Towards Access Control for Logical Document Structures*

Fredj Dridi, Gustaf Neumann
Information Systems and Software Techniques
University of Essen, Universitätsstraße 9, D-45141 Essen, Germany
{*Fredj.Dridi,Gustaf.Neumann*}*@uni-essen.de*

Abstract

This paper presents a first step towards a security model that defines access control for logical document structures. This model benefits from roles to abstract from users and from security levels (classifications) that abstract from objects. The security levels are defined on top of a complex document structure which will be needed for real web applications. Since a user clearance for an operation can be designated from roles and permissions, we use a lattice that defines a partial order over classifications to make an authorisation decision. Ordinary users should be able to handle the right management of their documents. The proposed model can be used in a decentral way.

1. Introduction

When the Inter- or Intranet is used for cooperative work the users want to create new documents to be shared with others or they want to get access to documents created by others. In a project complex hyper-linked documents are shared among the participants who are working collectively on documents that are built from several physical units. These units are located typically on various decentralised web servers that are owned and maintained by fairly independent parts of an organisation or by a single user. In this paper we envision an environment where for example each user is able to publish his own documents on his private web server. This owner of the web server gives other users the ability both to read and to upload (write) documents.

The web is growing to be a distrubuted platform for truly (e-commerce) cooperative applications that deals with a large number of documents which are not equally sensitive and consequently are not entrusted to every user. For this reason both the management of the documents and the administration of access rights is becoming increasingly complex. This paper is the first step towards a system that should empower and enable the user to administer the access rights of his documents while keeping these coherent with enterprise wide policies. In this paper we concentrate on single web servers and address solely the read operation. The write operation and therefore secure information flow will be considered in future work.

The paper is structured as follows: In Section 1.1, we define the logical document structure. We describe in Section 1.2 how document structuring and layouting concepts are used to define some policies and how these concepts are relevant for access control. A lattice will be used to define a partial order over the relevant classifications of the components of a document. Section 2 offers a brief discussion of the requirements for access control in distributed hypertext systems and gives brief description of approaches in this direction. In Section 3 and 4, we describe our lattice-based authorisation model that can be used in an decentralised environment. This model uses roles to abstract from subjects and classifications to abstract from objects in order to make simplify domain management (administration). In Section 5, we propose an architecture for an implementation. Finally, Section 6 offers concluding remarks and a brief description of future work.

1.1. Logical hypertext documents

As mentioned above this work focuses on access control in Inter-/Intranet based hypertext systems using XML. XML [4] is a recommendation of the W3 Consortium and is regarded as a likely successor of HTML [22]. In a hypertext system a logically closed document is built from several physical units. We distinguish here between a physical document and a logical document. The physical document structure is simply the set of files (text, images, DTD, style information, etc.) needed to define a document (see Figure 1). A logical document structure can be defined along the following orthogonal concepts: Document structure, Composite structure, Navigational structure, Layout definition.

The *document structure* defines how a single unit (file)

*Published in: Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, DEXA 98, pages 322-327, Vienna, Austria, August, 1998.

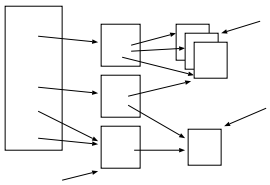

Figure 1. Physical document structure (many interlinked files).

is built of elements (title, paragraph, address, figure, etc.). The document structure is defined through a markup language. Markup languages based on SGML [1] (like HTML or XML) allow each element to have attributes. We recommend the use of XML for Web based documents since (a) it ensures (in contrary to HTML) a well defined structure for the documents and (b) it is the basis for various novel extensions (such as RDF, XLink, and name spaces [2]).

The *composite structure* defines how a composite document is built from smaller components which can be either composite documents as well or basic units. A typical Web document is composed of several source files that can be markup text, images, or other components.

The *navigational structure* defines how a user can move around within and between documents by following links. The navigational links permit non-sequential reading (or writing) [19].

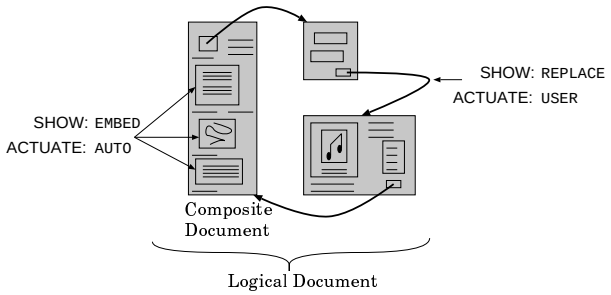

Figure 2. Embedding and linking of basic units to form composite and logical documents.

The composite and navigational structure is defined by XLink (XML Linking Language) [17]. XLink provides a simple set of constructs (e.g. `XML element link`) to describe links between elements. A link has some attributes to express its behaviour. The SHOW attribute (value either EMBED or REPLACE) of a link indicates whether each linked resource should be embedded in the current document or should be presented separately. The ACTUATE attribute (value either AUTO or USER) defines whether the link should be activated automatically on load or by user interaction. For instance a link with the attribute values AUTO and EMBED is similar to the `<IMG>`-Tag in HTML. Figure 2 shows how basic units can be embedded to form a composite document.

A logical document is defined as the set of documents connected through navigational links. The elements of this set represent a single logical content. In this paper we assume for the technical realization that a logical document is formed by the transitive closure of all the navigational links starting from an entry point.

Finally the *layout definition* specifies how the elements of the document, composite and navigational structure are rendered and presented to the users. CSS (Cascading Style Sheets) [16] is a good candidate for a layout definition language. CSS describes the presentation (e.g. fonts, colors and spacing) of structured documents.

1.2. Access policies for logical documents

We use in this paper the term *classification* as in the most of the literature (e.g. [3]) to express the level of the sensitivity of a document or its component. The term *clearance* is used to express the degree of trust associated with a user. Both classification and clearances are called also security labels that can be attached to the documents and to the user. In a lattice model such as [24, 5] the set of security labels form a lattice structure with a partially ordered dominance relation $\geq$ and a least upper bound operator. In this paper we use such a dominance relation to compare the classification of a requested document and the clearance of the requester in order to make the authorisation decision. The set of security labels used in this work will be determined from the document owner which form a lattice. For some applications a hierarchy of security labels can be sufficient.

As mentioned in [14, 18] information must be generally protected from three kinds of threats: (a) the unauthorised disclosure, (b) the unauthorised modification and (c) the unauthorised withholding of information. In this paper we address the unauthorised disclosure of information stored in Web documents. Therefore we distinguish the following mutual exclusive classes of policies for enforcing access control in an Inter-/Intranet context: Access Denial Policy, Covert Censure Policy and Censure Policy. The other kinds of threats will be addressed in future work.

The *Access Denial Policy* is used when an access to an entire logical document will be denied if it contains unauthorised information (for a specific user).

The *Covert Censure Policy* filters the unauthorised information from the document. The presented document is incomplete and the user can not infer the incompleteness.

The *Censure Policy* filters also the unauthorised information from the document and replace it with a standard place holder. The presented document is also incomplete but the

user is able to see the incompleteness.

From the security point of view and with respect to these policies both the structuring concepts and the layout definition are relevant. For example we can now apply the policies above on the *composite structure* and the *navigational structure*. Consider for example Figure 2 that depicts a logical document containing composite components (gray box) and navigational links. Both the composite components and the navigational links can be subjects of security. In order to be able to present a composite document completely to the user according to the "Access Denial Policy", the composite document as a whole must have a classification that is at least the level of classification of the most classified information it contains in all embedded parts (multilevel document in [14]). If we use the "Covert Censure Policy" all navigational links pointing to unauthorised information must be filtered before the document is presented. By using the layout definition language CSS it is also possible to prohibit visualisation of some document elements or parts.

The *document structure* can be used as well for example in e-commerce applications. Because it can be necessary to sign some elements of a document (e.g. contract agreement). This can be achieved by using digital signatures on those elements of a document (document structure). Since the main goal of the paper is to deal with access control these two issues will not be discussed further.

2. Requirements for Access Control in Distributed Hypertext Systems

In this paper we are concerned with cooperative work that is carried out by people working in an organisation. Documents are shared among people or projects in an Inter- or Intranet context. The documents are placed typically on various decentralised web servers that are owned and maintained by fairly independent parts of the organisation. In an Intranet there is often one web server per department, sometimes personal web servers are used. One of the challenges of this paper is to design a security mechanism that is both applicable in the decentralised context (no complete knowledge about all objects) and which is manageable by users (rather than system administrator) that want to share information in a controlled way.

Finally there is a quantity problem: the documents we are dealing with have complex structure sometimes composed of hundred of physical units which have to be handled in a convenient way. The number of potential users is overwhelming as well. In other words there is a strong need for abstraction mechanisms for the subjects and objects in the access control system. Therefore we propose the following abstraction mechanisms:

- User are modelled by roles [25].
- Roles obtain rights (clearances) based on classifications (for certain operation).

In addition we propose some of the design goals for access control in distributed hypertext systems:

- Navigation restriction. Only authorised users should be able to navigate through a logical document or a part of it (read).
- Document authoring integrity. Only authorised users should be able to create or write a logical document or a part of it.
- Decentral management of access control. To improve autonomy, privacy and the document ownership a decentral management of access control is needed here. By this way user and document administration must be simplified.

2.1. Related work

In [21, 20] "role-based access control for the Web" is defined on the file or directory level. Every user is assigned to a role. Every role has permissions in order to approve a particular operation to access a document (file). This system uses an LDAP server [26] to obtain access control information. The system uses distinguished names for the roles as defined by the X500 standard [8].

The Phoenix project [15] is a distributed hypermedia authoring system which integrates access control information to hypermedia documents. Documents (files) are protected by means of ACLs. Each document includes a HTML mark-up element giving the URL of an ACL. In order to authorise document requests, a document server sends the requested method, the ACL URL reference and the authenticated client-name to an authorisation server. The authorisation use the proposed ACL to make a decision which be returned to the document server. The administration of access control is here done by editing the HTML files. In addition a server must unusually parse the HTML file before sending it to the client.

A capability-based authorisation model for the WWW is proposed in [11, 10, 12]. This model uses a distributed authorisation model which provides authorisation at document and contents level (file or logical document level). The client (user) administration is simplified as only one server needs to know its potential clients. Document and content servers make local authorisation decisions using capabilities presented by their clients. Indeed without an initial capability related to the root document, a client will not be able to access any node from the document. By this way a logical document could be protected. The model supports existing WWW node migration techniques. The major limitation of this model is the regeneration of capabilities, since they have a validity period that expires.

The next section describes the authorisation model for a single document repository. Section 4 discusses access control in decentralised environment. Section 5 proposes an architecture suited for implementation.

3. The Lattice-Based Authorisation Model

3.1. Document repository

In the following we call the set of documents maintained by a single authority a *document repository*. Following the approach in [19] we assume that every document has an owner. The owner is the person who created and stored the document in this repository and who is responsible for it. Only the owner has the right to withdraw a document, change or delegate its access rights. Figure 3 shows an example of a document repository which contains a logical document that is built from several hperlinked components. We use the labels C_1 – C_6 to denote the classification of these components (multilevel document). This labels can form a lattice as shown on the right hand of Figure 3 where $C_0 \geq C_1$, $C_1 \geq C_3$, $C_1 \geq C_2$, $C_3 \geq C_6$, $C_3 \geq C_5$ and $C_2 \geq C_4$. A document repository can be e.g. the set of documents belonging to a project or a person stored on a web server. This web server can be owned by single user. Therfore each document repository has an owner who has

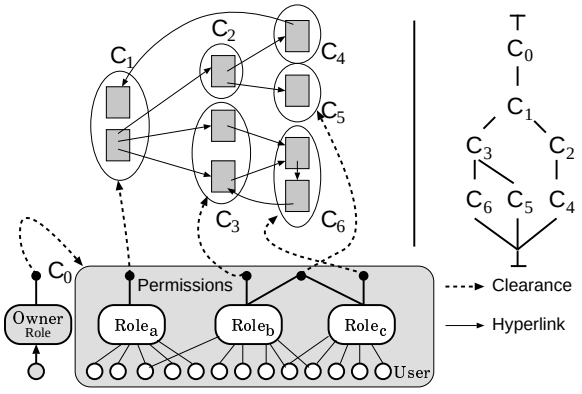

Figure 3. Autonomous document repository.

the most access rights (see Figure 3 – shaded role). This implies that the owner is more trustworthy than other users and therefore he is assigned to the highest security label in the system (in Figure 3 – C_0 that dominates all other labels). In addition the owner is able to (a) define new classifications in the document repository, (b) define all the roles needed, and (c) create new documents and classify existing ones. If assigned classifications and clearances change over time a reorganisation of affected roles, permissions and documents will be initiated from the owner of the document repository.

The following basic requirements must be supported by a document repository:

- Authentication: In order to get access to the document repository each user has to be identified.
- Authorisation: A document repository must be able to decide if a given user is allowed to access a document.
- Autonomy: Authentication and authorisation are decision determined solely within the repository. Role names and security labels can be freely chosen, since each document repository should has its own name space e.g. by using DN (distinguished name) from X500 standard [8].

We assume here that a successful authentication has been established for example by using X509 certificates [9]. Authorisation and autonomy will be discussed below.

3.2. Lattice-based authorisation model within a document repository

The main task of an authorisation model is to determine whether a subject is allowed to perform an operation on a document. In this paper a subject is either the owner or an arbitrary user. Figure 4 shows which steps are needed to make an authorisation decision:

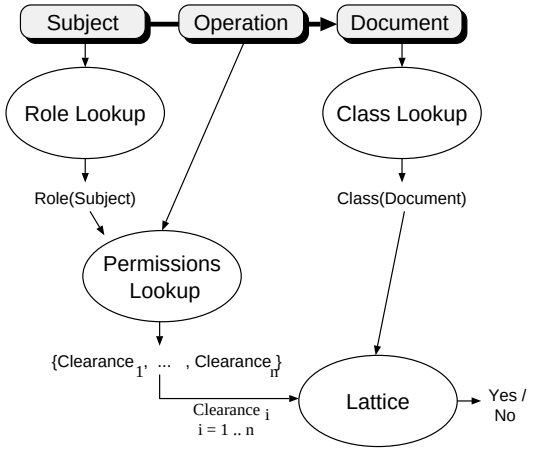

Figure 4. Authorisation decision.

1. Role Lookup. A role for each user will be automatically or manually determined.
2. Class Lookup. A classification for each document will be determined.
3. Permission Lookup. Every role is equipped with permissions. Permissions designate the clearances for operations. Permission lookup results in a set of clearances that is implied by the role hierarchy.

4. Lattice. At this step it will be determined whether an access will be allowed or denied by using the clearances of the subject, classification of the document and a dominance relation.

Consider for example the multilevel document depicted in Figure 3. Each user in role "Role$_a$" will obtain a permission that corresponds to the clearance C_1. Since this clearance dominates all other security labels this user is allowed to navigate (read) through the entire logical document. In contrary, according to the lattice another user in either "Role$_b$" or "Role$_c$" will have a restricted read access to the whole logical document. Role$_b$ (Role$_c$) is equipped with permissions that correspond to C_3 and C_5 (C_5 and C_6).

4. Authorisation in Decentralised Environment

In an Inter- or Intranet context a user will need documents from several independent document repositories. Since each document repository is an autonomous environment it has its own security policy typically with different roles, permissions and document classifications. Furthermore each document repository has its own local lattice that controls access to the repository. In order to define access control within several independent document repositories these local lattices are involved in a decentral way. This causes that the labels from different lattices must be either direct or through a mapping function comparable.

Figure 5 depicts two document repositories with different lattices. The documents `D1`, `D11` and `D12` are hper-

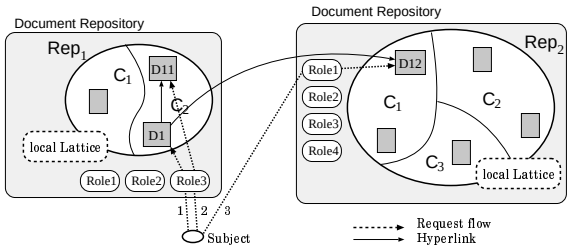

Figure 5. Decentralised authorisation model.

linked components of a given logical document. Consider a user that desires to navigate through this logical document. Since these documents are differently classified and belonging to different document repositories the user must be separately authorised. For this purpose three requests are required (Figure 5 – dashed arrows). Each request will be separately handled. This means that each document repository has to achieve the two steps mentioned above (Role Lookup and Permission Lookup) in order to authorise the requests 1 , 2 and 3 by using the local lattices.

5. Implementation Issues

In this section we give a short description of an architecture suited to implement the lattice-based access control for a single document repository. This architecture has three basic components (see Figure 6):

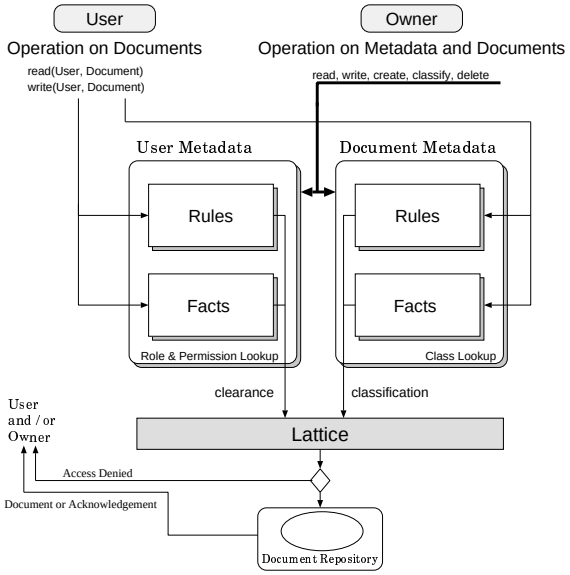

Figure 6. Authorisation manager for a document repository.

User Metadata. This component is used to determine the clearances of a user. These clearances will be designated from both roles and permissions. In other words the main task of this component is to determine the roles of a user (Role Lookup) and to determine from these roles the permissions (Permissions Lookup). The assignment can be either achieved by using some facts (e.g. asking a LDAP server [26]) or by evaluating some rules.

Document Metadata. This component is used to determine the classification of a document (Class Lookup). The classification of the document will be given by a fact (e.g. trusted PICS [23]).

Lattice. This component is used to determine whether an access will be allowed or denied by using the previously determinated clearance of a given subject and classification of the desired document. The Owner is responsible to define the dominance relation.

Generally a subject needs for every operation at least the right to read the documents metadata. We distinguish between operations performed on documents and on metadata:

- *Operations on Documents* can be performed both by user and owner. A *read*-operation returns either the requested document or failure code, a *write*-operation returns always a return code that indicates whether the write was successful or not.

- *Operations on Metadata* can be performed solely by the owner. The owner is allowed (a) to create a role, (b) to assign roles permissions, (c) to classify a document, and (d) to delete documents, roles and permissions.

6. Conclusion and Future Work

In this paper we have described a first step towards designing a security mechanism that can be used in decentralised applications over the Inter- or Intranet. We focus on the ease of administration to allow users to share information in a controlled way. We use a dual abstraction mechanism to reduce the number of administrative steps: Roles are used to abstract from subjects, classifications are used to abstract from objects. Permissions in the form of clearances for operations are assigned to roles rather than to subjects and can be inherited from other roles. A lattice is used to define a partial order over the classifications of the documents. We have finally presented an architecture for an implementation.

The future work will go into two directions: Most of the operations in the web are read operations for which the proposed approach is sufficient. In order to obtain some experiences with our approach we want to implement a prototype for the rights management and access control. We will implement the system using the prototyping environment Cineast [13] which has server and browser functionality and covers cryptographic support via SSL (Secure Socket Layer) [7]. Secondly we want to put more effort into the secure handling of write operations in the web (support for the HTTP PUT method [6]). We are trying to use the lattices to restrict the information flow. In our opinion, if we succeed this will be an important step towards truly collaborative web environments.

References

[1] The SGML/XML Web Page. http://www.sil.org/sgml/.

[2] World Wide Web Consortium. http://w3c.org/.

[3] D. Bell and L. LaPadula. *Secure Computer System: Unified Exposition and Multics Interpretation*. In Technical Report MTR-2997. MITRE Corp., USA, July 1976.

[4] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. *World Wide Web Consortium Recommendation*, February 1998. http://w3c.org/TR/1998/REC-xml-19980210.

[5] D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, May 1976.

[6] R. Fielding, T. Berners-Lee, and H. Frystyck. Hypertext transfer protocol - HTTP/1.0. *RFC 1945*, May 1996. http://w3c.org/Protocols/rfc1945/rfc1945.

[7] A. Freier, P. Karlton, and P. Kocher. The SSL protocol version 3.0. *Netscape*, 1996. http://netscape.com/eng/ssl3/.

[8] ITU-T. *Recomendation X.500: Information Technology – Open Systems Interaction - The Directory: Overview of Concepts, Models, and Services*. 1993.

[9] ITU-T. *Recomendation X.509: Information Technology – Open Systems Interaction -The Directory: Authentification framework*. 1993.

[10] J. Kahan. A capability-based authorization model for the World Wide Web. In *Third World Wide Web Conference*, pages 1055–1064, April 1995.

[11] J. Kahan. A distributed authorization model for WWW. In *INET*, May 1995.

[12] J. Kahan. *Conception et expérimentation d'un modèle de contrôle d'accès non nominatif pour les systèmes hypermédia répartis*. Université de Rennes 1, Thèse, December 1997.

[13] E. Köppen, S. Nusser, and G. Neumann. Cineast – an extensible Web browser. In *Proceedings of the WebNet World Conference on WWW, Internet and Intranet*, November 1997.

[14] C. E. Landwehr. Formal models of computer security. *ACM Computing Surveys*, 13(3):247–278, September 1981.

[15] M. G. Lavenant and J. A. Kruper. The phoenix project: Distributed hypermedia authoring. In *First World Wide Web Conference*, 1994.

[16] H. Lie and B. Bos. Cascading style sheets, level 1. *W3C Recommendation*, December 1996. http://w3c.org/TR/REC-CSS1.

[17] E. Maler and S. DeRose. XML linking language (XLink). *World Wide Web Consortium Working Draft*, March 1998. http://w3c.org/TR/1998/WD-xlink-19980303.

[18] D. Merkl and G. Pernul. Security for next generation of hypertext systems. *Hypermedia*, 6(1):1–18, 1994.

[19] T. H. Nelson. *Literary machines*. Ted Nelson, 1986.

[20] G. Neumann and S. Nusser. A framework and prototyping environment for w3 security architecture. In *Proceedings of Communications and Multimedia Security*, September 1997.

[21] S. Nusser. *Entwurf und Implementierung von rollenbasierten Sicherheitskonzepten im World Wide Web*. Wirtschaftsuniversität Wien, Dissertation, May 1997.

[22] D. Raggett. HTML 3.2 reference specification. *W3C Recommendation*, January 1997. http://w3c.org/TR/REC-html32.html.

[23] P. Resnick and J. Miller. PICS: Internet access controls without censorship. *Communication of the ACM*, 39(10):87–93, 1996. http://w3c.org/PICS.

[24] R. S. Sandhu. Role hierarchies and constraints for lattice-based access controls. In *Proc. Fourth European Symposium on Research in Computer Security*, September 1996.

[25] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[26] W. Yoeng, T. Howes, and S. Kille. Lightweight directory access protocol. *RFC 1777*, March 1995.