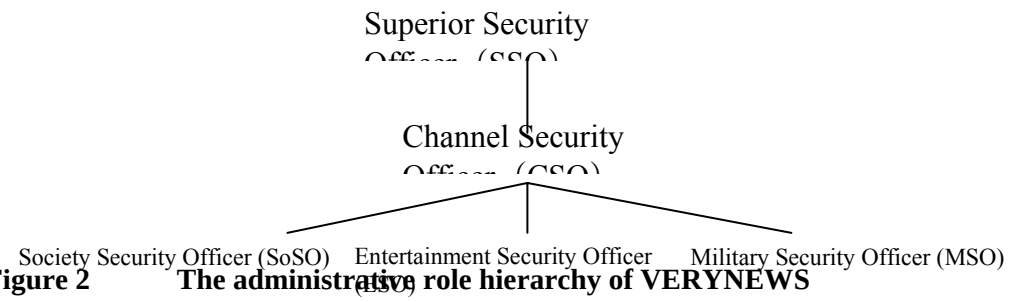

Figure 2 The administrative role hierarchy of VERYNEWS

Part of the *can_assign* relations of URA97 is shown in Table 1.

Table 1part of the *can_assign* relations of URA97

Administrative Role	Prerequisite Condition	Role Range
SoSO	ED	$[SE, SE]$
SoSO	$SE \wedge \overline{SWE}$	$[SAE, SAE]$
CSO	$ED \wedge \overline{ME}$	$[SE, SE]$
CSO	ED	(ED, DIR)
SSO	EP	$[ED, ED]$
SSO	ED	$(ED, DIR]$

Through an analyzing to Figure 1, Figure 2 and Table 1, following shortcomings of above model are addressed.

SC1. Over-administration

A senior administrative role inherits all permissions from its junior administrative roles in Figure 2. This implies that a member of the senior administrative role can operate directly within its junior administrative roles' role ranges. For example, a member of CSO can assign John to the role 'SAE' only if he has been a member of 'ED'. Given the relation $can\_assign(SoSO, SE \wedge \overline{SWE}, [SAE, SAE])$, this operation invalidates the prerequisite

condition $SE \wedge \overline{SWE}$ and thus bypasses SoSO. This is an intervention to the role administration of the autonomic branch 'Society Channel', and violates the autonomy of it.

SC2. The complexity of role naming

There may be a great many of roles existing in the RBAC system of a large organization. Obviously the name of each role has to be unique. In Figure 1, 'Society Article Editor (SAE)' and 'Military Article Editor (MAE)' are all article editor roles, but they have to be prefixed by 'Society' and 'Military' to keep the uniqueness of their names. If a role junior to 'SAE' is to be added, we have to name it by a longer name such as 'Society XXX Article Editor'. More complex the role hierarchy is, more long names we have to use.

SC1 is originated from the domination relationship of the administrative roles. The permissions of an administrative role can be inherited by a senior administrative role, which enables the senior one to do anything that the junior one can do.

SC2 is originated from that all the roles in RBAC96 and ARBAC97 are defined in a global namespace. We have to use different names to ensure the uniqueness.

We propose a new model to overcome the both shortcomings. We extend the concept of *organization structure* introduced in (Sejong et al. 2002) to a hierarchical namespace structure. The roles are not defined in a single global namespace any longer but in many different namespaces respectively. Roles defined in a namespace cannot see any roles out of the namespace. We call it the *Namespace-Based RBAC (N-RBAC)* model.

3. The N-RBAC Model

3.1 Namespace

Namespace is a popular term in cyber science. It can be defined as follows (SUNY O 1993).

Definition 1 A *namespace* is an autonomic scoping construct to subdivide the set of names and their visibility within a system.

A namespace may have several sub namespaces. All the namespaces compose of a tree-like *namespace hierarchy*.

The word 'names' in definition 1 refers to all the symbols defined in current system. In a RBAC system it includes user name, role name, permission name or the names of other resources. A name defined inside a namespace must be unique. Two names in two different namespaces can use the same symbol because their visibility is restrained by their namespaces. For example, the name 'Article Editor (AE)' can be defined in both namespace A and namespace B. Within any namespace 'AE' is a unique symbol. They can be referred as A.AE or B.AE out of their own namespace to ensure the global uniqueness. Actually this is a kind of segmented naming style and has no essential difference with the long naming style described in SC2. However, under the assumption that most of the operations are inside a certain namespace in a N-RBAC system, we believe the introduction of the namespace hierarchy will remarkably alleviate the naming trouble described in SC2.

The visibility of a name is restrained by its namespace. There will be no dominance relation between any roles in different namespaces, even if these namespaces are directly

senior and junior ones. We can regard the term *namespace* as an abstract notation of the *autonomic branch* described in section 2.

3.2 Resources and Operations

In access control, the meaning of *permission* is allowing a user to operate on a certain object. We divide the permission set *P* into two sets: the resource set *RS* and the operation set *O*. The relation of *P, RS* and *O* is defined as follows.

$$P \subseteq RS \times O$$

RS denotes all the resources that need to be protected by the RBAC system. In N-RBAC these resources are subdivided into different namespaces.

O denotes the operations that are worked on a certain resource, such as creation, deletion and modification.

The *user*, *role* and *namespace* are also resources in N-RBAC. The convenience of treating them as resources is that we can use a single role set *R* to represent all administrative or regular roles.

Definition 2 is the formal description of the N-RBAC0 model. It is derived from the RBAC0 model (Sandhu et al. 1996, pp. 8).

Definition 2. The N-RBAC0 model has the following components:

1) *U*, *R*, *RS*, O, *S* and *N* (users, roles, resources, operations, sessions and namespaces),

2) $UA \subseteq N \times U \times R$, a many-to-many user to role assignment relation, $\forall n \in N, n.UA \subseteq U \times n.R$

3) $PA \subseteq N \times RS \times O \times R$, a many-to-many permission to role assignment relation, $\forall n \in N, n.PA \subseteq n.RS \times O \times n.R$

4) $user : S \to U$, a function mapping each session *s* to the single user *user*(*s*) (constant for the session's lifetime), and

5) $role : S \to 2^R$, a function mapping each session *s* to a set of roles $roles(s) \subseteq \{r \mid (user(s), r) \in UA\}$ (which can change with time) and session *s* has the permissions $\cup_{r \in roles(s)} \{(rs, o) \mid (rs, o, r) \in PA\}$

The main difference between RBAC0 and N-RBAC0 is the introduction of namespaces. The roles and other resources are subdivided into the namespace hierarchy. Note that the user set *U* is defined globally. The reason to treat *user* as a global resource is that in most organizations the human resource is managed wholly, even if they are composed of many autonomic branches.

The N-RBAC1 model can be defined similarly, and the N-RBAC2 model is unchanged from RBAC2. Both N-RBAC1 and N-RBAC2 will not be discussed in this paper.

3.3 The Administrative Roles

As described in 1.3, the administrative roles are those roles that have the permissions to create, delete a regular role or modify the dominance relation of regular roles. In N-RBAC, the administrative roles have the permissions to operate on *user*, *role* or *namespace*. The *user, role* and *namespace* are also resources as described in 3.2. Thus we can represent all roles by a single role set *R.* The *namespace* resource of a namespace refers to the sub-namespaces of it.

Still we can create an administrative role hierarchy within a namespace. However, the introduction of the administrative role hierarchy in ARBAC97 is to facilitate

decentralized administration of roles. In N-RBAC, we use the namespace hierarchy to deal with decentralized administration of roles. So we discard the administrative role hierarchy and define only one administrative role for each namespace.
We define following constrains on the administrative role:
Constraint 1: There is only one administrative role in a namespace.
Constraint 2: the resources *user*, *role* and *namespace* can only be accessed by the administrative role.

The two constraints imply there is only one administrative role that can create or delete roles or modify the dominance relation of roles in a namespace. The operations of creating, deleting or modifying users can only be done by the administrative role of the root namespace.
Administrative roles in different namespaces have no dominance relation. The administrative role of a namespace **cannot** modify the URA, PRA and RRA relation of junior namespaces. Those operations can only be done by their own administrative roles. By this SC1 is overcome.

3.4 Origin of A Namespace

As a kind of resource, a namespace is created by the administrative role of its senior namespace. The administrative role is a concomitant of the namespace. That means, as soon as the namespace is created, the administrative role is produced as well. The administrative role cannot be deleted or modified during the lifecycle of the namespace to which it belongs. Thus we define another constraint as follows.
Constraint 3 The administrative role can only access the resources *user, role* and *namespace.*
This constraint implies that the administrative role cannot access those resources out of *user, role* and *namespace.* The duty of an administrator is to manage the N-RBAC system. He has no permission to access other resources such as the business policy, the system parameters, and so on. Those resources will be maintained by some regular roles within the namespace.

3.5 URA, PRA and RRA In N-RBAC

As described above, a fix-authorized administrative role is employed instead of an administrative role hierarchy to administrate the RBAC system. This change largely simplifies the URA, PRA and RRA relations.
For a *can_assign* relation expressed as $can\_assign(x, y, [a, c])$, the administrative role x is unique in a namespace; and the role range of it is the whole regular role hierarchy of the namespace. The prerequisite condition y is not needed any more, because a member of the administrative role can directly assign a user to any roles in the namespace. In fact the *can_assign* relation has disappeared in N-RBAC. In a well-constructed namespace hierarchy, the organization structure within a namespace is often compact and centralized. So The URA model is not needed, as well as the PRA model.
The *can_modify* relation is simplified due to the disappearance of the administrative role hierarchy. However, any constraints defined in RRA model are still required to maintain global consistency of authorization.

4. A Sample of N-RBAC

We implemented the N-RBAC system of VERYNEWS as an illustration to N-RBAC. Figure 3 shows the namespace hierarchy of VERYNEWS.

This is a two-level namespace hierarchy. The three channels are sub-namespaces of the root namespace.

The resource set *RS* is defined as follows.

$$RS = \{Article, Column, Template, Workflow, Role, User\}$$

Article represents the published or unpublished articles in VERYNEWS.

Column represents a group of articles.

Template represents the layout templates of the display pages of articles and columns.

Workflow represents a sequence of procedures to process an article. These procedures may include creation, verification, publish, and so on.

Role represents the regular or administrative roles.

User represents all the users of the RBAC system.

The operation set *O* is defined as follows.

$$O = \{Create, Modify, Enable, Disable, Delete\}$$

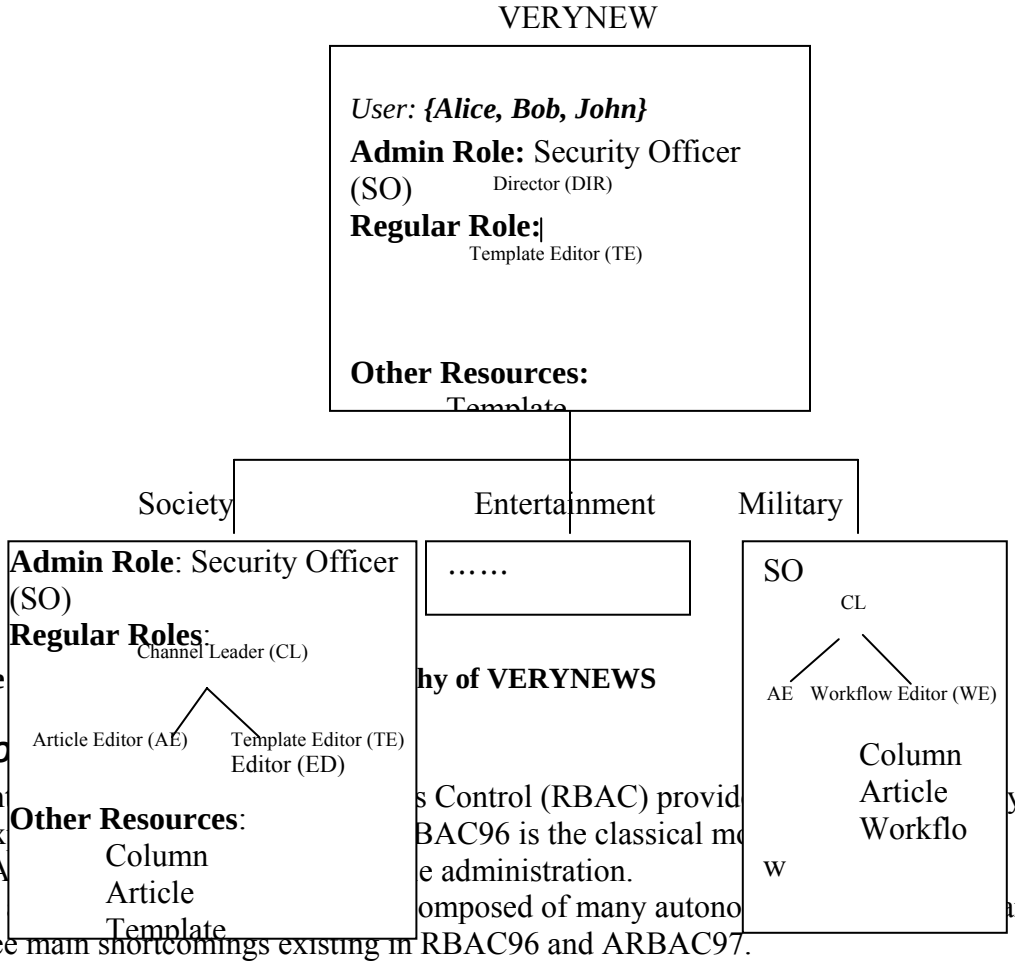

Figure hy of VERYNEWS

6. Co

The in s Control (RBAC) provid y and ex BAC96 is the classical m ARBA e administration.

When omposed of many autono are three main shortcomings existing in RBAC96 and ARBAC97.

SC1. Over-administration

SC3. The complexity of role naming

We proposed a new model named N-RBAC and introduced the namespace hierarchy to fulfill the autonomic requirements in RBAC administration. The N-RBAC model subdivides the roles and other resources into multi-level namespaces. The visibility of the resources of a namespace is also limited by the namespace. The administrative role hierarchy in ARBAC97 is discarded in our model. There is only one fix-authorized administrative role in a namespace. The administrative roles of different namespaces have no dominance relation, which overcomes SC1. The limited visibility of the resources in a namespace arouses the possibility of name reuse in different namespaces. By this way, SC2 is settled.

The N-RBAC model has good expansibility and compatibility. The RBAC96 or ARBAC97 models can be implemented without any change in a namespace, which facilitates the upgrading from existing RBAC systems to N-RBAC.

References

Ferraiolo, D., and Kuhn, R., "Role Based Access Controls," In *15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554-563.

Sandhu, R. S., Coyne, E. J., Feinstein, H. L., and Youman, C. E., "Role-Based Access Control Models," *IEEE Computer* (29:2), 1996, pp. 38-47.

Sandhu, R. S., Bhamidipati, V., and Munawer, Q., "The ARBAC97 Model for Role-Based Administration of Roles," *ACM Transactions on Information and System Security* (2:1), 1999, pp. 105-135.

Sandhu, R. S. and Bhamidipati, V., "Role-Based Administration of User-Role Assignment: The URA97 Model and its Oracle Implementation," *Journal of Computer Security* (7), 1999.

Sandhu, R. S. and Bhamidipati, V., "An Oracle Implementation of the PRA97 Model for Permission-Role Assignment," In *Proceedings of the 3rd ACM Workshop on Role-Based Access Control (RBAC'98)*, 1998.

Sejong O and Sandhu. R, "A Model for Role Administration Using Organization Structure," In *SACMAT'02*, June 3-4, 2002.

SUNY O, "WISR 1993 Design-for-Reuse Working Group Report," November 3, 1993, retrieved from http://gee.cs.oswego.edu/dl/WISR93WG/WISR93WG/WISR93WG.html at March. 2006.