Injecting Task Delegation Constraints into a Role-based Access Control Model

Khaled Gaaloul[1], H.A. (Erik) Proper[1,2], and François Charoy[3]

[1] Public Research Centre Henri Tudor,
L-1855 Luxembourg-Kirchberg, Luxembourg
[2] Radboud University Nijmegen
P. O. BOX 9010 6500, GL Nijmegen, The Netherlands
[3] LORIA, Université de Lorraine
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
{khaled.gaaloul,erik.proper}@tudor.lu,charoy@loria.fr

Abstract. In role-based access control models, delegation of authority involves delegating roles that a user can assume or the set of permissions that he can acquire, to other users. Several role-based delegation models have been proposed in the literature. However, these models consider only delegation in presence of the role type, which have some inherent limitations to *task delegation* in workflow systems. In this paper, we address task delegation in a workflow and elaborate a security model supporting delegation constraints. Delegation constraints express security requirements with regards to task's resources, user's assignment and *privileges* (delegation of authority). Further, we show how, using a role-based security model, we inject formalised delegation constraints to compute delegation principals with their respective privileges.

Key words: Access control, delegation, constraints, privileges

1 Introduction

With the broad adoption of workflow management systems to model and automate business processes cross organisations, security becomes a crucial and essential topic. Typically, activities that are part of a process are represented as tasks. Organisations establish a set of authorisation policies that regulate how business processes and resources should be managed within a workflow [1]. Authorisation information is given which authorises users to perform tasks. Such authorisation information may be specified using a simple access control list or more complex role-based structures [2].

In current workflow management systems, the role-based access control (RBAC) model is widely adopted, where system administrators assign roles to users. It is more convenient for administrators to manage roles than to manage users directly [3]. One important factor that affects access control (authorisation) distribution among users is delegation. Delegation involves a user passing its authority to other users. If delegation is allowed, a delegator delegates authority (a privilege) to another active entity, called the delegatee, to carry out a task on behalf of the former. In the context of workflow systems, delegation can be very useful for real-world situations where a user who has to perform a task is either unavailable or too overloaded [4]. Hence, we define task delegation as a means for assigning a task and its access rights from a delegator to a delegatee.

The concept of delegation has been presented in [1, 5]. Significant contributions to role-based delegation can be found in [6, 7]. While much of the work in the area of delegation is limited to role-based access control, the goal of our paper is to consider task delegation constraints in workflow systems. Delegation constraints needs to tackle several issues with regards to workflow's invariants in terms of users, tasks and resources. In doing so, we need to come up with an access control model supporting the assignment of task delegation. Delegation assignment deals with delegation principals (delegator, delegatee) their respective rights (privileges) and their availability (no conflicts during task assignment). In this paper, we extend the RBAC model of Sandhu et al. in two directions: (i) our formal security model defines a *Task-oriented Access Control (TAC)* model which is capable of supporting task assignment condition in workflows and (ii) we leverage TAC specifications to inject delegation constraints, thereby computing potential delegatees and their required privileges.

The remainder of this article is organised as follows. Section 2 defines workflow authorisation constraints during task execution. In section 3, we present a formal security model to reason about task assignment within a workflow. This model is used in Section 4 to integrate delegation constraints in order to compute delegatees with their respective privileges. Finally, we conclude and discuss future work.

2 Workflow Authorisation Constraints

A workflow comprises various activities that are involved in a business process. Activities that are part of a process are represented as tasks [8]. Authorisation information is given which authorises users to perform tasks. Such authorisation information may be specified using a simple access control list or more complex role-based structures [9].

We define a task execution model using an activity diagram composed of three main activities : *Initialisation*, *Processing* and *Finalisation* (see Fig. 1). During the initialisation of the task, a task instance is created and then assigned to a user. During task processing, the assigned user can start or delegate the task which gathers all operations and rights over the business objects related to task's resources (see Definition 1). Finally, the task finalisation would notice the workflow management system that the task is terminated, where termination defines completeness, failure or cancellation.

Seeing a task as a block that needs protection against undesired accesses, the activity diagram includes an access control transition which is in charge of granting access to a task. Access control defines a transition from the creation of a task to its assignment to a user. This assignment will lead to the processing or

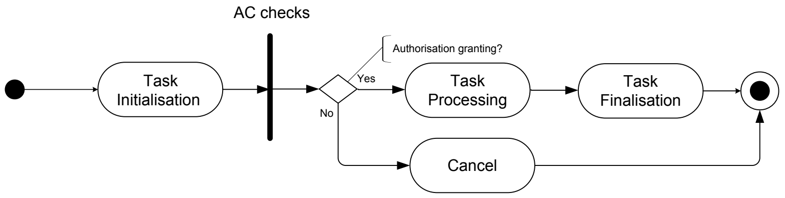

Fig. 1. Task execution model.

the cancellation of a task. Cancellation can be triggered when an assigned user does not fulfill the required authorisation to execute a task.

Definition 1 (Permission). P *is a set of permissions.* P *defines the right to execute an operation on a resource type. A permission* p *is a pair* (f,o) *where* f *is a function and* o *is a business object. We note :* $P \in F \times O$ *where* F *is a set of functions and* O *is a set of business objects.*

Authorisation information define the access control transition. We define a permission as an authorisation allowing a users to perform a task. Authorisation makes an explicit binding between a user, a task resource (business object) and his rights over it (fuction/action). In our work, we define a task oriented access control model based on the RBAC model. We focus on task's requirements to analyse and specify security constraints while accessing workflow's data. Data access defines permissions on business objects related to task's resources.

3 Task-oriented Access Control Model

We propose a *Task-oriented Access Control (TAC)* model to support authorisation requirements in workflow systems (see Fig. 2). Authorisation information will be inferred from access control data structures, such as user-role assignment (URA) and task-role assignment (TRA) relations. In addition, we model permission assignment relations for tasks and roles in order to support the task execution context. The remaining relations are generic relations based on the RBAC model [3].

Formally, we define sets *U, R, OU, T, P, S* and *TI* as a set of users, roles, organisations units, tasks, permissions, subjects and task instances, respectively. We use a subject to denote the time a user selects roles for a session. During the task instantiation assignment, we create a user's current active role set and define it as a subject (see Fig. 2). For example, the user *Alice* with the role *clerk* defines a subject to execute the instance of a task "Check credit" in a bank loan process.

We define *RH* (Role Hierarchy), where *RH* is a partial order on *R*, r_i and $r_j \in R$. *RH* denotes that r_i is a role superior to r_j, as a result, r_i automatically inherits the permissions of r_j.

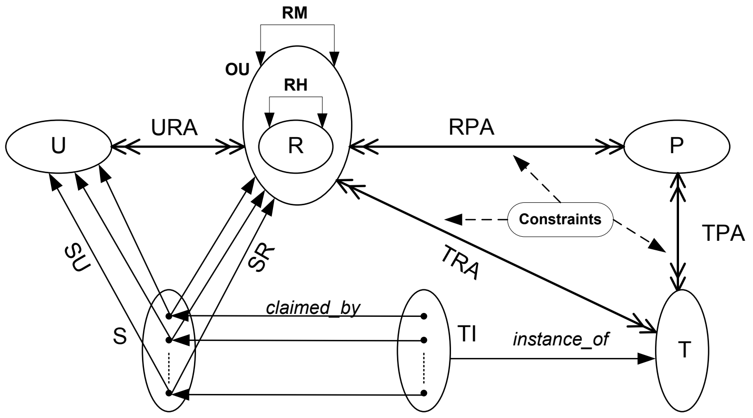

Fig. 2. Task-oriented access control (TAC) model.

We define RM (Role Mapping), where $RM \subseteq OU_i \times OU_j$ with OU_i and OU_j two organisations units. RM defines external roles accessing distributed resources cross-organisations. It provides a decentralised access control mechanism where externally known roles are publicly available :
$r_k \in OU_i$ and $r_l \in OU_j$, RM denotes that r_l is a role mapped to r_k, as a result, r_l shares the permissions of r_k.

3.1 Definitions of map relations

Formally, we define sets of relations as follows:

- $URA \subseteq U \times R$, the user role assignment relation mapping users to roles they are member of.
- $RPA \subseteq R \times P$, the permission role assignment relation mapping roles to permissions they are authorised to.
- $TPA \subseteq T \times P$, the task permission assignment relation mapping tasks to permissions. This defines the set of permission required to execute a task (see Definition 2).
- $TRA \subseteq T \times R$ the task role assignment relation mapping roles to tasks they are assigned to.

3.2 Definitions of functions

Formally, we define sets of functions as follows:

- $SU\colon S \rightarrow U$ a function mapping a subject to the corresponding user.
- $SR\colon S \rightarrow R$, a function mapping each subject to a role, where $SR(s) = r, (SU(s), r) \in URA\}$ with a subject s having a permission $p|(r,p) \in RPA\}$.
- $instance_{of}\colon TI \rightarrow T$, a function mapping a task instance to its task type.

- $claimed_{by}$: $TI \rightarrow S$, a function mapping a task instance to a subject to execute it. It defines the user-task assignment condition s = $claimed_{by}(\mathrm{t}_{i1})$ where : $\{t_i = instance_{of}(t_{i1}), (r,u) \in URA | (SR(s) = r \bigwedge SU(s) = u), (t_i, r) \in TRA\}$.

3.3 Definitions of constraints

Here we discuss Separation of duty (SoD) and Binding of duty (BoD) constraints. It defines security constraints between two tasks that compose a business process [10]. Such constraints help to verify whether a user is not allowed to execute a task due to some conflicts (e.g., conflict of interest). We define an exclusive relation between tasks for SoD, and a binding relation between tasks for BoD :

$$TT_{SOD} : \{(t_i, t_j) \in T \times T \mid t_i \textit{ is exclusive with } t_j\}$$

$$TT_{BOD} : \{(t_i, t_j) \in T \times T \mid t_i \textit{ is binding with } t_j\}$$

If $(t_i, t_j) \in TT_{SOD}$, then t_i and t_j cannot be assigned to the same user. If $(t_i, t_j) \in TT_{BOD}$, then t_i and t_j must be assigned to the same user which defines a binding relation between two tasks.

3.4 Model contributions

The main contribution of the TAC model is to specify the task assignment relation where two conditions have to be verified: (1) the first condition is related to task's resources requirements. The role's permissions defined in RPA (role-permission assignment) needs to satisfy the permissions defined in TPA (task-permission assignment). (2) the task is executed if and only if the user/role is assigned to it. Basically, having a permission to execute a task but not being assigned to it will not satisfy the outlined conditions and, therefore, will deny the access to task resources.

Definition 2 (Task Assignment). *A task instance t_i is assigned to a user u with an active subject s if and only if :*
$(t, r) \in TRA \Rightarrow \{p \in P | (t, p) \in TPA\} \subseteq \{p | (r, p) \in RPA\} \bigwedge claimed_{by}(t_i) = s$, where $(SR(s) = r \bigwedge SU(s) = u)$.

The user-task assignment requires the $claimed_{by}$ function. For instance, a task t_i is assigned a set of permissions based on the TPA relation in order to carry out this task. A user u_1 with a role r_j is assigned to t_i if and only if u_1 verifies the TRA and $claimed_{by}$ conditions. However, if we consider another user u_2 member of same role r_j having the same permissions based on the RPA relation but u_2 is not defined in $claimed_{by}(\mathrm{t}_i)$, which means not assigned to this task. In this case, u_2 is not allowed to execute t_i since he does not fulfill the user-task assignment relation (see condition 2).

In the loan process example, let user *Bob* a member of role *Clerk* but not from the same bank agency. Bob is not allowed to perform the task “Check credit” since he is not assigned by the system to execute it. Within organisations, users can share different roles but are not assigned to the same tasks. This is due to privacy and security constraints such as the separation of duty. Therefore, we leverage condition 2 as an additional constraint when claiming a task instance by a user.

In the next section, we leverage the user-task assignment conditions to support task delegation assignment with regards to the delegatees and its required privileges.

4 Access Control Over Delegation using TAC

Delegation is a mechanism that permits a user to assign a subset of his assigned authorisations (privileges) to other users who currently do not possess it.

Definition 3 (Delegation Relation). *We define a delegation relation $DR \subseteq T \times U \times U \times 2^{DC}$ where* T *a set of tasks,* U *a set of users and* DC *a set of delegation constraints. A task delegation relation is defined as* DR = (t,u_1,u_2,{DC})*,* t *is the delegated task and* t ∈ T*, u_1 the delegator and u_2 the delegatee* ∈ U*.*

For instance, delegation constraints (DC) can be related to time or evidence specifications [4]. In addition, organisational constraints regarding roles mapping cross organisations or role hierarchies within an organisation define user-to-user delegation constraints (see RM and RH relations of the TAC model in Fig. 2). For instance, a subordinate in an organisation hierarchy can act on behalf of his superior where the latter is the delegator and the former is the delegatee.

Here, a delegation relation defines the main constraints to be considered when delegating privileges with regards to users/roles, task and resources. Our focus is to integrate such constraints in a secure manner. In doing so, we leverage the TAC (task-oriented access control) model specifications to compute delegatees and privileges. The TAC model allows to compute the list of potential delegatees using the RPA (role-permission assignment) relation that may satisfy the delegated task requirements based on the TPA (task-permission assignment) relation. In doing so, we define a method for access control over task delegation using TAC. In the following, we detail our method and describe how valid delegatees are checked and whether they need delegated privileges grant.

Input: u_1, u_2 ∈ U; r_1, r_2 ∈ R; t_i, t_j ∈ T.

1. Defining the role and permission assignments for each user (URA and RPA);
2. Instantiating the task t_{i1} and assigning it to the delegator s_1 who is the current user u_1;
3. Checking security constraints before delegation (SoD and BoD);
4. Computing the delegatee s_2, who is the current user u_2, based on his permissions assignment ($(t_i, p_{r2}) \in TPA$) or;

5. Granting privileges for s$_2$ based on the task instance permissions assignment ($p'_{r2} \leftarrow p_{r2} \cup p_{ti}$) which is defined in the $claimed_{by}$ function;

   Output: Delegation relation instance : dr$_1$ = (t$_{i1}$,s$_1$,s$_2$,{DC});

The main contribution of this method is to specify the delegated task assignment conditions based on Definition 2. If the two conditions are satisfied, then the task t$_i$ is delegated to the delegatee u$_2$. However, if u$_2$ does not have the permission required and there is no conflicts (BoD or SoD) to execute t$_i$. Then the delegated privileges are granted for u$_2$ based on the $claimed_{by}$ function.

The computation of the privileges is based on the TRA and $claimed_{by}$ specifications defined in our TAC model (see $claimed_{by}$ condition for permissions). Basically, we provide a method to compute the least privileges to delegate based on the current requirements of the task instances t_{i1} which is generated from the delegated task. At this stage, delegated privileges are done manually supporting a user-to-user delegation. However, the administration of new access rights has to be specified later on into authorisation policies in a compliant and dynamic manner. Authorisation policies will regulate how the business process and resources should be managed when delegating a task within a workflow. Delegation policies are not discussed in this paper due to space restrictions.

5 Related Work

Barka et al. proposed a role-based delegation model based on the RBAC model. Their unit of delegation is a role. Authors focused also on role-based models supporting role hierarchies when studying delegation in the context of both RBAC0 model (flat roles) and RBAC1 model (hierarchical roles) of the RBAC96 family [6]. However, users may want to delegate a piece of permission which is not supported in such models. This is the case when computing delegated privileges.

Task-based access control (TBAC) aims to provide a task context during permission assignments [11]. A workflow system consisting of tasks is assumed. Each of these tasks is then assigned a "protection state", providing information as to who gets to have which permission on a task basis. According to the current state of the workflow system moving through the process instance, different permission assignments are activated or deactivated as ordered by the protection state. The TBAC design is process oriented, however, ignoring human-centric interactions such as user-to-user delegation.

Team based access control (TMAC) is an access control scheme similar to RBAC, but it provides the assignment of both users and permissions to teams [12]. Each team then is bound to the task it was created for. At runtime, more than one team can be created out of the same template, but each team will be working on a different task instance and accordingly will need access to different object instances. TMAC model is out of the scope of this paper where we consider constraints on tasks and users rather than a team.

6 Conclusion

In this paper, we integrated task delegation constraints into a formal security model. In doing so, we analysed task authorisation constraints to support security requirements for delegation. We defined a Task-oriented Access Control (TAC) model to support access control over task delegation in workflow systems. Moreover, we presented a method to compute potential delegatees and their delegated privileges. In future work we plan to examine the XACML (eXtensible Access Control Markup Language) standard for the TAC model to support task delegation constraints in particular and authorisation policies in general.

References

1. V. Atluri and J. Warner, "Supporting conditional delegation in secure workflow management systems," in *SACMAT '05: The tenth ACM symposium on Access control models and technologies*, New York, NY, USA, 2005, pp. 49–58.
2. J. Crampton and H. Khambhammettu, "On delegation and workflow execution models," in *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2008, pp. 2137–2144.
3. R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
4. K. Gaaloul, "A Secure Framework for Dynamic Task Delegation in Workflow Management Systems," 2010, Ph.D. thesis, The University of Henri Poincaré, Nancy, France.
5. J. Crampton and H. Khambhammettu, "Delegation in role-based access control," in *Proceedings of the Computer Security - ESORICS 2006, 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006*, ser. Lecture Notes in Computer Science. Springer, 2006, pp. 174–191.
6. E. Barka and R. Sandhu, "Framework for role-based delegation models," in *Proceedings of the 16th Annual Computer Security Applications Conference*. Washington, DC, USA: IEEE Computer Society, 2000, pp. 168–176.
7. X. Zhang, S. Oh, and R. Sandhu, "PBDM: a flexible delegation model in RBAC," in *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*. New York, NY, USA: ACM Press, 2003, pp. 149–157.
8. WFMC, The Workflow Management Coalition, "Workflow Management Coalition Terminology and Glossary," 1999, document Number WFMC-TC-1011.
9. J. Crampton and H. Khambhammettu, "Delegation and satisfiability in workflow systems," in *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2008, pp. 31–40.
10. R. A. Botha and J. H. P. Eloff, "Separation of duties for access control enforcement in workflow environments," *IBM Systems Journal*, vol. 40, no. 3, pp. 666–682, 2001.
11. R. K. Thomas and R. S. Sandhu, "Task-based authorization controls (tbac): A family of models for active and enterprise-oriented autorization management," in *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI*. London, UK, UK: Chapman & Hall, Ltd., 1998, pp. 166–181.
12. R. K. Thomas, "Team-based access control (tmac): a primitive for applying role-based access controls in collaborative environments," in *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*. New York, NY, USA: ACM, 1997, pp. 13–19.