SAFE USE OF PROTECTED WEB RESOURCES

Sylvia Encheva
Stord/Haugesund University College
Bjornsonsg. 45, 5528 Haugesund, Norway
sbe@hsh.no

Sharil Tumin
University of Bergen
IT-Dept., P. O. Box 7800, 5020 Bergen, Norway
edpst@it.uib.no

Abstract This paper focuses on a framework that ensures the safe use of protected Web resources among independent organizations in collaboration. User membership and group membership in each organization are managed independently of other organizations. User authentication and user authorization for a protected resource in one organization is determined by user group membership in other organizations. Furthermore, users never discloses their user-identifiers and passwords in a foreign domain. Every set of related roles in a single organization is defined as an antichain and every set of related roles in the collaborating organizations is defined as a complete lattice. The ranking order of roles for a resource depends on operations. One can add or remove users from roles by managing their membership in corresponding groups.

Keywords: E-services

Introduction

One of the most difficult problems in managing large networked systems is information security. Computer-based access control can prescribe not only who or what process may have access to a specific system resource, but also the type of access that is permitted. In Role-Based Access Control (RBAC), access decisions are based on an individual's roles and responsibilities within the organization or user base [10].

The majority information and communication technology (ICT) based systems are constructed in such a way that user authentication and authorization data have to reside locally in their user database. As a consequence, any orga-

nization using such a system is forced to export its users' data to that system. Such a requirement implies a complicated data synchronization mechanism.

User management in a large networked system is simplified by creating a group for each role where addition or removal of users from roles is done by managing their membership in corresponding groups. The problem of a person affiliated with many organizations at the same time is difficult to solve and may not be a major issue if a conflict of interests can be resolved in a role-group relationship. As a possible solution we suggest defining every set of related roles in a single organization as an antichain and every set of related roles in the system of collaborating organizations as a complete lattice.

Lattices have been used to describe secure information flow in [7] and [17]. However, to the best of our knowledge the problem of groups and roles has not been considered in relation to formal concept analysis, concept lattices and complete lattices.

The rest of the paper is organized as follows. Related work is listed in Section 1. Basic terms and concepts are presented in Section 2. A collaboration among independent organizations and a conflict of roles are discussed in Section 3. The paper ends with a conclusion.

1. Related Work

Formal concept analysis [23] started as an attempt of promoting better communication between lattice theorists and users of lattice theory. Since 1980's formal concept analysis has been growing as a research field with a broad spectrum of applications. Various applications of formal concept analysis are presented in [11].

Methods for computing proper implications are presented in [4] and [22].

A formal model of RBAC is presented in [9]. Permissions in RBAC are associated with roles, and users are made members of appropriate roles, thereby acquiring the roles' permissions. The RBAC model defines three kinds of separation of duties - static, dynamic, and operational. Separation of duties was discussed in [2], [9] and [20]. The use of administrative roles for decentralization of administration of RBAC in large-scale systems is considered in [18]. Assigning roles to users in systems that cross organizational boundaries is discussed in [13] and [14]. A framework for modeling the delegation of roles from one user to another is proposed in [1]. A multiple-leveled RBAC model is presented in [5]. The design and implementation of an integrated approach to engineering and enforcing context constraints in RBAC environments is described in [21].

While RBAC provides a formal implementation model, Shibboleth [19] defines standards for implementation, based on OASIS Security Assertion Markup Language (SAML) [16]. Shibboleth defines a standard set of instructions be-

tween an identity provider (Origin site) and a service provider (Target site) to facilitate browser single sign-on and attribute exchange. Our work differs from Shibboleth in modeling implementation and user/group/role management. Shibboleth invests heavily on Java and SAML standards. Our model is more open-ended based on SOAP written in Python [12]. The Origin site manages user and group memberships of users while the Target site manages permissions and role memberships of groups. The Origin site provides procedures callable using SOAP from Target sites to facilitate authorization on a protected resource. Additional needed procedures come to being by mutual agreement betwen sites.

2. Users, Groups, Roles and Permissions

In this paper a *user* φ is defined as a valid net identity at a particular organization Γ. A valid net identity can be a human being, a machine or an intelligent autonomous agent.

A *group* Ω is a set of users $\{\varphi_j\}_1^s$, i.e. $\Omega = \{\varphi_j | \varphi_j \in \Gamma\}$. A group is used to help the administration of users. The security settings defined for a group are applied to all members of that group.

A *role* Φ contains a set of groups $\{\Omega_i\}_1^l$ associated with similar duty and authority. User administration is simplified by creating a group for each role. One can add or remove users from roles by managing their membership in corresponding groups.

A *resource* Υ defines a set of protected Web objects $\upsilon_j, j = 1, ..., m$.

An *action* Ψ, where

$$\Psi = \begin{pmatrix} (\varsigma_1, \upsilon_1) & ... & (\varsigma_1, \upsilon_l) & ... & (\varsigma_1, \upsilon_m) \\ ... & ... & ... & ... & ... \\ (\varsigma_i, \upsilon_1) & ... & (\varsigma_i, \upsilon_l) & ... & (\varsigma_i, \upsilon_m) \\ ... & ... & ... & ... & ... \\ (\varsigma_n, \upsilon_1) & ... & (\varsigma_n, \upsilon_l) & ... & (\varsigma_n, \upsilon_m) \end{pmatrix}$$

is a matrix of operations $\varsigma_i, i = 1, ..., n$ on objects $\upsilon_j \in \Upsilon, j = 1, ..., m$.

EXAMPLE 1 *If operations are read ρ, write ξ, delete τ, copy ϑ, and move μ on the objects $(\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4, \upsilon_5)$, then the action is*

$$\Psi = \begin{pmatrix} (\rho, \upsilon_1) & (\rho, \upsilon_2) & (\rho, \upsilon_3) & (\rho, \upsilon_4) & (\rho, \upsilon_5) \\ (\xi, \upsilon_1) & (\xi, \upsilon_2) & (\xi, \upsilon_3) & (\xi, \upsilon_4) & (\xi, \upsilon_5) \\ (\tau, \upsilon_1) & (\tau, \upsilon_2) & (\tau, \upsilon_3) & (\tau, \upsilon_4) & (\tau, \upsilon_5) \\ (\vartheta, \upsilon_1) & (\vartheta, \upsilon_2) & (\vartheta, \upsilon_3) & (\vartheta, \upsilon_4) & (\vartheta, \upsilon_5) \\ (\mu, \upsilon_1) & (\mu, \upsilon_2) & (\mu, \upsilon_3) & (\mu, \upsilon_4) & (\mu, \upsilon_5) \end{pmatrix}$$

A *permission* Λ defines a right of a role Φ to perform an action Ψ_Φ on a resource Υ. A user φ has a role Φ_Ω when $\varphi \in \Omega$ and Ω has a role Φ.

A user has a permission only if the user is a member of a group with authorized actions associated with a role. A user φ automatically inherits all permissions associated with the groups to which φ belongs. An authorization gives a set of permissions to a user to execute a set of operations (e.g. read, write, update, copy) on a specific set of resources (e.g. files, directories, programs). An authorization also controls which actions an authenticated user can perform within a Web-based system. A non zero element of the matrix Ψ_ϕ defines a permission. All non zero elements of the matrix Ψ^o_ϕ (see Example 2) define the permissions of a role within a system.

EXAMPLE 2 *If operations are read ρ, write ξ, delete τ, copy ϑ, and move μ on objects $(\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4, \upsilon_5)$, then*

$$\Psi_{\phi^o} = \begin{pmatrix} (\rho,\upsilon_1) & 0 & (\rho,\upsilon_3) & (\rho,\upsilon_4) & (\rho,\upsilon_5) \\ (\xi,\upsilon_1) & (\xi,\upsilon_2) & 0 & (\xi,\upsilon_4) & (\xi,\upsilon_5) \\ 0 & (\tau,\upsilon_2) & (\tau,\upsilon_3) & 0 & (\tau,\upsilon_5) \\ (\vartheta,\upsilon_1) & (\vartheta,\upsilon_2) & 0 & (\vartheta,\upsilon_4) & (\vartheta,\upsilon_5) \\ (\mu,\upsilon_1) & (\mu,\upsilon_2) & (\mu,\upsilon_3) & (\mu,\upsilon_4) & 0 \end{pmatrix}$$

By Ψ_{ϕ^o} we denote the matrix Ψ where at least one of its elements is equal to 0.

An authenticated user, who belongs to a group Ω in an organization Γ_i, will have permissions to perform actions at another organization Γ_j if Ω defined at Γ_i is a member of a role in Γ_j.

DEFINITION 3 *A set P is an* ordered *set if $x \leq y$ only if $x = y$ for all $x, y \in P$.*

Let P be a set. An order *(or* partial order*) on P is a binary relation $\leq$ on P such that, for all $x, y, z \in P$,*
i) $x \leq x$,
ii) $x \leq y$ and $y \leq x$ imply $x = y$,
iii)$x \leq y$ and $y \leq z$ imply $x \leq z$.

A set P equipped with an order relation $\leq$ is said to be an *ordered set*. An ordered set $\mathcal{P}$ is an *antichain* if $x \leq y$ in $\mathcal{P}$ only if $x = y$. For $x, y \in \mathcal{P}$, we say x is *covered* by y, if $x < y$ and $x \leq x < y$ implies $z = x$.

Let $S \supseteq P$. An element $x \in P$ is an upper bound of S if $s \leq x$ for all $s \in S$. A lower bound is defined dually. The least element in the set of all upper bounds of S is called the *supremum* of S and is denoted by $supS$. The greatest lower bound of S is called the *infimum* of S and is denoted by $inf S$.

DEFINITION 4 *Let P be a non-empty ordered set.*
i) If $sup\{x, y\}$ and $inf\{x, y\}$ exist for all $x, y \in P$, then P is called a lattice.
ii) If $supS$ and $infS$ exist for all $S \subseteq P$, then P is called a complete lattice.

A *context* is a triple (G, M, I) where G and M are sets and $I \subset G \times M$. The elements of G and M are called *objects* and *attributes* respectively [6]. The set of all concepts of the context (G, M, I) is a complete lattice and it is known as the *concept lattice* of the context (G, M, I).

For $A \subseteq G$ and $B \subseteq M$, define

$$A' = \{m \in M \mid (\forall g \in A)\ gIm\}, \quad B' = \{g \in G \mid (\forall m \in B)\ gIm\}$$

so A' is the set of attributes common to all the objects in A and B' is the set of objects possessing the attributes in B. Then a *concept* of the context (G, M, I) is defined to be a pair (A, B) where $A \subseteq G$, $B \subseteq M$, $A' = B$ and $B' = A$. The *extent* of the concept (A, B) is A while its intent is B.

3. Collaborative Management Model

Suppose an organization provides services and defines which domains can share its resources by giving a specific role membership to a group from another domain. A security administrator, working at this organization, needs a model for enforcing a policy of static separation of duties and dynamic separation of duty.

We propose an SOAP communication mechanism for determining a domain user authentication and authorization where a role with less permissions has lower rank than a role with more permissions, and every set of related roles in each organization is an *antichain* [6].

An alternative way is to only allow the minimum permission if a domain user has conflicting roles on the same resource. This is possible only if every set of related roles is a *complete lattice*. What is actually needed is that every set of related roles is a lattice, since a set of related roles in collaborating organizations is a finite set and any finite lattice is a complete lattice [6].

Roles can be ranked in such a way that a higher ranked role also contains all the rights of all lower ranked roles. Thus both roles and permissions are ordered sets with a *covering relation*. The ranking order of roles on a resource depends on operations. Role managers define a ranking order of roles on a resource.

EXAMPLE 5 *Suppose a user has two roles Φ_1 and Φ_2 effectively activated at the same session. Φ_1 and Φ_2 are defined in Table 1 and Table 2 respectively. The resulting role is role Φ^* that the system will provide under conditions defined in Table 3.*

Table 1. Context for role Φ_1

	read (r)	copy (c)	write (w)	delete (d)	move (m)
file 1 (f1)	×				
file 2 (f2)	×	×			
file 3 (f3)		×	×		×
file 6 (f6)	×	×		×	×

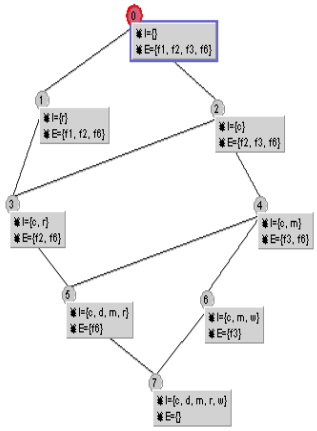

Figure 1. Context lattice for the role Φ_1

Table 2. Context for role Φ_2

	read (r)	copy (c)	write (w)	delete (d)	move (m)
file 1 (f1)		×			
file 2 (f2)		×	×		
file 4 (f3)	×		×	×	
file 5 (f5)	×	×			×
file 7 (f7)	×		×	×	×

Table 3. Context for role Φ^*

	read (r)	copy (c)	write (w)	delete (d)	move (m)
file 1 (f1)	×				
file 2 (f2)	×	×			
file 3 (f3)		×	×		×
file 4 (f4)	×		×	×	
file 5 (f5)	×	×			×
file 6 (f6)	×	×		×	×
file 7 (f7)	×		×	×	×

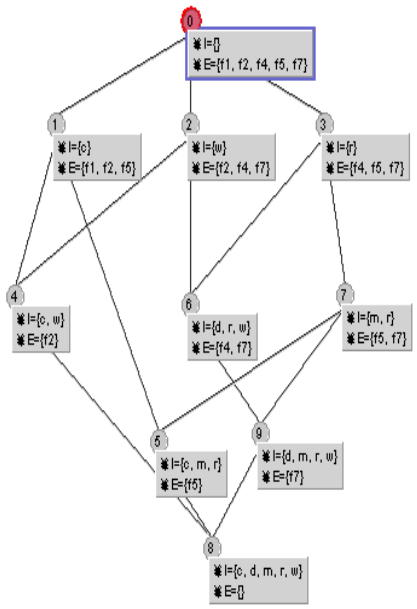

Figure 2. Context lattice for the role Φ_2

A role is given to a user after authentication, defines authorization on a resource, defines operational rights and responsibilities of a user on a resource, and is a dynamic attribute of a user operating on a resource. Roles conflicts appear when a user simultaneously has both a higher ranked role and lower ranked roles on a resource. In such a case, the use will get the role with the least rank, and, therefore receives minimum permission on that resource. Role data in a service provider organization contains references to external group data from client organizations.

EXAMPLE 6 *Let* $\Phi^{(m)}_{org2} = \{\Omega^{(m)}_{org4}, \Omega^{(m)}_{org3}, \Omega^{(m)}_{org2}, \Omega^{(m)}_{org1}\}$ *be a defined role at (*`org2`*) and a user* $\varphi \in \Omega^{(m)}_{org1}$ *also belongs to* $\Phi^{(m)}_{org2}$*. Then the administration for each group in* $\{\Omega^{(m)}_{org4}, \Omega^{(m)}_{org3}, \Omega^{(m)}_{org2}, \Omega^{(m)}_{org1}\}$ *is done locally at the corresponding organizations (*`org4, org3, org2, org1`*), while the administration for* $\Phi^{(m)}_{org2}$ *is done by the resource owner (*`org2`*). A permission* $\Lambda^{(m)}_{org2}$ *defines a right of the role* $\Phi^{(m)}_{org2}$ *on a resource* $\Upsilon^{(m)}_{org2}$*.*

A number of caveats exist that should be considered under implementation. Some of the more important are that:

- a Web-browser must support cookies,

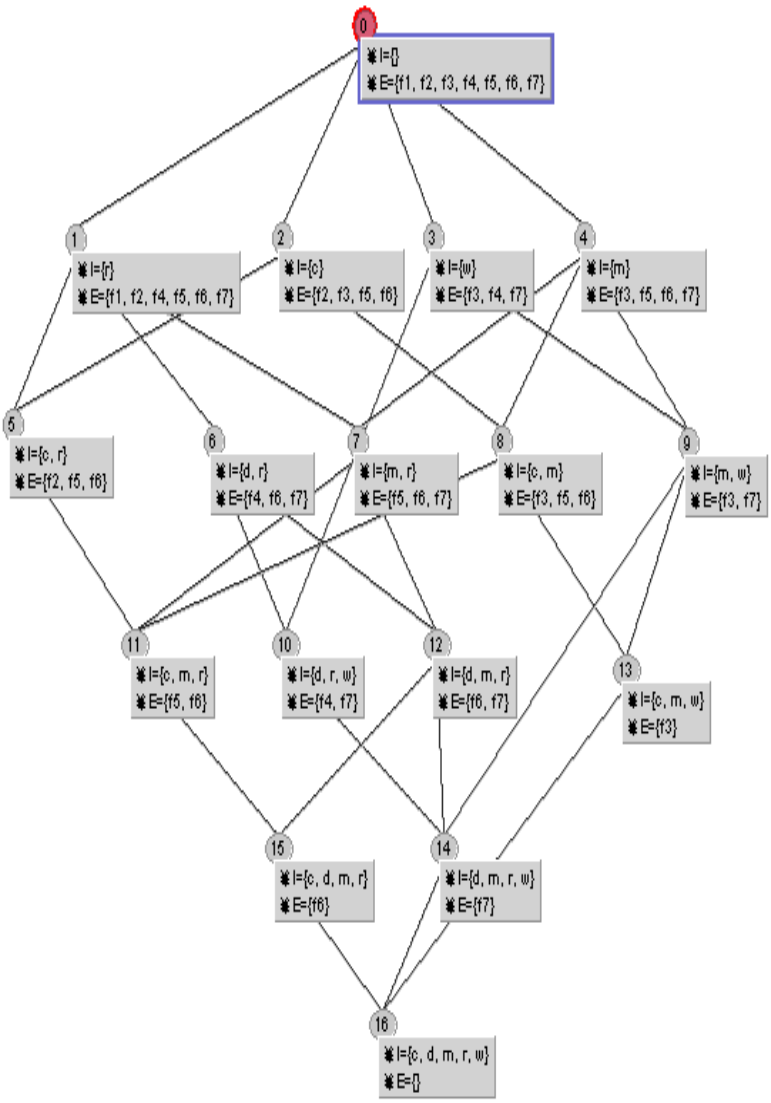

Figure 3. Context lattice for the role Φ^*

- a Web-browser must not change its IP address, i.e., behind an Internet service provider that rotates client IP addresses,
- an XML-RPC port must be allowed to pass through a firewall, and
- a Web-browser must be able to do redirection.

4. Conclusion

In this paper we propose a model that simplifies user management in co-operating educational organizations by creating a group for each role. Organizations share their user and group data with each other through a common communication mechanism using SOAP.

Arranging users into groups and roles makes it easier to grant or deny permissions to many users at once. We argue that our model may be used across organizations, based on the group structure and independent collaborative administration; and in the future, because it provides a high level of flexibility and usability.

References

[1] E. Barka and R. Sandhu. Role-based delegation model/ hierarchical roles. *20th Annual Computer Security Applications Conference*, Tucson, Arizona, 2004.

[2] E. Bertino E., P.A. Bonatti and E. Ferrari. TRBAC: A temporal Role-Based Access Control model. *ACM Transactions on information and system security* 3(3):191-223, 2001.

[3] R. Bhatti, E. Bertino, A. Ghafoor and J.B.D. Joshi. XML-based specification for Web services document security. *IEEE Computer* 37(4), 2004.

[4] C. Carpineto and G. Romano. *Concept Data Analysis: Theory and Applications*. John Wiley and Sons, Ltd., 2004.

[5] S-C. Chou. L^nRBAC: A multiple-levelled Role-Based Access Control model for protecting privacy in object-oriented systems. *Journal of Object Technology* 3(3):91-120, 2004.

[6] B.A. Davey and H.A. Priestley. *Introduction to lattices and order*. Cambridge University Press, 2005.

[7] D. Denning. A lattice model of secure information flow. *Communications of the ACM* 19(5) 1976.

[8] J. Dowling and V. Cahill. Self-managed decentralized systems using K-components and collaborative reinforcement learning. *Proceedings of the Workshop on Self-Managed Systems*, 41-49, 2004.

[9] D. Ferraiolo, R. Sandhu, S. Gavrila, R.D. Kuhn and R. Chandramouli. Proposed NIST standard for Role-Based Access Control. *ACM Transactions on Information and System Security*. 4(3):224-274, 2001.

[10] D. Ferraiolo, and R.D. Kuhn and R. Chandramouli. Role-Based Access Control. *Computer Security Series*. Artech House, 2003.

[11] B. Ganter, G. Stumme and R. Wille. *Formal Concept Analysis - Foundations and Applications*. Springer LNCS 114, Berlin, 3626, 2005.

[12] A. Martelli and D. Ascher. *Python Cookbook*. O'Reilly, UK, 2002.

[13] T. Hildmann and J. Barholdt. Managing trust between collaborating companies using outsourced role based control. *4rd ACM Workshop on RBAC*, 105-111, 1999.

[14] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor and Y. Ravid. Access control meets public key infrastructure, Or: Assigning roles to strangers. *IEEE Symposium on security and privacy*, 2000.

[15] B. Kropp and M. Gallaher. Role-based access control systems can save organizations time and money. *Information Security Magazine*, 2005.

[16] http://www.oasis-open.org

[17] R. Sandhu. Lattice-Based access control models. *IEEE Computer*, 26(11), 1993.

[18] R. Sandhu. Role activation hierarchies. *3rd ACM Workshop on RBAC*, 33-40, 1998.

[19] http://shibboleth.internet2.edu/shib-intro.html

[20] R. Simon and M. Zurko. Separation of duty in role-based environments. *Proceedings of 10th IEEE Computer Security Foundations Workshop*. Rockport, Mass., 183–194, 1997.

[21] M. Strembeck and G. Neumann. An integrated approach to engineer and enforce context constraints in RBAC environments. *ACM Transactions on Information and System Security*, 7(3):392-427, 2004.

[22] R. Taouil and Y. Bastide. Computing proper implications. *Proceedings of the ICCS-2001 International Workshop on Concept Lattice-Based Theory, methods and Tools for Knowledge Discovery in Databases*, Palo Alto, CA, USA, 49–61 2001.

[23] R. Wille. Concept lattices and conceptual knowledge systems. *Computers Math. Applic*. 23(6-9):493–515, 1992.