A Role Based Access Control Method Considering Tasks in the Mobile Agent-based Workflow System

Seong-Min Jeong, Seung-Wan Han, Hyeong-Seok Lim
Dept. of Computer Science, Chonnam National Univ.
300 Yongbong-Dong, Puk-Gu, Kwangju, KOREA
Tel: +82-62-530-0755, Fax: +82-62-530-3439
E-mail: {u9897888, hansw, hslim}@chonnam.chonnam.ac.kr

Abstract: We address an access control in the mobile agent-based workflow system. The Role Based Access Control (RBAC) is suitable to model the access control for business processes. However, current RBAC models are not adequate to mobile agent-based workflow system. Because separation of duties becomes complicated and it is impossible to perform several workflows at the same time. To solve these problems, we limit the scope of privilege within the specified task. We define considerations, specification of constraints needed in RBAC when tasks are involved. Also, we present an access control scenario and algorithms in the mobile agent-based workflow system.

1. Introduction

A workflow is a business process in an organization. Workflow Management System (WFMS) completely defines, manages and executes a workflow [1]. WFMS is used to run in numerous application including office automation, banking, telecommunications, and manufacturing. Recently business processes become more complicated and they are often demanded in distributed environments [2]. However, currently centralized WFMS hardly provide support for it. In such environments, mobile agent paradigm can be a solution to the problem using autonomy, mobility and interaction of mobile agents [3].

To make a workflow system suitable in distributed environments, currently we are developing mobile agent-based workflow system [4] that uses mobile agent technology. The mobile agent-based workflow system is composed of 3-layerd architectures. Workflow coordinator in layer-1 manages several workflow engines. Workflow engines in layer-2 make sub-agent and assigns a specific role to each task, and task performers in layer-3 perform workflows. The system needs identification, authentication, privilege, access control in order to process workflow securely. Among these, access control is used to protect resources from illegal accesses [5]. Mandatory Access Control (MAC) is based on the rules, security level that a security administrator sets up. It has a very narrow application. Discretionary Access Control (DAC) is granted or revoked at discretion of the owner of information. A defect of DAC is that the central controls are difficult. Role Based Access Control (RBAC) is determined a role in the organization. A role characterizes the functions that a user is allowed to perform within an organization. Users are assigned to appropriate roles based on their qualifications and responsibilities. Specifying privileges on roles is not only convenient but also reduces the complexity of access control because the number of roles in an organization is significantly smaller than that of users [6]. RBAC provides more flexibility and applicability on the various commercial fields and business processes than MAC and DAC [7].

This work was supported by the Korea Science and Engineering Foundation under contract 98-0102-11-01-3.

RBAC is suitable to WFMS. However, current RBAC models are not adequate to mobile agent-based workflow system, because separation of duties becomes complicated and it is impossible to perform several workflows at the same time. Therefore, we limit the scope of privilege within the specified task.

The remainder of the paper is organized as follows. Section 2 reviews access control and current RBAC models in workflow system. Section 3 defines considerations, specification of constraints, presents access control scenario and algorithms. Finally, Section 4 provides conclusion.

2. Related Work

Access control can be divided into MAC, DAC and RBAC. MAC is usually used for strict security because it enables central controls by the security manager with the security level and rules. However, it has not only difficulty of determining the level of security, but has inefficiency for task performances in distributed/commercial environments where there are a lot of users and objects even if the security level is determined. In DAC, the grant, the proxy and the revocation of roles for objects can be done by the owner subject. So there can be a flexibility in security controls. But, the controls by a security manager are impossible and system control becomes difficult where the users are changed frequently. RBAC is appropriate for distributed/commercial environments because it is based upon user's role in the organization. In RBAC, the privilege assignment to a role is easier than the assignment to each user, and complexities of user addition and deletion can be reduced. But, it has a demerit that a user with several roles can act against the security.

The current access control in WFMS is as follows: The separation of duties is considered in the current WFMS in [2]. Users are assigned to appropriate roles based on their qualifications and responsibilities and it has an assumption that the assignment also gives users privileges on roles. Constraints are specified and algorithms for examining constraints in term of consistency are presented. Current RBAC models are not adequate to mobile agent-based workflow system, because the assumption in [2] is granting users' privileges on roles without considering tasks and those constraints applied to a single workflow system. [8] proposed authorization-step, executor permission and enabled permission, within the specified task in order for Task-Based Authorization access control.

So for, we mentioned access control and the current access control in WFMS. A RBAC method considering tasks in the mobile agent-based workflow system has several merits. It enables strict security access control with security level and the rules and also can cover the defect of MAC which has a very narrow application and can cover the defect of DAC that the central controls are difficult owing to distributed security control. Hierarchical role structure helps efficient task performances and a role, not a user, has privilege of objects so that this system is much suitable for business processes. However in the current RBAC, separation of duties becomes complicated in order to stop improper use of privilege, which a role already took in the whole WFMS. If one user cannot have mutually exclusive roles through the separation of duties, problem can happen that he cannot execute, at a time, two workflows, which have no relation with each other.

Therefore, we would enforce the least privilege by limiting the scope of privilege, which a role can have within the specified task and subdividing the separation of duties. Two tasks, which are not related with each other, can be executed with mutually exclusive roles by assigning a role to different privilege respectively in each task as well. It is no wonder that privilege assignment can be complicated because we additionally should consider the relation between roles and tasks. But, extra work won't be needed after building WFMS because the change of roles and tasks in significantly smaller than that of users.

3. A Role Based Access Control Method Considering Tasks

3.1 Consideration

We consider the following to propose the task considering RBAC model.

(1) Role Assignment

Users should be assigned to appropriate roles based on their qualifications, responsibilities and tasks they participate in.

(2) Task Considering Privilege Assignment

The executing privileges to roles in each task should be different. That is, the limit of the scope of privilege within the specified task can strengthen least privileges.

(3) Role Hierarchy

The possibility of inheritance should be determined not only to be appropriate for commercial environments, which have hierarchical privilege structure, but also to assign roles efficiently. When role hierarchy is taken into account, task hierarchy should be considered as well.

(4) Separation of Duties

Because various tasks depend on users' roles in workflow system, roles should be checked if they are mutually exclusive and both static separation of duties and dynamic separation of duties between tasks should be considered together.

(5) Task and Role Assignment in 3-layerd Architecture

A workflow coordinator establishes all of roles needed for workflow system. A proxy agent in workflow engine makes sub-agent and assigns a specific role according to each task. Task performer limits the scope of privileges within that task.

3.2 Specification of Consideration

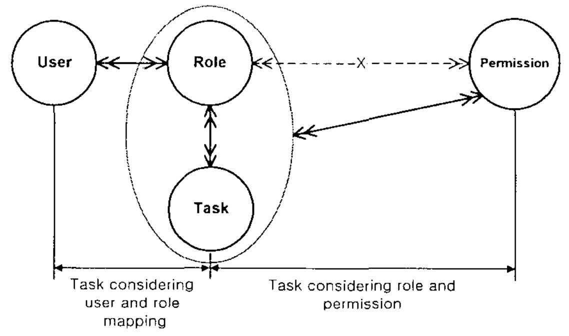
User
Role
X
Permission
Task
Task considering user and role mapping
Task considering role and permission

Fig. 1. Task considering user and role, role and permission mapping

In here, we redefine the current RBAC [9] considering tasks like Figure 1.

Rule 1 [Role Assignment]

Rule 1-1: Users are assigned to roles based on qualifications and responsibilities and obtain the roles only related with the task.

$\forall$ *u: user, r: role, s: subject, t: task*
$u \in$ *t-authorized-user(s)* $\wedge$ $u \in$ *t-role-member(r)*
$\Rightarrow u \in$ *t-authorized-role(s)*

Rule 1-2: Users are admitted to have least privileges and least roles which are required for task execution.

$\forall$ *u: user, t: task*
t-role-number(u) $\leq$ *t-role-limit(u)*

Rule 2 [Role Hierarchy]

Rule 2-1: Higher positional roles include task-related low positional roles

$\forall$ *s: subject,* r_i, r_j*: role,* t_1, t_2*: task,* r_iRt_1, r_jRt_1, r_jRt_2
(R is relationship)
$r_j > r_i$ $\wedge$ $r_j \in$ t_1*-authorized-role(s)*
$\wedge$ $r_j \in$ t_2*-authorized-role(s)*
$\Rightarrow$ $r_i \in$ t_1*-authorized-role(s),*
$r_i \notin$ t_2*-authorized-role(s)*

Rule 3 [Separation of Duty]

Rule 3-1: One user charging with mutually exclusive roles cannot execute one task.

$\forall$ *u: user,* r_i, r_j*: role, t: task,* $i \neq j$
(Each role is mutually exclusive role)
t-execute(r_i) $\wedge$ *t-execute*(r_j)
$\Rightarrow$ $(u \in$ *t-authorized-role*$(r_i))$ $\oplus$
$(u \in$ *t-authorized-role*$(r_j))$

Rule 3-2: Even though roles assigned to one user are mutually exclusive, independent tasks can be processed by the user.

$\forall$ *u: user,* r_i, r_j*: role,* t_1, t_2*: task,* $i \neq j$
(Each role is mutually exclusive role)
t_1*-execute*(r_i) $\wedge$ t_2*-execute*(r_j)

$\Rightarrow (u \in t_i\text{-}authorized\text{-}role(r_i)) \oplus$
$(u \in t_2\text{-}authorized\text{-}role(r_j))$

Rule 3-3: One user cannot execute mutually exclusive tasks.

$\forall$ *u: user, r: role, t_i, t_j: task, $i \neq j$*
(Each task is mutually exclusive task)
$t_i\text{-}execute(r) \wedge t_j\text{-}execute(r)$
$\Rightarrow (u \in t_i\text{-}authorized\text{-}role(r)) \oplus$
$(u \in t_j\text{-}authorized\text{-}role(r))$

Rule 3-4: One user cannot perform two tasks associated with each other hierarchically.

$\forall$ *u: user, t_i, t_j: task, $i \neq j$*
$t_j > t_i \wedge t_i\text{-}execute(r) \wedge t_j\text{-}execute(r)$
$\Rightarrow (u \in t_i\text{-}authorized\text{-}role(r)) \oplus$
$(u \in t_j\text{-}authorized\text{-}role(r))$

Rule 4 [Role Execution]

***Rule* 4-1**: *User's roles get different permissions* according to tasks.

$\forall$ *s: subject, p_i, p_j: permission, $t_i, t_j \in$ task, $i \neq j$*
$\{t_i, execution(s)\} \Rightarrow execution(s, p_i)$
$\{t_j, execution(s)\} \Rightarrow execution(s, p_j)$

Rule 5 [Object Access Authorization]

Rule 5-1: If a role is authorized by an object within a specific task, the role obtains the permission to the object.

$\forall$ *r: role, p: permission, t: task*
$r \in t\text{-}execute(r) \wedge r \in t\text{-}authorized\text{-}role(r)$
$\Rightarrow object(r, p)$

Rule 6 [Object Access Deadline]

Rule 6-1: If the task is finished, the privilege for the role is withdrawn.

$\forall$ *r: role, p: permission, t: task*
$r \in t\text{-}end(r) \Rightarrow object\text{-}end(r, p)$

3.3 Access Control Scenario

Figure 2 presents virtual scenario when access control is executed hierarchically in mobile agent-based workflow system. Each step is described in the following.

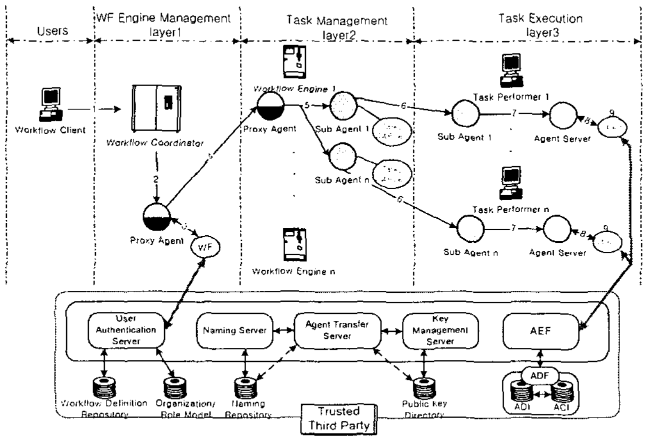

Fig. 2. Access Control Scenario

Step 1: Users request a certain task to a workflow coordinator.

Step 2: A workflow coordinator creates a proxy agent.

Step 3: A proxy agent takes the role in workflow system according to organization/role model of workflow system.

Step 4: The proxy agent is transferred to a workflow engine.

Step 5: The proxy agent creates sub-agents in each task and does scheduling for task execution in the workflow engine. And each sub-agent determines sub role needed for corresponding tasks.

Step 6: Each sub-agent makes his way to a task performer for task execution.

Step 7: Each sub-agent requests permission to agent server of each task performer.

Step 8: Agent server determines a permission for an object according to role-permission database.

Step 9: Each sub-agent with the permission for an object executes tasks.

3.4 Role assignment, Permission Assignment and Task Activation Algorithms

Figure 3 defines Role assignment related to users and tasks and permission assignment algorithm. Figure 4 defines task activation algorithm within the specified task.

```
Algorithm Role_Assignment&Permission_Assignment
/* Role assignment related to users and tasks and permission assignment algorithm */

Input: Users = user_1, user_2, ... user_l  /* The set of user */
       Roles = role_1, role_2, ... role_m  /* The set of role */
       Workflows = task_1, task_2, ... task_n
         /* The set of task which belongs to workflow */

Output: task_k_role_i[j] /* The role_j that user_i can have in task_k */
        permission_k_role_i[j] /* The permission of role_j that user_i can have in task_k */

begin

for i := 1 to l:
  for j := 1 to m:
    for k := 1 to n:

/* Possible role assignment according to user's privilege*/
  if user_i_privilege_degree ≤ role_j_privilege_degree:
    user_i_assignment[j] := role_j
  endif

/* Select the number of role-among roles' user is assigned the roles except for mutually exclusive ones */
  user_i_role_number := 0
  while user_i_role_number ≤ user_i_rolelimit_number:
    if user_i_select := user_i_assignment[j] and
                              mutuallyexclusiveroleset:
      assign user_i_select to role_i[j]
      user_i_role_number := user_i_role_number + 1
    endif
  endwhile

/* Among roles user selected, only assign the roles which is related with the participating tasks */
```

```
    if $role_i[j] \in task_k\_role$:
        assign $role_i[j]$ to $task_k\_role_i[j]$:
    endif
```

/* Among roles related to tasks, if one role is related to mutually exclusive tasks, assign privilege in order not for one user to execute the tasks */

```
    if $task_k\_role_i[j] \notin mutuallyexclusivetaskset$:
        assign $task_k\_role_i[j]$ to $permission_k\_role_i[j]$
    endif
endfor

end Role_Assignment&Permission_Assignment
```

Fig. 3. Role assignment related to users and tasks and permission assignment algorithm

```
Algorithm Task_Activation
/* Role activation algorithm within the specified task */

Input: Users = $user_1, user_2, \dots user_l$  /* The set of user*/
       Roles = $role_1, role_2, \dots role_m$  /* The set of role */
       Workflows = $task_1, task_2, \dots task_n$
        /* The set of task which belongs to a workflow */
        $task_k\_role_i[j]$  /* The $role_j$ that $user_i$ can have in $task_k$ */
        $permission_k\_role_i[j]$  /* The permission of $role_j$ that $user_i$ can have in $task_k$ */
Output: $task_k$-activation    /* Activation in $task_k$ */
begin
for i := 1 to l:
  for j := 1 to m:
    for k := 1 to n:

  /* Check if a user has the privilege */
    if $user_i \in authorizedset$:
      return TRUE
    endif

  /* Check if a user has a role which is needed in a task */
    if $user_i \in task_k\_rolei[j]$:
      return TRUE
    endif

  /* Check if a role has the permission for the task */
    if $task_k\_role_i[j] \in permission_k\_role_i[j]$:
      return TRUE
    endif

  /* if every condition is true, the user can execute a job in the task */
    if TRUE:
      $user_i$ has $task_k$-activation
    endif
endfor

end Task_Activation
```

Fig. 4. Role activation algorithm within the specified task

4. Conclusion

In the current RBAC, separation of duties becomes complicated in order to prevent the improper use of privilege when a role already took in the whole WFMS. Also, if one user cannot have mutually exclusive roles through the separation of duties, problem can happen that user cannot execute two independent workflows, at the same time. To solve these problems in the RBAC for the mobile agent-based workflow system, we enforced the least privilege by limiting the scope of privilege. So a role can have a privilege within the specified task. And by subdividing the separation of duties, it becomes possible that independent tasks can be executed with mutually exclusive roles by assigning a role to different privilege in each task. We defined considerations and specification of constraints needed in RBAC where tasks are involved. We present an access control scenario and algorithms in mobile agent-based workflow system.

In future, the research for the proper division of access control into agent level and workflow level will be needed.

References

[1] A. L. Scherr, "A new apporoch to business processes," IBM SYSTEMS Journal. Vol.32, No.1, pp.80-98, 1995.

[2] E. Bertino, E. Ferrari, V. Atluri, "The Specification and Enforcement of Authorization Constraints in Workflow Management Systems," ACM Transactions on Information and System Security, Vol.2, No.1, pp.65-104, 1999.

[3] Yariv Aridor, Mitsuru Oshima, "Infrastructure for Mobile Agents: Requirements and Design," Mobile agents: second international workshop; proceedings / MA'98, pp.38-49, 1998.

[4] Jeong-Joon Yoo, Youg-HO Suh, Sang-Bum Song, Dong-Ik Lee, "Agent-based Workflow System Architecture and Mobile Agent Requirements," Korea Information Processing Society, Vol.6, No.2, 1999.

[5] ITU-T Recommendation X.812 Information Technology - Open Systems Interconnection - Security Frameworks for Open Systems: Access Control Framework, 1995.

[6] Warwick Ford, Computer Communications Security, Prentice Hall, pp.149-158, 1994.

[7] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, C. E. Youman, "Role-Based Access Control Models," IEEE Computer, Vol.29, No.2, pp.38-47, 1996.

[8] R. K. Thomas, R. S. Sandhu, "Task-based Authorization Controls (TBAC): A Family of Models for Active and Enterprise-oriented Authorization Management," IFIT WG11.3 Workshop on Database Security, 1997.

[9] D. F. Ferraiolo, J. A. Cugini, D. R. Kuhn, "Role-based access control: Features and Motivations," 11th Annual Computer Security Applications Conference, 1995.