Implementation of Hierarchical Authorization For A Web Based Digital Library

Andreas GEYER-SCHULZ
Informationsdienste und elektronische Märkte, Universität Karlsruhe (TH)
76128 Karlsruhe, Germany

and

Anke THEDE
Informationsdienste und elektronische Märkte, Universität Karlsruhe (TH)
76128 Karlsruhe, Germany

ABSTRACT

Access control mechanisms are needed in almost every system nowadays to control what kind of access each user has to which resources and when. On the one hand access control systems need to be flexible to allow the definition of the access rules that are actually needed. But they must also be easy to administrate to prevent rules from being in place without the administrator realizing it. This is particularly difficult for systems such as a digital library that requires fine-grained access rules specifying access control at a document level. We present the implementation and architecture of a system that allows definition of access rights down to the single document and user level. We use hierarchies on users and roles, hierachies on access rights and hierarchies on documents and document groups. These hierarchies allow a maximum of flexibility and still keep the system easy enough to administrate. Our access control system supports positive as well as negative permissions.

Keywords: role-based authorization, digital library, hierarchies

1 INTRODUCTION

Offering personalized access to web applications to a broad user community is a crucial element for implementing a variety of services, such as electronic commerce applications, telecommunication services or digital libraries. Personalizing a web site requires a mechanism to establish who a user is (authentication) and what this user is allowed to do (authorization). The choice of architectures and data models for these procedures is very important because user administration involves a considerable amount of transaction costs.

Authorization issues are particularly interesting for digital or hybrid libraries where access to a large number of documents must be handled. An authorization scheme that is able to handle a collection of very heterogenous documents as well as different user groups with very specific access rights has to be very flexible. Yet the system must allow the system administrator a clear and comprehensible overview of the current set of permissions. An authorization system is useless if it is not possible to have full understanding of and control over the complete set of access definitions. Such a system could not be considered secure. This is the reason why most of the authorization systems in use only allow for very basic definitions of access rules.
In this paper, we present the structure, architecture and implementation of an authorization system with a three-dimensional hierarchy that allows for very flexible and fine-grained access definitions. The system is in use for an institutional hybrid library with many different types of content and user groups but is applicable to very general types of applications requiring access control. We show how the hierarchies serve to combine utmost flexibility with the structures for comprehensible and manageable review functionalities that support system administration. We explain the transformation of the hierarchies into a canonical form of authorization information which is needed to establish the concrete access rights for given resources and we give an overview of the data model supporting the fast check of access rights.

Sec. 2 starts with a presentation of other access control mechanisms that were proposed for digital libraries. In sec. 3 we explain the structure of the system including the hierarchies and propagation rules. Sec. 4 describes the architecture of our implementation while sec. 5 presents the most important user interfaces. The last section concludes our work and proposes directions for further research.

2 ACCESS CONTROL MECHANISMS

Several general models for organizing access control have been proposed. Discretionary Access Control (DAC) [12] uses a matrix format to record permissions for each user on each resource. Depending on the implementation of the matrix as capability list or access control list permissions can either be easily retrieved by user or by a given resource. The method is very simple but on the other hand every single permission has to be specified separately thus imposing high transaction cost on system administration. DAC does not allow for any grouping of similar users or resources.

Role-based Access Control (RBAC) is a class of models that introduce user roles for facilitating the definition and modification of permissions [9, 4]. A user role usually corresponds to a position in an organization's hierarchy. Permissions are granted to user roles instead of individual users and users are allowed to play a certain number of roles corresponding to their position in the organization. A permission in RBAC is a combination of a certain privilege and a resource. Roles may be organized in a hierarchy corresponding to the hierarchy in an organization. Permissions propagate up the hierarchy where the most powerful position is found on top of the hierarchy. Privileges and resources are not structured in RBAC, neither does RBAC explicitly model negative permissions.

RBAC proposes constraints for enforcing separation of duty where users are not allowed to play conflicting roles at the same time. Those are typically roles that would aggregate too much power for one user or which violate data protection laws. Other proposed extensions to RBAC include the addition of a temporal dimension to the constraints that allows the possibility of role activation to be time-dependent for certain users [5].

Our model borrows some ideas from RBAC, mainly the introduction of user roles and their organization into a hierarchy along which permissions are propagated. Still, the interpretation of our role hierarchy differs from that of RBAC, as for our application we use a hierarchy with a "contains" semantic that organizes users into supergroups and subgroups. As we show later, this has an important implication on the propagation of permissions.

In [2] a model is sketched that organizes documents in a digital library into a hierarchical structure representing collections and subcollections of documents. Permissions are inherited along this structure. The authors also give some examples of permissions that imply other permissions, but they do not consider a hierarchical organization of the privileges as a consequence, as we do. The notion of collections and subcollections seems a bit less flexible than our very general formulation of a hierarchy induced by a partial order. We do not make any limitations on the type of the hierarchy elements although collections of documents are a very good example. Our hierarchy also allows the modelling of document parts and their grouping into other categories than the document itself.

The models most similar to our proposal are probably the ones proposed in [1, 7, 3, 10]. These authors describe a content-based access control mechanism. Authorization to users is granted based on credentials that represent certain user attributes like age, nationality etc. Credential types can be organized into a hierarchy with permission propagation. A very similar procedure is applied to the documents that can be assigned different concepts. Concepts, too, are hierarchically structured. Concepts are content-related and are extracted from the documents automatically, as well as the hierarchical structure of the concepts. Permissions can be granted for documents as well as parts of documents or links between different documents.

The advantage of this approach is that explicit creation and maintenance of roles is no longer necessary. On the other hand, groupings of users that are not specified via any concrete user attribute are not possible. Neither do the authors make any concrete statement on how users are restricted from changing their attribute values, thereby altering their access permissions. A similar reasoning can be applied to the concepts of documents. It is very practical to have concepts and their structure generated automatically because it shifts the burden of assigning concepts or catalogues to documents or document parts to the classification process. On the other hand it is not possible to generate a group of documents based e.g. on the document structure or any other, not content-related, characteristic. For example, one might wish to group documents that are used by a group of researchers for a special project. The documents can be of very different types, like session minutes, papers, CVs etc. such that it will be difficult to find the concepts that identify all these documents as part of the project. Still, the notion of concepts is similar to the document catalogues we use for grouping the elements of our digital library.

Similar to the papers cited above, we also allow permissions for document parts. Although, as we model this through different resource types that can be linked through the hierarchy, our approach is more flexible. It is not per se restricted to certain attributes such as document parts or links but can be applied to other sort of document-related information such as reviews, ratings, questionnaires, recommendations and many more (cf. Fig. 3 in sec. 3).

In the works cited above, negative permissions are modelled as well as positive permissions. The conflict resolution strategy favors permissions directly set for certain users or objects before those permissions attributed to users or objects via their credentials or concepts. In any other conflict situation, negative permissions prevail. The conflict resolution strategy in our system does not distinguish different kinds of permission settings, all permissions are treated in the same way. We let explicit negative permissions always override positive permissions. We believe that this is the most secure strategy and easiest to overview for the system administrator who does not have to care about different types of permissions.

Other, content-based authorization models are proposed in [11] and [8]. While [11] describes a combination of DAC, mandatory access control and RBAC for content-based authorization of multimedia data, [8] focuses on federated digital libraries and an authorization method for controlling access at the level of the central search interface. The authorizations are based on keywords found in the documents, which is somehow similar to the concepts described above. These authors do not use any hierarchical organisation of their authorization data.

A theoretical work that defines many of the concepts used in this work is [6]. A general logical framework is developed that should be able to accomodate all possible access control models and serves as a basis for comparison of different models. In fact, the framework deals with hierarchical relationships in subjects, privileges and objects that include the relationships that we use for our system. Nevertheless, the work does not make any conclusion on how useful the hierarchical relationships are for practical implementations and just serves as the most general model that is able to describe other existing access control models or implementations.

3 THREE HIERARCHIES FOR FLEXIBLE ACCESS CONTROL

The major goal of our access control system is highest flexibility and generality combined with ease of specification. Flexibility means that we should be able to grant or deny access permissions to single documents and even to document parts so that all possible forms of access restrictions can be represented. The system should also be as general as possible in order to transfer the methods to other kinds of applications. The permissions and structures should not be explicitly tailored to the special requirements of digital libraries but accomodate other applications and their authorization needs as well.

The simplest canonical access control system is defined as a relation $R_S \subseteq S \times P \times O$ where S denotes the set of subjects, P the set of privileges and O the set of objects. The triple $\langle s, p, o\rangle$ is interpreted as subject $s \in S$ may perform privilege $p \in P$ on object $o \in O$ [10]. Note that in these systems only positive permissions are specified explicitly, access denial is inferred by an absence of the positive permission. In this paper, the canonical access model is defined as a relation $R \subseteq S \times P \times O \times G$ with S, P, O as defined before and $G = \{+, -\}$. The quadruple $\langle s, p, o, +\rangle$ is then defined as a positive permission as before. However, access denials are treated differently. In this model we distinguish between denials because of a lacking positive right and explicitly defined denials of the form $\langle s, p, o, -\rangle$ which always override conflicting positive permissions. This form of modelling allows expressing denials as imperatives for the future with the aim of permanently excluding users and serving as a signal that a user is prohibited to perform p on o.

To facilitate the task of specifying and maintaining access control definitions, we introduce hierarchies on subjects, privileges, and objects. Each hierarchy is defined as a partial order on its elements.

For the set of subjects S, the partial order is defined on the power set $\mathcal{P}(S)$. For elements $s_i, s_j \in \mathcal{P}(S)$ $s_i \prec_{\mathcal{P}(S)} s_j$ means that s_j contains s_i ($s_i \subseteq s_j$). The names of elements of $\mathcal{P}(S)$ reflect the semantics of the access control system. The set of names of elements of $\mathcal{P}(S)$ is denoted by S_N. For example, a set $s_1 \in \mathcal{P}(S)$

may be associated with the name "technical staff". In general, for any set X, the notation X_N is used for the set of names. The symbol $* \in X_N$ denotes the supremum of the set X. The relation between elements and their names is expressed by the bijection $\nu_X : X_N \rightarrow X$. Furthermore, each hierarchy on a set X comes with two functions, namely $su : X \rightarrow \mathcal{P}(X)$ and $pr : X \rightarrow \mathcal{P}(X)$. $su(x)$ computes $\{y : y \prec_X x\}$ the set of all subsets of $x \in X$, and correspondingly $pr(x) = \{y : y \succ_X x\}$ is the set of all supersets of $x \in X$.

Elements of S are individual users, such that the elements of $\mathcal{P}(S)$ correspond to user groups. In the context of a university's digital library, we prefer user groups over roles since the identification of roles with organizational positions is overly restrictive in a university setting. For example, the user group with the name "all employees" may contain the user group "scientific staff", and "scientific staff" may contain "user John".

For the set of privileges, the partial order $\prec_p$ is directly defined on the set P and not on the power set of P. For $p_i, p_j \in P$, $p_i \prec_P p_j$ means that privilege p_j implies privilege p_i, that is a user who is allowed to perform p_j is also allowed to perform p_i. The semantics that this hierarchy represents has been proposed earlier and proved to make sense for privileges [13, 6]. An example would be a "write" permission on a document that implies the "read" permission on it, because when editing a document its data are visible to the user in any case. P_N denotes the set of names for privileges in P.

For the set of objects O, the hierarchy is defined on the power set $\mathcal{P}(O)$. The semantic of pr_O corresponds to the semantic of pr_S. For $o_i, o_j \in \mathcal{P}(O)$, $o_i \prec_O o_j$ means that o_j contains o_i ($o_i \subseteq o_j$). The set of names of elements of $\mathcal{P}(O)$ is O_N. In general, the set of objects contains all atomic resources that require access control protection. An atomic resource is one that cannot be further partioned into smaller, differentiated parts. An example may be an attribute or a file of a digital library document. A document itself may then be represented by the element in $\mathcal{P}(O)$ that contains all the single parts of this document. Elements grouping several documents into one set are called "categories". There are no restrictions on the kind of objects and object groups that an application works with. The interrelations between these elements may be of different semantic type. For example, one element in the power set may be a category "digital library publications" that groups together documents that are related by content as they deal with the same topic. People of a digital library research group may then be granted access to all documents contained in this category. Another category may be called "internal institute documents". This category does not imply any similarity in content of the documents contained but groups documents with similar access control requirements as only very privileged users may access those internal documents.

Of course, many different object types may be grouped together in one access control system. As different and independent applications use the same access control system each application may define its own objects and hierarchies which need not be interrelated. Besides a digital library a contact management system may need restriction on data about persons and organizations. There will probably be no order relations between any two objects of the different applications but this does not affect the access control system.

These hierarchies help to reduce the system complexity as access definitions propagate along the partial orders. This considerably reduces the amount of access definitions that need to be created and maintained. Another advantage is the transparency and ease of use for the end user. If a user inserts a new publication on digital libraries into the system he will group it into the corresponding category just for the sake of correct categorization without having to be aware of the consequences for the document's privileges. Practical experiences have shown that users do not care much about privilege management of documents. Users can directly understand and profit from grouping their documents into categories because this facilitates retrieval and organisation of document groups. But, on the contrary, as owners of a document often have most of the privileges because of their owner status they are not aware of the consequences of setting privileges on documents unless they are in direct contact with other users that require certain access permissions. Therefore it seems very important to us to make the privilege management very easy and quick to use without requiring extra actions by the users.

Now how exactly do permissions propagate along the hierarchies? And how are conflicts between different access definitions resolved? Permissions are granted or revoked by specifying quadruples of elements of $S_N \times P_N \times O_N \times G$, called access specifications. Each quadruple has then to be translated into the corresponding set of canonical access definitions. For this purpose, we define a translation function $\mathcal{T} : S_N \times P_N \times O_N \times G \rightarrow \mathcal{P}(S \times P \times O \times G)$ that maps the specifications to canonical access tuples. The translation function describes the propagation of access specifications along the hierarchical structures. Let $a = \langle a_S, a_P, a_O, a_G \rangle$ be an access specification, then

$$\mathcal{T}(a) = \begin{cases} su_S(\nu(a_S)) \times su_P(\nu(a_P)) \times su_O(\nu(a_O)) \times + \\ \hfill \text{if } a_G = + \\ su_S(\nu(a_S)) \times pr_P(\nu(a_P)) \times su_O(\nu(a_O)) \times - \\ \hfill \text{if } a_G = - \end{cases}$$

The definition of the translation function shows that there is a difference in the sense of propagation for positive and negative access specifications. In fact, the propagation sense stays the same for the hierarchies on the subjects and on the objects but is reversed for the hierarchy of privileges. This is due to the relationship expressed by the partial order $\prec_P$. For $p_i \prec_P p_j$ p_j implies p_i. So if privilege p_j is granted, then p_i is granted, too. On the other hand, if privilege p_i is denied, then the even stronger privilege p_j should also be denied.

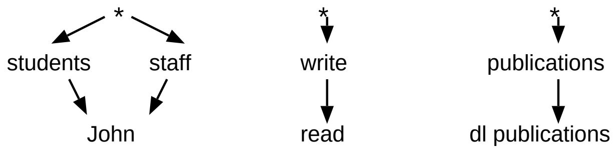


Figure 1: Example hierarchies of subjects (left), privileges (middle) and objects (right)

To illustrate this, consider the hierarchies on subjects, permissions and objects depicted in Fig. 1, where an arrow from a to b means $a \succ b$. Consider further the access specifications ⟨staff,write,publications,+⟩ and ⟨students,read,dl publications,−⟩. This situation occurs for example if the professor wants to give a seminar on digital libraries and requires the students to do some literature research. Corresponding publications are hidden from the students to force them to do a search from external sources. In this case, of course, we expect that the student worker John, who is eligible for the seminar as well, is not granted neither read nor write access to the dl publications but he may still have the staff read and write access to the remaining documents. Negative permissions are propagated up the privilege hierarchy but down the other two hierarchies. In fact, [6] propose the same propagation rules on groups, privileges and objects with corresponding hierarchical relationships.

$cA : S \times P \times O \rightarrow \{true, false\}$ is the function for checking access that returns whether access is granted for a given subject, permission and object. For $a = \langle s, p, o\rangle$, $cA(a)$ is defined as:

$$cA(a) = \begin{cases} true & \text{if } \exists\langle s', p', o'\rangle \in S_N \times P_N \times O_N : \\ & \langle s, p, o, +\rangle \in \mathcal{T}(\langle s', p', o', +\rangle) \\ & \text{and } \nexists\langle \hat{s}, \hat{p}, \hat{o}\rangle \in S_N \times P_N \times O_N : \\ & \langle s, p, o, -\rangle \in \mathcal{T}(\langle \hat{s}, \hat{p}, \hat{o}, -\rangle) \\ false & \text{otherwise} \end{cases}$$

Note that $x \prec x$ holds for a partial order. The function returns $true$ if the given canonical triple of subject, privilege and object can be derived from a positive access specification using the translation function and if there is no explicit access denial specification that also translates to the given subject, privilege and object. This function definition expresses resolution of conflicts by precedence of explicit access denials over access permissions, like already mentioned in this section. We believe that the basic set of access definitions needed for everyday use can be and mostly is structured such that only positive permissions are defined. This possibility depends on a suitable definition of the three hierarchies and their elements. Negative permissions are often very useful for exceptional situations that require an immediate action.

We finish this section by giving some motivating examples of hierarchy structures that are useful for a scientific digital library possibly combined with other types of applications. Figures 2 and 3 show hierarchies for users and groups, and for privileges and resources of different types, respectively. The user-group hierarchy in Fig. 2 shows possible groups for a university institute where students are represented as well as institute staff. The hierarchy shows that the administrators for the web site can be people from the scientific staff as well as the technical staff.

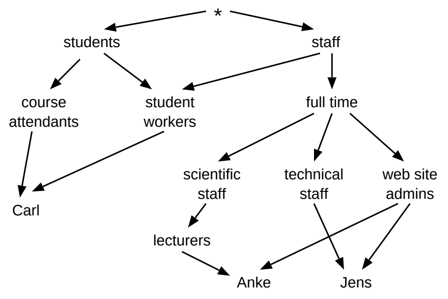


Figure 2: Example hierarchy of groups and users (capital letters)

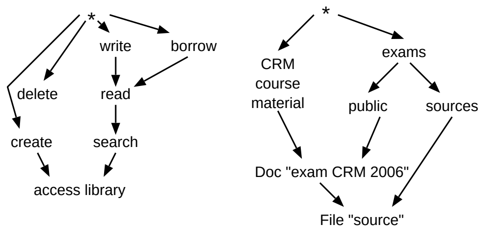


Figure 3: Example hierarchies of privileges (left) and resources (right – catalogues, a document and a document part)

Fig. 3 shows a privilege hierarchy on the left and a resource hierarchy on the right. The privilege to write a document in the digital library is the most powerful privilege as it implies many other privileges. In fact, write implies delete, as someone who can change the contents of a document can also erase all its contents which is similar to deletion. If someone can borrow a document (hybrid library) he must be able to have its contents displayed (read) and consequently to view this document as part of search results (search). The resource hierarchy contains four document catalogues, one document marked "Doc" and one of its files marked "File". The structure shown is useful for documents containing past exams for a given course. Students are given the privilege to read data contained in the catalogue "public" while they are prohibited download access for "sources". So the document may contain one exam version in PDF for viewing its contents and one version containing the sources that is not disclosed to the course attendants because explicit negative rights override positive permissions.

4 SYSTEM ARCHITECTURE AND IMPLEMENTATION

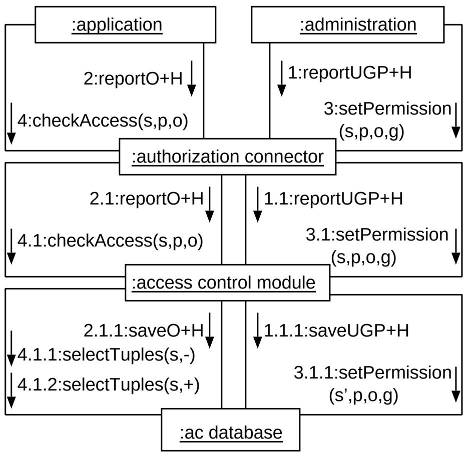


Figure 4: System components and architecture

The components of the access control system and their architecture are depicted in Fig. 4. The figure shows an example application and the administration as a special application that handles the management of users, groups and privileges. The authorization connector serves to abstract from the concrete authorization method in use, whenever it needs to be modified or exchanged against another authorization module the interface between the authorization connector and the applications remains unchanged. The access control module is the layer below the authorization connector and serves as the interface to the access control method described in this paper. The access control module accesses the access control (ac) database containing all relevant data about the subjects, objects, privileges, their hierarchical structures and the permission specifications.

Fig. 4 shows the basic actions that are required to make the access control operational. For initialization of the data, the administration and the application have to deliver information about the subjects, privileges and objects. User, group and privilege management are done via the administration with restricted access. The functions *reportUGP+H* and *saveUGP+H* carry information about the users, groups and privileges plus the corresponding hierarchical structures. This information is saved in the access control database. Similarly, each application must provide information about its objects or resources that require access control and their hierarchical structure using the function *reportO+H*. Each resource is described by a resource type and a resource identifier. Within a resource type, the resource identifier is unique. There is no restriction on the data type of the resource identifiers, it may be numbers as well as strings or other data types. Resource types are coded using integer numbers. The distinction in resource types and identifiers allows each application to maintain its own set of unique identifiers for its objects, without interfering with different resources from other applications.

As soon as the information about the subjects, privileges and objects are successfully reported to the access control module permissions can be specified. This, too, is done by the administration. The definition of permissions by normal users for delegation purposes is not yet included in our implementation. A permission is defined by the subject *s* (user or group), the privilege *p*, the resource *o* and a sign *g* that marks positive permissions and access denials. In our current implementation, we use redundancy on the database level to improve the speed of access checks. This means that, given the new permission specification tuple $\langle s, p, o, g \rangle$, the function *3.1.1:setPermission(s',p,o,g)* is repeatedly called for every $s' \in S : s' \prec_S s$. Each function call produces a database entry. As the sense of propagation for the subject hierarchy is the same for positive and negative permissions no such distinction is needed here.

Right now we only use redundancy at the subject level, but further redundancy can be considered if the system is too slow. On the other hand, redundant data entries raise the update complexity. Updates have to be made atomic operations e.g. by using transactions to ensure the data consistency. Therefore we restrict the redundancy to one of the three hierarchies to keep the update complexity at a lower level.

Algorithm 1 checkAccess function

```
Require: subject s, privilege p, object o;
 1: set_n := select tuples from database with subject s and sign -
    (4.1.1 in Fig. 4);
 2: set_p := select tuples from database with subject s and sign +
    (4.1.2 in Fig. 4);
 3: if set_p is empty then
 4:     return false;
 5: end if;
 6: for each element $\langle \hat{s}, \hat{p}, \hat{o}, - \rangle$ in set_n do
 7:     if $\hat{p} \prec_P p$ and $\hat{o} \succ_O o$ then
 8:         return FALSE;
 9:     end if;
10: end for;
11: for each element $\langle \hat{s}, \hat{p}, \hat{o}, + \rangle$ in set_p do
12:     if $\hat{p} \succ_P p$ and $\hat{o} \succ_O o$ then
13:         return TRUE;
14:     end if;
15: end for;
16: return FALSE;
```

Finally, applications call the *checkAccess* function that returns *true* if the corresponding access can be permitted or *false* if it is denied. In the current implementation with redundancy only over the subject hierarchy the *checkAccess* function has to loop over all possible privileges and objects that may propagate a permission to the privilege and object requested. The steps taken are described in Alg. 1.

The component architecture shows that each application takes care of the resources it needs and for which it wants access control to be set up. This allows for quick extension of the access control over new applications and does not impose any restrictions on the types of resources or privileges in use. There are, however, cases in which it is useful to grant a permission either on the whole set of resources of a special type or not linked to a specific resource at all. For example, the "create" privilege for a digital library cannot be linked to any concrete document as it concerns the creation of new objects. This privilege can only be associated with the type of resource for which creation is granted. To implement this possibility the access management creates, for each type of resource known to it, one special resource of this type without an identifier, called the "any" resource, which is used for privileges concerning only the type of a resource. The "any" resource is not linked to the other resources by any hierarchical link. A second special resource is created for each resource type, the "all" resource. This resource is made the supremum of all other resources of the same type (except the "any" resource). It is used for such cases in which a privilege has to be granted to all resources of a specific type. For example, for a digital library administrator it is useful to always have the permission to read and write all digital library objects. He is then granted the write privilege on the "all" resource which, by the hierarchical relationships, is propagated to every other resource of this type, even newly created ones (the read-privilege is included, as write normally implies read). The "all" resource, just like the "any" resource, is an internal element of the access management that is not visible to the applications.

5 USER INTERFACE

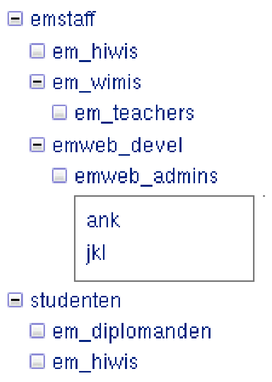
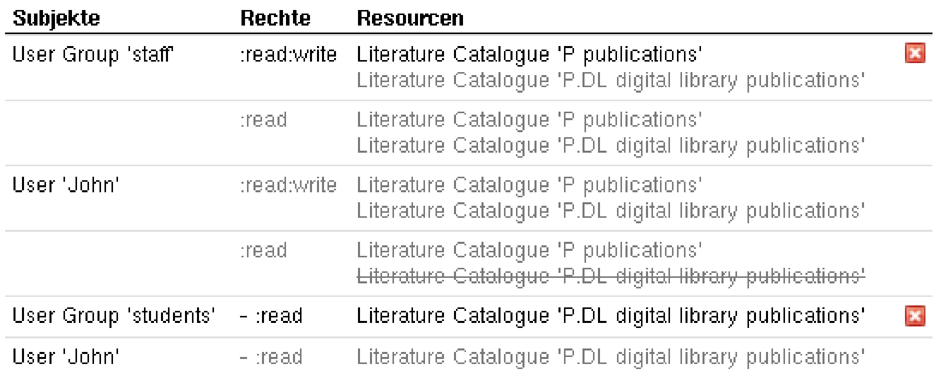


Figure 5: Tree view of user groups and users

This section should give a very brief overview over the current user interfaces for permission administration. Fig. 5 shows the hierarchical view of user groups and users. Multiple inheritance is shown by displaying the same entry beneath each of its predecessors, as it is the case for the user group 'em_hiwis' that is part of 'studenten' as well as 'emstaff'. Clicking on a group displays the users contained in that group, like shown for 'emweb_admins'. The same tree view is used for the other hierarchical structures, as well. Search functionality allows to search for entries in the tree and have only those parts displayed that contain the search term. This is especially useful for comprehensive trees like e.g. the one of the document catalogues.

Fig. 6 shows the administration view of the current permission specifications, including derived permissions. The permissions correspond to the example hierarchies shown in Fig. 1 and the corresponding permission specifications (see sec. 3). Specifications are printed in black and have a delete button, propagated permissions are printed in gray and do not have a delete button. This view can be printed either without any propagated permissions which is useful for getting an overview over the current access specifiations. Propagation information can be included separatedly for every hierarchy, excluding e.g. the propagation information for the objects would result in not printing the lines for the literature catalogue 'P.DL digital library publications', in the first lines for User group 'staff' and User 'John'.

One line is shown crossed out. This indicates that there is a positive permission that is overriden by another negative permission and is thus not valid. The information is nevertheless useful because it means that upon deletion of the negative permission this positive permission would become valid. The interface does at this stage not directly show how the propagated permissions are derived. In our system it can be inferred by the data shown because the privileges are always shown with their whole hierarchical path which makes evident which privileges are inherited. A similar mechanism applies to the catalogues that contain index letters (like 'P', 'P.DL' in Fig. 6). Further enhancements of the



Subjekte	Rechte	Resourcen	
User Group 'staff'	:read:write	Literature Catalogue 'P publications' Literature Catalogue 'P.DL digital library publications'	✖
	:read	Literature Catalogue 'P publications' Literature Catalogue 'P.DL digital library publications'	
User 'John'	:read:write	Literature Catalogue 'P publications' Literature Catalogue 'P.DL digital library publications'	
	:read	Literature Catalogue 'P publications' ~~Literature Catalogue 'P.DL digital library publications'~~	
User Group 'students'	- :read	Literature Catalogue 'P.DL digital library publications'	✖
User 'John'	- :read	Literature Catalogue 'P.DL digital library publications'	

Figure 6: Permission administration

interface are planned in order to show more clearly how permissions are propagated.

As the list of permission specifications can be long, especially when including derived permissions, the system allows to restrict the displayed information to certain subsets of subjects, privileges and objects and any combination of these. For example, the list can be restricted to some users or user groups and maybe certain privileges. This is very useful for understanding why a specific access is granted or denied.

6 CONCLUSION

In this paper we have presented an access control system that is used for a hybrid library and other, web-based applications. The major aim of the proposed system is high flexibility in the definition of permissions which is accomplished by introducing hierarchical relationships on the subjects, the privileges and the objects. Subjects are users or user groups and objects can be different types of resources identified by a combination of resource type and resource identifier. Subjects and objects are organized in a hierarchy of supergroups and subgroups by partial order relationships, whereas the partial order on the privileges expresses implication of a privilege by another. We have introduced the propagation rules for positive and negative permissions using a conflict resolution policy that always lets negative permissions prevail. Some motivating examples of the application of the hierarchies show how the flexibility can be exploited in practical situations. We have described the architecture and detailed implementation of the function for checking access together with redundancy issues on the database level.

The system presented here is implemented on a central server. Future work includes the possibility of distribution of the access control mechanism. The design of the corresponding database model is also subject to further research. A higher level of redundancy could be used to ensure good performance of the frequently used checkAccess function. We are also working on further enhancements of the user interface especially to show from where propagated permissions originate. A very nice feature would be the automatic calculation or proposition of document catalogues based on document attributes similar to the system proposed in [10]. Another very useful property that we did not deal with yet is the delegation of permissions by single users. Questions are which permissions exactly can be delegated by a user and how a comprehensible user interface can be designed that lets the user be aware of all implications of a delegated permission.

Acknowledgement The authors gratefully acknowledge the funding of the project “Recommendersysteme für Meta-Bibliothekskataloge” (RecKVK) by the Deutsche Forschungsgemeinschaft (DFG).

References

[1] N. Adam, V. Atluri, E. Bertino, E. Ferrari, “A content-based authorization model for digital libraries”, **IEEE Trans. on Knowledge and Data Engineering**, Vol. 14, No. 2, 2002, pp. 296–315.

[2] C. Baru and A. Rajasekar, “A hierarchical access control scheme for digital libraries”, **International Conference on Digital Libraries**, ACM Press, 1998, pp. 275–276.

[3] E. Bertino, E. Ferrari, and A. Perego, “Max: an access control system for digital libraries and the web”, **Computer Software and Applications Conf.**, 2002, pp. 945–950.

[4] E. Bertino, “Rbac models – concepts and trends”, **Computers & Security**, Vol. 22, No. 6, 2003, pp. 511-514.

[5] E. Bertino, A. Bonatti, and E. Ferrari, “TRBAC: A temporal role-based access control model”, **ACM Transactions on Information and System Security**, Vol. 4, No. 3, 2001, pp. 191-223.

[6] E. Bertino, B. Catania, E. Ferrari, and P. Perlasca, “A logical framework for reasoning about access control models”, **ACM Transactions on Information and System Security**, Vol. 6, No. 1, 2003, pp. 71-127.

[7] E. Bertino, E. Ferrari, and A. Perego, “An access control system for digital libraries and the web: The max prototype demonstration”, **Research and Advanced Technology for Digital Libraries**, Springer-Verlag, Sep 2002, pp. 656–657.

[8] K. Bhoopalam, K. Maly, F. McCown, R. Mukkamala, and M. Zubair, “A standards-based approach for supporting dynamic access policies for a federated digital library”, **Digital Libraries: Implementing Strategies and Sharing Experiences**, Springer-Verlag GmbH, 2006, pp. 242–252.

[9] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, and R. Chandramouli, “Proposed NIST standard for role-based access control”, **ACM Transactions on Information and System Security**, Vol. 4, No. 3, 2001, pp. 224 – 274.

[10] E. Ferrari, N.R. Adam, V. Atluri, E. Bertino, and U. Capuozzo, “An authorization system for digital libraries”, **The VLDB Journal – The International Journal on Very Large Data Bases**, Vol. 11, No. 1, 2002, pp. 58–67.

[11] N. Kodali, C. Farkas, and D. Wijesekera, “An authorization model for multimedia digital libraries”, **Int. Journal on Digital Libraries**, Vol. 4, No. 3, 2004, pp. 139–155.

[12] R.S. Sandhu and P. Samarati, “Access control: principle and practice”, **Communications Magazine, IEEE**, Vol. 32, No. 9, 1994, pp. 40–48.

[13] H. Shen and P. Dewan, “Access control for collaborative environments”, **Computer-supported cooperative work**, ACM Press, New York, NY, USA, 1992, pp. 51–85.