Designing Role Hierarchies for Access Control in Workflow Systems

Reinhardt A. Botha
Faculty of Computer Studies
Port Elizabeth Technikon
Port Elizabeth, South Africa
rbotha@computer.org

Jan H.P. Eloff
Department of Computer Science
Rand Afrikaans University
Johannesburg, South Africa
eloff@rkw.rau.ac.za

Abstract

Due to the correspondence between the role abstraction in Role-based Access Control (RBAC) and the notion of organizational positions, it seems easy to construct role hierarchies. This is, however, a misconception. This paper argues that, in order to reflect the functional requirements, a role hierarchy becomes very complex. In a bid to simplify the design of role hierarchies suitable for the expression of access control requirements in workflow systems, the paper proposes a “typed” role hierarchy. In a “typed” role hierarchy a role is of a specific type. The associations between different types of roles are limited by rules that govern the construction of a role hierarchy. This paper proposes a methodology to systematically construct a “typed” role hierarchy. Since the “typed” nature of the role hierarchy is only relevant during the construction of the role hierarchy, it can seamlessly be integrated into existing RBAC schemes that support the concept of role hierarchies.

1 Introduction

Within organizations individuals fill a specific position. A position in this respect represents a specific “seat” which has certain privileges and accompanying responsibilities. In Role-Based Access Control (RBAC) the concept of a role corresponds to “a job function within the organization that describes the authority and responsibility conferred on a user assigned to the role” [8].

In the RBAC96 model [8] a role hierarchy is a partial order whereby roles are associated with roles. Users associated with superior roles in the role hierarchy inherits the permissions of their inferiors.

Role $r_1$ in Figure 1 therefore inherits the permissions associated with roles $r_2$, $r_3$ and $r_4$. Note that $r_1$ and $r_5$ are not related through the role hierarchy.

A role such as $r_5$ can be called a “private” role with respect to $r_3$ [8] since the superiors of $r_3$ don’t inherit any of its permissions. Private roles may be used in instances where the upward inheritance of permissions is inappropriate.

The association of roles with roles to form a role hierarchy is responsible for much of the powerful features of RBAC. However, it also makes it more difficult to ensure that only the appropriate permissions are assigned to users.

Moffett [5] argues that various different hierarchies exist within an organization. There are, for example, the organizational structure (e.g. departments and business units), generalization hierarchies (is-a relationship – a nurse “is-a” healthcare provider), aggregation hierarchies (financial control consists of Financial Forecasting and Financial Accounts) and supervision hierarchies (managers and supervisors). They show that careful consideration must be given to the inheritance of access rights when an existing hierarchy is used as the role hierarchy.

This paper continues Moffett’s argument in the following way. The next section discusses the access con-

![Figure 1: Example Role Hierarchy](image.png)
control requirements that exist, and, intuitively, are influenced by the role hierarchy. Thereafter it is suggested that the rules in the role hierarchy be “typed”. Several rules that govern the associations between the different types of roles are formulated. Thereafter, a methodology for the systematic construction of the “typed” role hierarchy is presented. A discussion on how (and why) the resultant role hierarchy meets the requirements follows. Finally, the paper concludes by a criticism of the proposed model and the presentation of a further workplan.

2 Access control requirements

Role-based Access control (RBAC) has become popular in recent years. However, ”RBAC is not a panacea for all access control issues” [8]. In workflow systems RBAC may form the basis of access control, however, it does not specifically provide all the features required of an access control service in a workflow environment. A more elaborate access control service needs to be employed to serve the specialized needs of workflow systems. Various authors [1, 3, 4, 5, 9] have identified requirements for access control in workflow systems. For the purpose of this paper, we summarize the requirements as follows: The access control service must

1. support the principle of strict least privilege [3]. This extends the interpretation normally associated with the principle of least privilege. According to the least privilege principle, a user receives only the permissions required to perform his job. Strict least privilege recognizes that, in performing a job, several tasks will be performed. The principle of strict least privilege limits, therefore, the permissions granted to a user to the immediate task at hand.

2. support the delegation of authority [5]. With delegation of authority, the delegator abrogate his immediate permission to perform an activity to another person. However, the delegator does not lose the capability to withdraw the delegation.

3. reflect the reporting structure [2, 4, 5]. The reporting structure reflects the control principles of supervision and review. Note that this does not necessarily relate to the relations in the organizational structure. For example, although a “junior programmer” may be reflected in the organization chart as being inferior to a “senior programmer”, this does not necessarily mean that a user fulfilling the “junior programmer” role reports to (or is being supervised by) a “senior programmer”.

4. allow for slight differences between positions. Positions in this respect reflect “seats” of the same role. Different positions of the same role may, for example, not report to the same person. A similar situation may arise after authority has been delegated to a specific user or position.

5. enforce separation of duty control principles. These principles are primarily concerned with ensuring the semantic integrity of information. Separation of duty principles may be enforced either at administration time or during runtime. Administration-time enforcement is concerned with preventing situations that should never occur. For example, a financial manager should never be allowed to be an auditor as well. Run-time enforcement is concerned with ensuring that permissions are not misused. An example of such case would be “a manager may not approve his own purchase order”.

This paper will show how these access control requirements are supported by the typed role hierarchy that results from the proposed methodology. Next, the typed role hierarchy are defined.

3 The typed role hierarchy

This paper adopts the RBAC96 model [8] for defining the foundational concepts of RBAC. The power of RBAC is vested in the associations between the sets of users, permissions and roles (U, P and R, respectively). Users and roles are associated through the user-assignment relation, UA ⊆ U × R. Roles and permissions are associated through the permission-assignment relation, PA ⊆ P × R. Roles are related through the role hierarchy partial order, RH ⊆ R × R. For (r1, r2) ∈ RH, we write r1 ⊳ r2, that is r1 dominates r2 in the partial order. For notational convenience the ≤ symbol is defined to indicate the inverse relation, is-dominated-by. Therefore, stating r2 ≤ r1 is equivalent to stating r1 ≥ r2. The expressions “r1 is superior to r2” and “r2 is inferior to r1” are considered synonyms for the dominates and is-being-dominated relations, respectively.

In the proposed model, the principles behind the role hierarchy and its operation, as defined in [8], stay intact. The ”typed” role hierarchy can thus be seen as an enhancement for constructing the role hierarchy. It attempts in no way to redefine the influence of the role hierarchy on the enforcement of access control.

A typed role hierarchy consists of different types of roles that are associated with each other. A distinction is made between organizational roles, task roles,
The top-most virtual role: $r_{\text{topmost}}$

The bottommost virtual role: $r_{\text{bottommost}}$

Roles for organization breakdown: $R_{\text{org}}$

Roles for tasks: $R_{\text{task}}$

Private roles: $R_{\text{priv}}$

Roles for specific positions: $R_{\text{pos}}$

Roles for broader jobs: $R_{\text{job}}$

Role A

dominates /is superior to

Role B

Figure 2: Graphical Notation for typed roles and hierarchy

private roles, positional roles and job roles. Virtual roles are introduced to ensure that a single role hierarchy can be designed for an organization. Figure 3 provides a graphical notation for each of these roles.

The various types of roles are defined as follows:

Definition 1 (Typed role set). The typed role set is the set of roles, $R = R_{\text{virt}} \cup R_{\text{org}} \cup R_{\text{job}} \cup R_{\text{pos}} \cup R_{\text{priv}} \cup R_{\text{task}}$, where

- $R_{\text{virt}}$ represents roles that are virtual in the sense that it is artificially created to ensure that only one hierarchy exist. This allows for the use of standard graph traversal algorithms when maintaining the role hierarchy [6]. In particular, we define \( \{r_{\text{topmost}}, r_{\text{bottommost}}\} \subseteq R_{\text{virt}} \) where $r_{\text{topmost}}$ is an artificial highest role, i.e. $\forall x \in R : r_{\text{topmost}} \succeq r_x$, and $r_{\text{bottommost}}$ is an artificial lowest role, i.e. $\forall y \in R : r_{\text{bottommost}} \preceq r_y$.

- $R_{\text{org}}$ represents roles that relate to the organizational chart. This typically reflects the functional decomposition (a vertical partitioning) of the organization [2]. This could refer to areas such as sales, finance or engineering.

- $R_{\text{job}}$ represents roles that broadly define a persons job. Examples can be found in general terminology, such as secretary, manager or accountants.

- $R_{\text{pos}}$ represents roles that relate to specific organizational positions. These roles are more specific, for example, financial manager or accounts-receivable manager.

- $R_{\text{priv}}$ represents private roles, i.e. roles where the permissions are not inherited upwards. These will largely correspond to the $R_{\text{pos}}$ and $R_{\text{job}}$ roles.

However their names would reflect the private nature of the roles. These roles will be used to prevent upward inheritance of certain permissions.

- $R_{\text{task}}$ represents the roles that relate to specific tasks, the building block of organizational workflows. These will represent work that requires to be done in one chunk. It could relate, for example, to tasks such as “approve order” or “grant loan extension”. It should be recognized that not all tasks in an organization is workflow related [7].

In this paper we are only interested in workflow-based tasks.

To form a typed role hierarchy, roles must be associated with roles according to a partial order. If an administrator attempts to associate two roles, their respective types will determine whether an association is allowed. The role hierarchy is thus defined as follows:

Definition 2 (Typed role hierarchy). A typed role hierarchy is defined to be the partial order $RH \subseteq R \times R$, $R$ being a typed role set. The tuples in $RH$ are restricted in the following way:

\[
(r_i, r_j) \in RH \iff \\
[r_i = r_{\text{topmost}} \land r_j \in (R_{\text{pos}} \cup R_{\text{priv}})] \lor \\
[r_i \in R_{\text{org}} \land r_j \in (R_{\text{org}} \cup R_{\text{pos}})] \lor \\
[r_i \in R_{\text{pos}} \land r_j \in (R_{\text{pos}} \cup R_{\text{task}} \cup R_{\text{job}})] \lor \\
[r_i \in R_{\text{job}} \land r_j \in (R_{\text{job}} \cup R_{\text{task}} \cup \{r_{\text{bottommost}}\})] \lor \\
[r_i \in R_{\text{task}} \land r_j = r_{\text{bottommost}}]
\]

We now consider how the named role hierarchy can be constructed to meet the requirements of mentioned in Section 2.

4 Creating the hierarchy

The creation of the named role hierarchy is best described through a methodology approach. The steps of the methodology are described below. At appropriate places in the discussion extracts from role hierarchies are given, subscribing to the graphical notation defined in Figure 3.

Step 1 (Vertical organization partitioning: $R_{\text{org}}$). Vertical partitioning happens according to the organizational view [10]. The organization is partitioned according to how it is typically broken into units used for human resource allocation, typically departments. Business functions are often viewed in terms of these units’ (e.g. departments) responsibilities. The chosen units may be grouped together using higher level units. For example, departments may be grouped
into divisions or groups. Figure 4 depicts an organization broken up into 3 departments: Finance, Engineering and Sales. If applicable, the departments are linked to bigger groupings. The highest levels are all linked to the topmost role $r_{\text{topmost}}$.

**Step 2 (Horizontal partitioning: $R_{\text{job}}$).** Each vertical partition could be partitioned horizontally according to the types of jobs (the typical RBAC notion of role) that may be observed. These roles may be related to each other through a generalization relationship. The bottommost role represents the most general role in the organization. Figure 4 shows the finance department horizontally partitioned into managers, accountants and secretaries. They are all shown to be a generalization of “financial employee” which is an “employee”. Furthermore the bottommost role, called “person” in this case is defined and linked to employee.

**Step 3 (Define organization positions: $R_{\text{pos}}$).** For each role there may be many different people that fulfill that role. For example, there may be one role “secretary”, but the organization may employ numerous secretaries. During this step the different positions need to be created as roles. Note that in this step a certain level of judgement regarding the appropriateness of distinguishing between roles that are positions and roles that are types of jobs is required. It is only necessary to create roles that represent a position if the access control requirements for the different roles are going to be different. Cases where this might be included are where the permissions required in the positions are slightly different or the people report to different supervisors. Position roles are superior to the roles (from the vertical partition step) to which they correspond. Position roles can also be linked inferior to the appropriate roles created in the horizontal partition step.

Furthermore, position roles may be linked according to specific reporting structure.

**Step 4 (Define private roles: $R_{\text{priv}}$).** Each job and position may require a private role. Permissions that are not appropriate for upward inheritance will be associated with these private roles. Private roles are superior to their corresponding position or job roles in the role hierarchy.

**Step 5 (Define task based roles: $R_{\text{task}}$).** Analysis of the workflows within an organization will yield the tasks to be done. Task-based roles are created and linked inferior to the “lowest” possible role that may perform the task. If the permission assigned to the task role should be inherited upward to all the superior roles, the task role should be linked to a job role. If the permissions to do a task should only be inherited by supervisors, the task should be linked to a positional role. Finally, if the task should only be done by a specific role, it should be linked to a private role of the jobs role.

The created task based role must be linked as superior to the bottommost role $t_{\text{bottommost}}$.

The result of linking the typed roles in a hierarchy is that specialization chains will emanate from each role. Figure 4 shows such a specialization chain starting at the $t_{\text{manager}}$ job role.

This concludes the creation of the typed role hierarchy. The access control administration task is, however, not finished until users are assigned to roles, and permissions are assigned to roles. The proposed methodology includes, therefore, the allocation of users and permissions as Steps 6 and 7. These steps are described below.
Step 6 (Assign users). Users are assigned to the highest possible role \( r_i \) where \( r_i \in R_{job} \cup R_{pos} \cup R_{priv} \) according to the following rules:

\[
(u_x, r_i) \in UA, r_i \in R_{job} \Rightarrow \exists r_j \in R_{pos} \cup R_{priv} : r_j \preceq r_i
\]

\[
(u_x, r_i) \in UA, r_i \in R_{pos} \Rightarrow \exists r_j \in R_{priv} : r_j \preceq r_i.
\]

Step 7 (Assign permissions). A permission is assigned to the lowest role that require that permission. A permission may be assigned to more than one role, provided that the only common senior role these roles have, is the topmost role. The following heuristics may be used when assigning the permission to a role.

- Permissions related to work that is governed by a workflow system, must be assigned to the specific task roles.
- A permission that should not be freely inherited by superiors, must be assigned to a private or positional role (\( \in R_{priv} \cup R_{pos} \)).
- If inheritance of a permission should only be based on immediate reporting structure it must be assigned to a positional role (\( \in R_{pos} \)).
- If inheritance of a permission should not occur at all it must be assigned to a private role (\( \in R_{priv} \)).

This step requires a careful consideration of the desired access control result. However, as the discussion that follows will show, the roles that will be available at this stage are sufficient to address the requirements specified in Section 2.

5 Discussion

It is clear that, even in the case of a fairly small organization, the role hierarchy would be fairly large and complex. Let us consider how the access control requirements with respect to workflow systems, as specified in Section 2, are met when the described methodology are applied.

**Strict least privilege** Within the workflow definition the tasks will be associated with task roles. The fact that a user belongs to a superior role is a necessary condition for the workflow engine to assign the task to the user. However, the permissions granted to the user will only be determined by the task role associated with the task, not the roles that the user may assume. Note that this requires a constraint to be placed on the session as defined in the RBAC96 model [3].

Since task roles can be associated with positional roles, it is possible to specify an upper limit for the inheritance. Consider Figure 5. The “evaluate manager” permission will only be received by users belonging to the clerk and accountant belonging to the respective specialization chains.

**Delegation of authority** The existence of private roles makes it possible for a user to delegate authority to a specific individual or a specific position in the organization. The task role(s) that corresponds authority that is being delegated must be disassociated with the current role and a “is-inferior-to” association with the relevant positional or private role must be made. At the same time an administration permission that allows the revocation of the assigned tasks must be associated with the private role of the delegator.

Figure 6 shows a portion of a role hierarchy where the \( r_{pos1} \) role delegated task \( t_{task_2} \) to role \( r_{2priv} \), while task \( t_{task_9} \) was delegated to the private role of a specific position, \( r_{2pos1priv} \).

**Reporting structures** The association of positional roles with one another allows for defining a “supervision” relation. This is based on the assumption that a supervisor must be able to do the job of the person being supervised. If there is anything that should not be done by the supervisor, a private role could be used.
Differences between positions  Positional roles inherit permissions from their corresponding job role. The different permissions required for the actual positions can then be associated with the positional role.

Separation of duty  Since the functionality behind a role hierarchy is not changed at all from the meaning attached by, for example [8] and [6] current work on separation of duty stays intact. The design approach ensures, furthermore, that task roles are available. In many instances the definition of separation of duty constraints may be more intuitive across smaller groupings of permissions, such as task roles, rather than job roles.

6 Conclusion

This paper supported the observations by Moffett [5] regarding the complexity of constructing a role hierarchy that exhibits the appropriate inheritance features. The complexity of the problem was reduced by introducing the concept of a “typed” role and, subsequently, a “typed” role hierarchy. The suggestions given do not alter the basic concept of a role hierarchy. To the contrary, the resultant role hierarchy stays a role hierarchy in the normal sense of the word [8, 6].

The main advantage of the suggested “typed” role hierarchy is that the construction and maintenance activities are eased. The easier way of doing is primarily contributed to the more rigid rules associated with building the hierarchy. Less associations are left to interpretation by the security administrator, making it easier to verify the results of a change in the role hierarchy.

However, without the appropriate tools to support the methodology, the design of role hierarchies will remain a rather unfriendly and complex activity. The development of tools to aid in the construction of the role hierarchy will thus receive attention in future work. Once such tools exist, the methodology could be applied in real world test cases to determine its ability to ease the design of effective role hierarchies.

References