Proceedings of 3rd ACM Workshop on Role-Based Access Control, Fairfax, Virginia, October 22-23, 1998

Decentralized User-Role Assignment for Web-based Intranets*

Ravi Sandhu and Joon S. Park

Laboratory for Information Security Technology
Information and Software Engineering Department
George Mason University

Abstract

The intricacy of security administration is one of the most challenging problems in large networked systems. This problem is especially serious in the Web environment, which consists of synthesis of technologies and composition of various constituents. Role-Based Access Control (RBAC) can reduce the complexity and cost of security administration in large networked applications. Using RBAC itself to manage RBAC provides additional administrative convenience. The main contribution of this paper is to extend the RBAC/Web system (developed at NIST) with the URA97 model for user-role assignment (developed at GMU) to decentralize the details of RBAC administration on the Web without losing central control over the system policy.

1 INTRODUCTION

Role-Based Access Control (RBAC) ensures that only authorized users are given access to protected data or resources. RBAC has recently received considerable attention as a promising alternative to traditional discretionary and mandatory access controls (see, for example, [FK92, FCK95, Gui95, GI96, HDT95, MD94, NO95, SCFY96, vSvdM94, YCS97]).

A role is a semantic construct forming the basis of access control policy. With RBAC, system administrators can create roles, grant permissions to those roles, and then assign users to the roles on the basis of their specific job responsibilities and policy. Therefore, role-permission relationships can be predefined, which makes it simple to assign users to the predefined roles. Without RBAC, it can be difficult to determine what permissions have been authorized for what users. A general family of RBAC models called RBAC96 was defined by Sandhu et al. Details for motivation and discussion about the RBAC96 family of models are described in [SCFY96, San97].

Barkley et al at NIST (National Institute of Standards and Technology) have recently implemented a prototype for doing RBAC on Web-based intranets [BCF+97]. This system conceptually consists of two parts: Web servers which enforce RBAC at the granularity of individual URL's and an administrative tool that allows a security administrator to define roles, role-role relationships and assign user to roles. The NIST implementation provides a centralized model for assignment of users to roles by actions of the security officer.

Sandhu and Bhamidipati have recently introduced a model for decentralized administration of user-role assignment known as URA97 [SB97]. The central contribution of this paper is to describe an extension of NIST's RBAC on the Web (RBAC/Web) implementation to incorporate URA97. One of the main advantages of RBAC is the decoupling of user-role assignment and permission-role assignment. It is natural to decentralize the administration of user-role assignment. Managers routinely assign employees to different tasks and they should be able to enroll employees directly into appropriate roles to carry out these tasks. URA97 is a intuitive yet powerful model for this purpose. Its implementation on the NIST prototype demonstrates that new technologies such as the Web can accommodate such models.

*This work is partially supported by grant CCR-9503560 from the National Science Foundation at the Laboratory for Information Security Technology at George Mason University.

All correspondence should be addressed to Ravi Sandhu, ISE Department, Mail Stop 4A4, George Mason University, Fairfax, VA 22030, `sandhu@isse.gmu.edu`, `www.list.gmu.edu`.

The rest of this paper is organized as follows. Section 2 describes the background and motivation of our work. We review the URA97 administrative model for user-role assignment which itself is role-based in section 3. In section 4 we describe an implementation of the URA97/WWW system. This is followed by a discussion of related work in section 5. Conclusions and discussions for future work are described in section 6.

2 BACKGROUND

NIST has recently implemented RBAC for the WWW (RBAC/Web) applied to an Intranet computing environment to provide access control service [BCF$^+$97, FB97]. They have developed RBAC/Web implementations for both UNIX and Windows NT based servers.[1] The NIST package consists of a database, database server, API library, CGI module, session manager, and administrative tool.

A user's request in the Intranet computing environment is configured as a RBAC controlled URL by means of the Web server configuration files that map URLs to file names. To enforce authentication and confidentiality, RBAC/Web can work with existing Web technologies such as SSL, TLS, SHTTP, etc. In the current prototype implementation, RBAC/Web uses the authentication mechanism of NCSA HTTPd Web server for user name/password authentication.

RBAC/Web demonstrates that authorization management based on roles is possible on the Web. However, it supports a single administrative role called `admin`. In large organizations which have large numbers of roles and users, managing these roles and users, and their interrelationships, is a burdensome task that often is highly centralized and delegated to a small group of administrators. Some suitable decentralization of this administrative task is essential. Due to the centralized administrative model it will be difficult to maintain RBAC in large organizations with RBAC/Web.

Our work extends RBAC/Web with a role-based administrative model, called URA97 (User-Role Assignment '97), adding the following features:

- decentralized user-role administration (itself based on administrative roles)
- multiple administrative roles (for instance, SSO, DSO, PSO1, and PSO2)
- restrictions on which users can be assigned to or revoked from a role by whom
- flexible and powerful constraints through prerequisite conditions
- users can be assigned to senior and junior roles implicitly as well as explicitly

The administrative cost, complexity, errors and inconsistency of security administration can be reduced by simple and decentralized administration. Use of the URA97 model for managing user-role assignment can decentralize tedious details of RBAC administration without losing central control over overall system policy.

3 THE URA97 ADMINISTRATIVE MODEL OVERVIEW

Sandhu and Bhamidipati developed a model called URA97 in which RBAC is used to manage user-role assignment [SB97]. We now give a quick overview of the model. URA97 is defined in two steps dealing with granting a user membership in a role and revoking a user's membership.

3.1 URA97 Grant Model

URA97 imposes restrictions on which users can be added to a role by whom. URA97 requires a hierarchy (partial order) of roles (such as in Figure 1) and a hierarchy of administrative roles (such as in Figure 2). The set of roles and administrative roles are required to be disjoint. Senior roles are shown towards the top of the hierarchies and junior are to the bottom. Senior roles inherit permissions from junior roles. We write $x > y$ to denote x is senior to y with obvious extension to $x \geq y$. URA97 is defined in context of the well-known RBAC96 model. For our purpose, the notion of a hierarchy of roles from RBAC96 suffices. A complete definition of RBAC96 is available in [SCFY96].

The notion of a prerequisite condition is a key part of URA97. User-role assignment is authorized in URA97 by the *can-assign* relation.

Definition 1 A **prerequisite condition** is a boolean expression using the usual $\wedge$ and $\vee$ operators on terms of the form x and $\overline{x}$ where x is a regular role (i.e., $x \in R$). A prerequisite condition is evaluated for a user u by interpreting x to be true if $(\exists x' \geq x)(u, x') \in UA$ and $\overline{x}$ to be true if $(\forall x' \geq x)(u, x') \notin UA$. For a given set of roles R let CR denotes all possible prerequisite conditions that can be formed using the roles in R. $\Box$

Definition 2 The URA97 model controls user-role assignment by means of the relation *can-assign* $\subseteq AR \times CR \times 2^R$. $\Box$

[1] The Unix version using CGI for NCSA HTTPd server was released in 1996, while the Windows NT version has not yet been released.

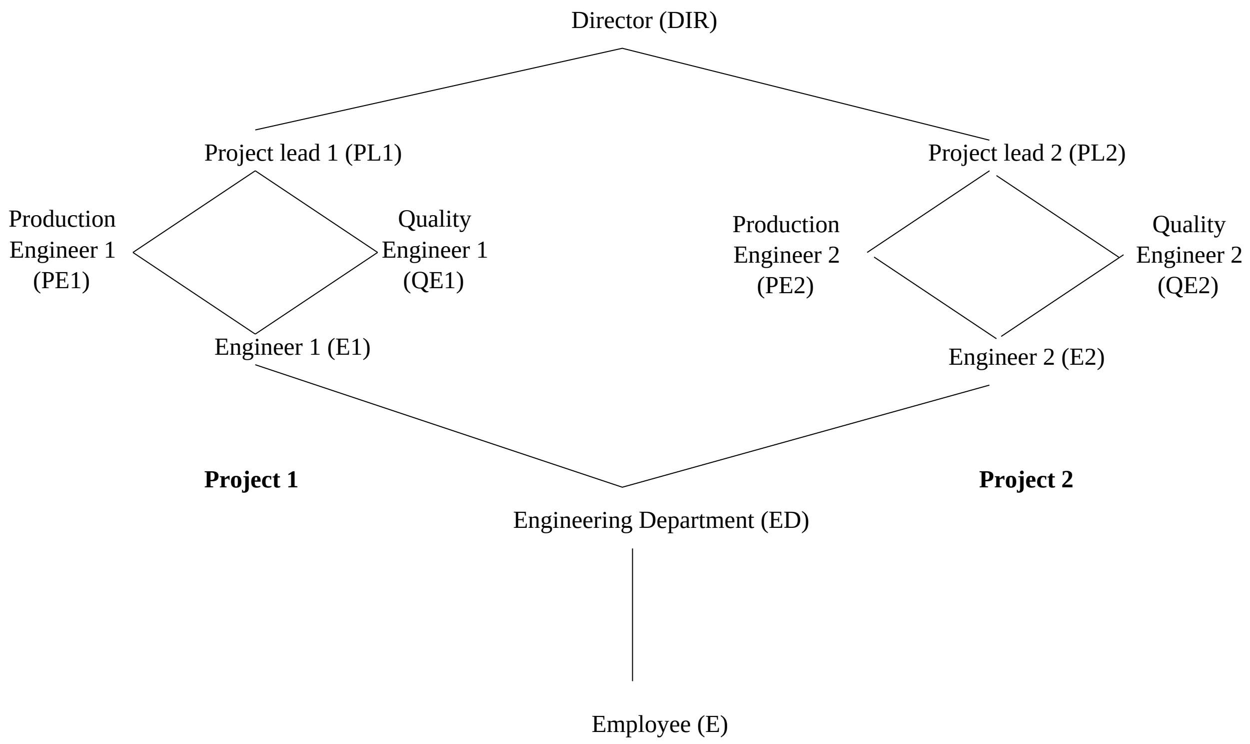

Figure 1: An Example Role Hierarchy

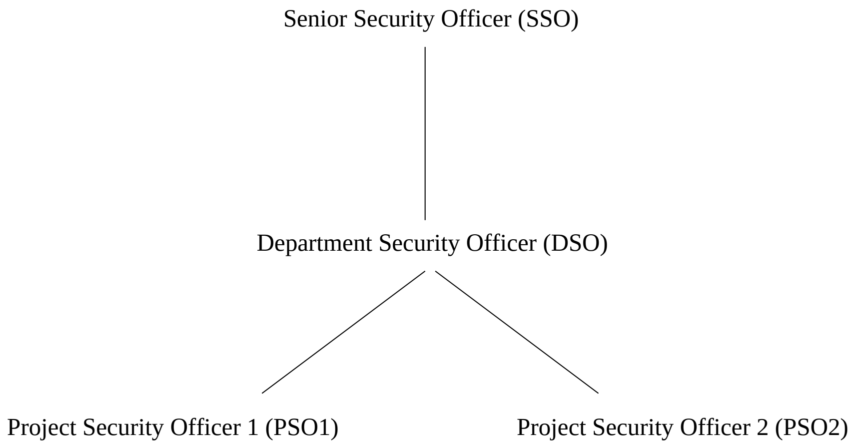

Figure 2: An Example Administrative Role Hierarchy

Admin. Role	Prereq. Condition	Role Range
PSO1	ED	[E1, E1]
PSO1	ED $\wedge$ $\overline{\text{QE1}}$	[PE1, PE1]
PSO1	ED $\wedge$ $\overline{\text{PE1}}$	[QE1, QE1]
PSO1	PE1 $\wedge$ QE1	[PL1, PL1]
PSO2	ED	[E2, E2]
PSO2	ED $\wedge$ $\overline{\text{QE2}}$	[PE2, PE2]
PSO2	ED $\wedge$ $\overline{\text{PE2}}$	[QE2, QE2]
PSO2	PE2 $\wedge$ QE2	[PL2, PL2]
DSO	ED	(ED, DIR)
SSO	E	[ED, ED]
SSO	ED	(ED, DIR]

Table 1: Example of *can-assign*

Admin. Role	Role Range
PSO1	[E1, PL1)
PSO2	[E2, PL2)
DSO	(ED, DIR)
SSO	[ED, DIR]

Table 2: Example of *can-revoke*

The meaning of *can-assign*$(x, y, \{a, b, c\})$ is that a member of the administrative role x (or a member of an administrative role that is senior to x) can assign a user whose current membership, or non-membership, in regular roles satisfies the prerequisite condition y to be a member of regular roles a, b or c.

3.1.1 Range Notation

For an example of the *can-assign* relation, we used the prerequisite conditions shown in Table 1. To identify a role range within the role hierarchy of Figure 1, we use the familiar closed and open interval notation [SB97].

Definition 3 Role sets are specified in the URA97 model by the notation below

$$\begin{array}{rcl} [x,y] & = & \{r \in R \mid x \geq r \wedge r \geq y\} \\ (x,y] & = & \{r \in R \mid x > r \wedge r \geq y\} \\ [x,y) & = & \{r \in R \mid x \geq r \wedge r > y\} \\ (x,y) & = & \{r \in R \mid x > r \wedge r > y\} \end{array}$$

□

3.1.2 Prerequisite Conditions

Let us consider the PSO1 tuples (analysis for PSO2 is very similar). The first tuple authorizes PSO1 to assign users with the prerequisite role ED into E1. The second one authorizes PSO1 to assign users with the prerequisite condition ED $\wedge$ $\overline{\text{QE1}}$ to PE1. Similarly, the third tuple authorizes PSO1 to assign users with the prerequisite condition ED $\wedge$ $\overline{\text{PE1}}$ to QE1. Taken together the second and third tuples authorize PSO1 to put a user who is a member of ED into one but not both of PE1 and QE1. This illustrates how mutually exclusive roles (for static separation of duty) can be enforced by URA97. PE1 and QE1 are mutually exclusive with respect to the power of PSO1. However, for the DSO and SSO these are not mutually exclusive. Hence, the notion of mutual exclusion is a relative one in URA97. The fourth tuple authorizes PSO1 to put a user who is a member of both PE1 and QE1 into PL1. Of course, a user could have become a member of both PE1 and QE1 only by actions of a more powerful administrator than PSO1.

3.2 URA97 Revoke Model

In classical discretionary access control the source (direct or indirect) of a permission and the identity of the revoker is typically taken into account in interpreting the revoke operation.[2] These issues do not arise in the same way for revocation of user-role assignment in RBAC. However, there are related subtleties that arise in RBAC concerning the interaction between granting and revocation of user-role membership and the role hierarchy.

3.2.1 The Can-Revoke Relation

Definition 4 The URA97 model controls user-role revocation by means of the relation *can-revoke* $\subseteq AR \times 2^R$. □

The meaning of *can-revoke*(x, Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a user from any regular role $y \in Y$. We say Y defines the *range of revocation*. Table 2 gives an example of the *can-revoke* relation.

3.2.2 Weak Revocation

Definition 5 Let us say a user U is an *explicit member* of role x if $(U, x) \in UA$, and that U is an *implicit member* of role x if for some $x' > x$, $(U, x') \in UA$. □

[2] This is true more in theory than practice, because many products and systems opt for a simpler semantics than implied by a strict owner-based discretionary viewpoint.

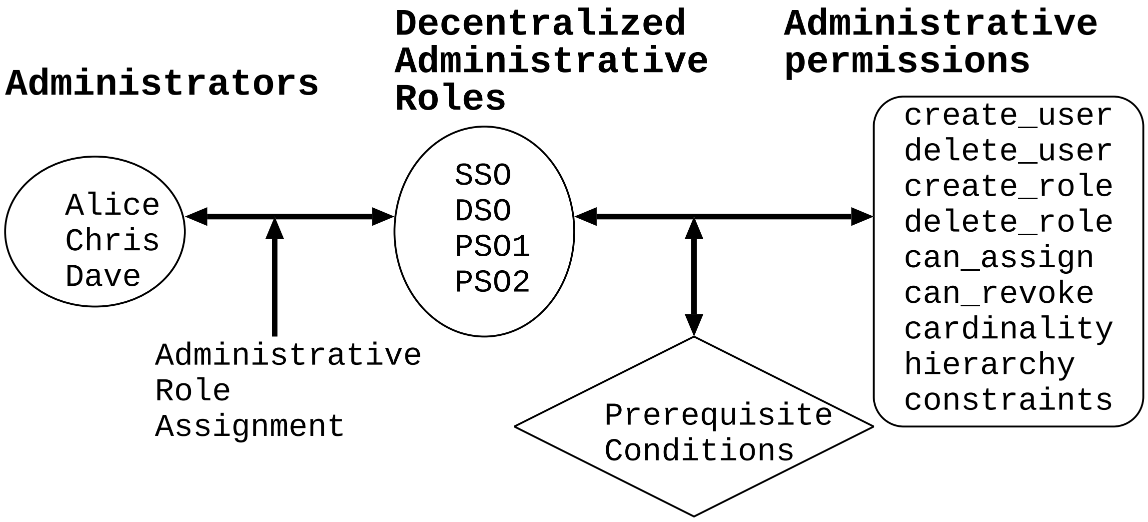

Figure 3: Decentralized Administration of URA97/WWW

Note that a user can simultaneously be an explicit and implicit member of a role,[3] however weak revocation has an impact only on explicit membership. It has the straightforward meaning stated below.

Definition 6 **[Weak Revocation Algorithm]**

1. Let Alice have a session with administrative roles $A = \{a_1, a_2, \ldots, a_k\}$, and let Alice try to weakly revoke Bob from role x.

2. If Bob is not an explicit member of x this operation has no effect, otherwise there are two cases.

   (a) There exists a *can-revoke* tuple (b, Y) such that there exists $a_i \in A, a_i \geq b$ and $x \in Y$.

   In this case Bob's explicit membership in x is revoked.

   (b) There does not exist a *can-revoke* tuple as identified above.

   In this case the weak revoke operation has no effect.

□

Revocation in URA97 has the following effect with respect to junior roles. Implicit membership in a role a is dependent on explicit membership in some senior role $b > a$. Therefore when explicit membership of a user is revoked from b, implicit membership is also automatically revoked on junior role a unless there is some other senior role $c > a$ in which the user continues to be an explicit member.[4]

3.2.3 Strong Revocation

Strong revocation in URA97 requires revocation of both explicit and implicit membership. Strong revocation of U's membership in x requires that U be removed not only from explicit membership in x, but also from explicit (or implicit) membership in all roles senior to x. Strong revocation therefore has a cascading effect upwards in the role hierarchy. However, strong revocation in URA97 takes effect only if all implied revocations upward in the role hierarchy are within the revocation range of the administrative roles that are active in a session.

Definition 7 **[Strong Revocation Algorithm]**

1. Let Alice have a session with administrative roles $A = \{a_1, a_2, \ldots, a_k\}$, and let Alice try to strongly revoke Bob from role x.

2. Find all roles $y \geq x$ and Bob is a member of y.

3. Weak revoke Bob from all such y as if Alice did this weak revoke.

[3] The NIST RBAC/Web implementation prohibits this from happening so a user cannot simultaneously be an explicit member of a senior and junior role on that basis that this would be redundant [BCF+97]. In URA97 the redundancy is an intrinsic part of the model. This redundancy is also useful in preserving membership in a junior role when a senior role is revoked.

[4] Models that do not allow redundant explicit membership in senior and junior roles will cause the user to loose membership in all junior roles when revoked from a senior role.

4. If any of the weak revokes fail then Alice's strong revoke has no effect otherwise all weak revokes succeed.[5]

4 IMPLEMENTATION OF THE URA97 MODEL ON THE WEB

In this section we describe our implementation of URA97 on the Web (URA97/WWW). The NIST RBAC/Web system conceptually consists of two parts: a Web server which enforces RBAC at the granularity of individual URL's and an administrative tool that allows a security administrator to define roles, role-role relationships and assign users to roles. The NIST implementation provides a centralized model for assignment of users to roles by actions of the security officer. To implement URA97 we only need to change the administrative tool. The enforcement of RBAC by the Web server remains unchanged.

Our implementation is illustrated in this section by an example of the role hierarchy of Figure 1, the administrative role hierarchy of Figure 2, the *can-assign* relation of Table 1, and the *can-revoke* relation of Table 2.

The communication mechanism between users and the URA97/WWW is basically the same as that used with regular Web servers. Figure 3 shows how decentralized administration works in the URA97/WWW. Administrative permissions are associated with administrative roles, and administrators are assigned to decentralized administrative roles, thereby acquiring the administrative role's permissions.

Suppose Alice is assigned to the SSO (Senior Security Officer) role in Figure 2 and wants access to the URA97/WWW her Web browser. When she requests access to URA97/WWW, the system requires her user ID and password for authentication. After the system checks her information, it shows all the roles available to her based on her assigned roles and role hierarchy of the system.

Because the URA97 model supports multiple administrative roles, there is an administrative role hierarchy, such as the one in Figure 2 for the system, and each administrative role can have different role range and prerequisite conditions.

In the above example, Alice would have SSO, DSO, PSO1, and PSO2 as her available administrative roles, shown in the left frame of Figure 4, since she is assigned to SSO, which is the highest role in the administrative role hierarchy of Figure 2. Then, she can activate one of those available roles as a session, and according to that activated administrative role, the proper prerequisite conditions are assigned to the user automatically. In other words, if Alice activates DSO from among her available roles, then the prerequisite condition of DSO (in our implementation, `prereq_dso`) is assigned to Alice automatically and goes with Alice's transaction until she changes her session to another administrative role.[6] In the above example, if Alice changes her session by selecting SSO then her prerequisite condition becomes `prereq_sso` automatically instead of `prereq_dso`.

The prerequisite conditions in URA97 provide flexible constraints, which make individual system administrators maintain the system efficiently and securely. In particular, we can enforce static separation of duty by configuring prerequisite conditions. NIST's RBAC/Web supports dynamic separation of duty which is outside the scope of URA97. Detailed steps of user-role assignment and user-role revocation using prerequisite conditions in the URA97/WWW are described in the following subsections.

4.1 Can_Assign in the URA97/WWW

Suppose Alice, a member of the administrative role SSO, wants to assign Bob, who has only a regular role E (employee) currently, to DIR (director) based on the prerequisite conditions in Table 1. According to the SSO tuples in the table, the administrative role SSO can add users in E to ED, and it can add users in ED to the roles in the range of (ED, DIR].[7] Therefore, the SSO must first enroll Bob in ED before he is enrolled any roles senior to ED.

Figure 4 shows Bob's current status with explicitly assigned role E. Since Alice is a member of the administrative role SSO, according to the administrative role hierarchy of Figure 2 she also has inherited administrative roles such as DSO, PSO1, and PSO2, as well as SSO. However, since only SSO can add users in E to ED, she needs to activate SSO. Currently E is the only explicitly assigned role for Bob, therefore only ED is assignable to Bob by SSO. If Alice activates an administrative role other than SSO, there would be no assignable role for Bob (since no role is assignable to Bob by other administrative roles except SSO).

[5] In some papers [SA98] two forms of strong revoke have been identified. These are called *drop* and *continue*. If one of the weak revokes fails the drop option stipulates that strong revocation has no effect, whereas with the continue option those weak revokes that are authorized will be performed.

[6] Although RBAC96 allows multiple (incomparable) administrative roles to be invoked in a single session, for simplicity our implementation restricts the user to only one administrative role at a time.

[7] For convenience we say that a role can do a certain operation whereas strictly speaking we should say that a user with that role can do that operation.

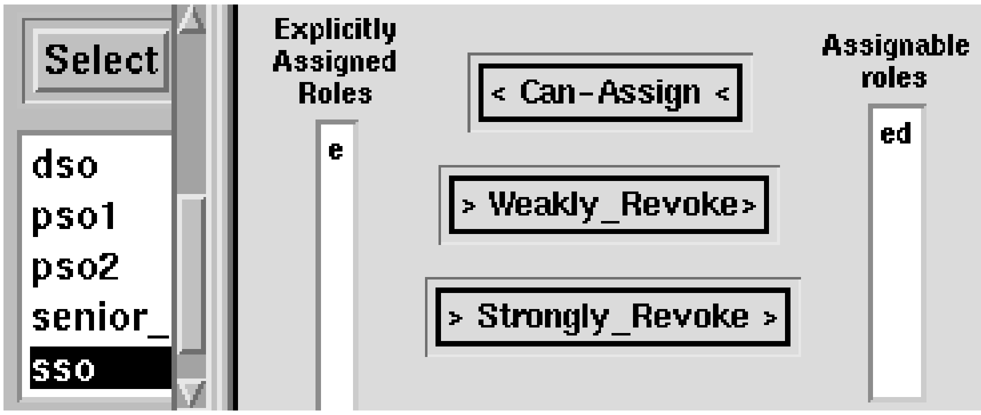

Figure 4: Bob's status with explicitly assigned role E

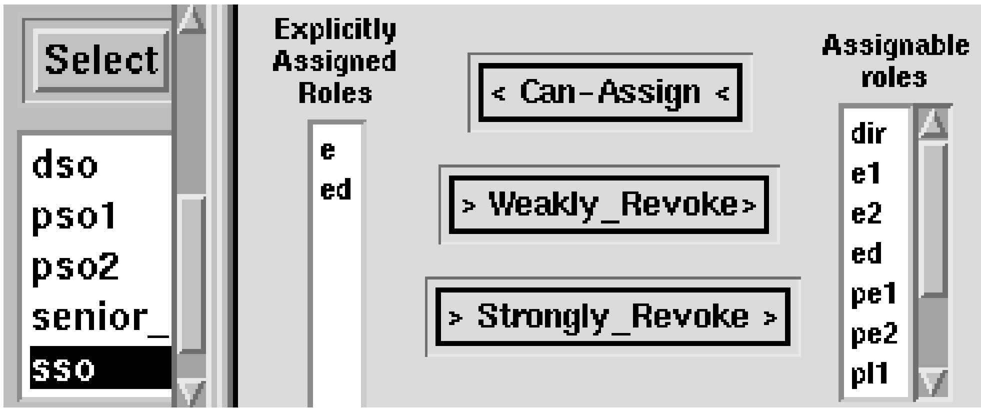

Figure 5: Assignable Roles for Bob by SSO after *can-assign* to ED

If Alice with the activated administrative role SSO selects ED from among Bob's assignable roles (in the above example, only ED is assignable) and clicks on the `Can_Assign` button, then the content changes as Figure 5. Now, Bob has the role ED as well as the role E. Therefore, according to the last tuple of the prerequisite conditions in Table 1, Alice with the activated role SSO can assign Bob, who has the prerequisite role ED, into roles within the role range (ED, DIR].

The assignable roles for Bob shown in the box depend on the activated administrative role by Alice and the prerequisite conditions of that role. For instance, if Alice selects PSO1 instead of SSO in the left frame of Figure 5, the assignable roles for Bob would be roles in the range [E1,PL1) based on the PSO1 tuples in Table 1.

Subsequently, if Alice with the activated PSO1 assigns Bob to PE1, the role QE1 is not assignable for Bob according to the third tuple in Table 1. However, if Alice activates an administrative role other than PSO1, let's say DSO, assignable roles for Bob are roles in the range (ED, DIR) based on the DSO tuple in Table 1.

4.2 User-Role Revocation in the URA97/WWW

We now turn to revocation of user-role membership.

4.2.1 Weak Revocation

Weak revocation has an impact only on explicit membership. Suppose Bob is an explicit member of PL1, PE1, PE2, ED and E1. If Alice with the activated administrative role PSO1 weakly revokes Bob's membership from E1, he continues to be a member of the senior roles to E1 and therefore can use the permission of E1. Figure 6 and Figure 7 show Bob's explicitly assigned roles before and after the weak revocation from E1 by PSO1.

Note that, in the above example, Alice should have enough power in the session to weakly revoke Bob's membership from his explicitly assigned roles. For instance, if Alice has activated PSO1 and then tries to weakly revoke Bob's membership from PL1, she is not allowed to proceed because PSO1 does not have the authority of weak revocation from PL1 according to the *can-revoke* role ranges in Table 2.

Therefore, if Alice wants to revoke Bob's explicit membership as well as implicit membership from E1 by weakly revocation, she needs to activate SSO and weakly revoke Bob's membership from E1, PE1, and PL1. This brings out the same result as the one after strong revocation from E1 by SSO. However, Alice does not need to revoke Bob's membership from ED and PE2, because they are not senior roles to E1 based on the role hierarchy of Figure 2.

4.2.2 Strong Revocation

Strong revocation in URA97 impacts not only on explicit membership but also on implicit membership. Suppose Bob is an explicit member of PL1, PE1, PE2, ED and E1. If Alice with the activated administrative role SSO strongly revokes Bob's membership from E1, then he is removed not only from explicit membership in E1, but also from explicit (or implicit) membership in all roles senior to E1. Actually, after the strong revocation from E1, Bob has been removed from PL1, PE1 as well as E1. However, he still has a membership of ED and PE2, because they are not senior roles to E1 based on the role hierarchy of Figure 2. This brings out the same result as the one after weak revocation from E1, PE1, and PL1 by SSO. Figure 8 and Figure 9 show Bob's explicitly assigned roles before and after the strong revocation from E1 by SSO.

Note that strong revocation has a cascading effect upwards in the role hierarchy. Therefore, all implied revocations upward in the role hierarchy should be within the revocation range of the administrative roles that are active in a session. For instance, if Alice activates PSO1 and tries to strongly revokes Bob's membership from PL1, she is not allowed to proceed because PL1 is out of the PSO1's *can-revoke* range in Table 2.

5 RELATED WORK

The hyperDRIVE system [Bar97], which was developed by the Internal Revenue Service, is programmed in Java to provide a consistent, integrated, auditable, manageable RBAC implementation, which is employed by multiple servers, clients, and applications. The implementation includes a Java-capable and SSL-enabled Web browser, an LDAP[8] server, an Object Request Broker, and the hyperDRIVE client Java applet.

The hyperDRIVE system makes it possible for servers, applications, and active objects with capabilities and facilities to consult the LDAP-hosted RBAC data. Through this consultation, the client's requests are accepted or rejected to support secure communication between clients and servers based on RBAC. The access includes location, methods, and modes of access to and of network computing resources. The authorization is the mapping of one or more operations or

[8]Lightweight Directory Access Protocol defines a relatively simple protocol for updating and searching directories running over TCP/IP.

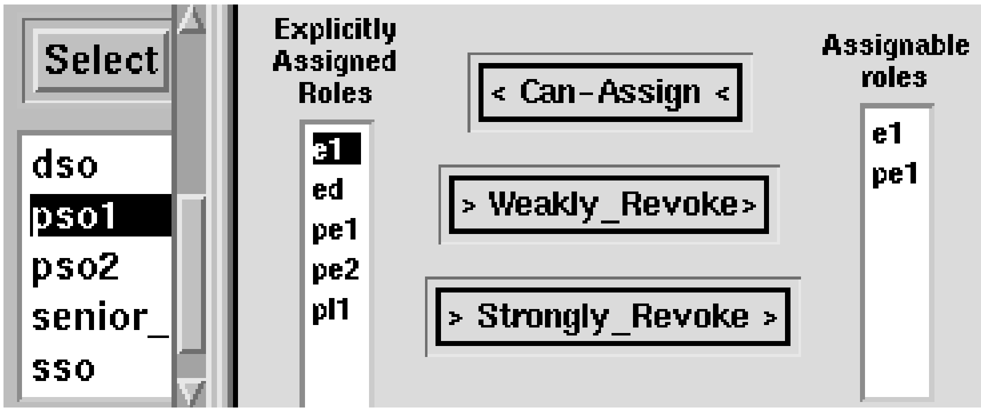

Figure 6: Before Weak Revocation from E1 by PSO1

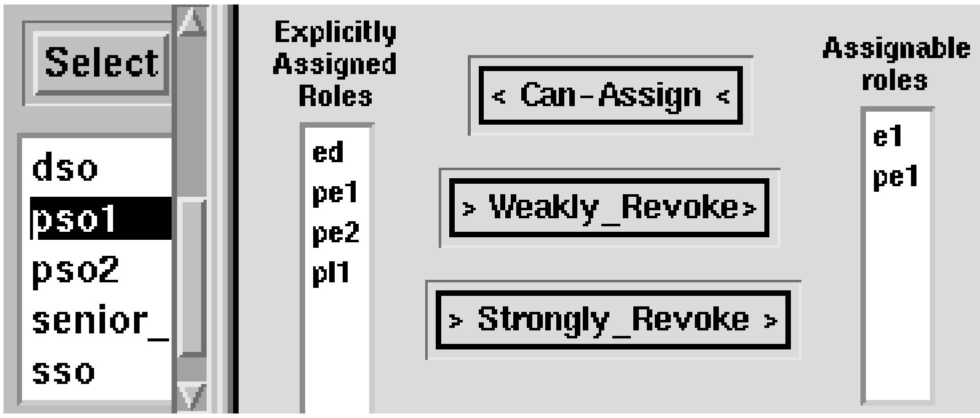

Figure 7: After Weak Revocation from E1 by PSO1

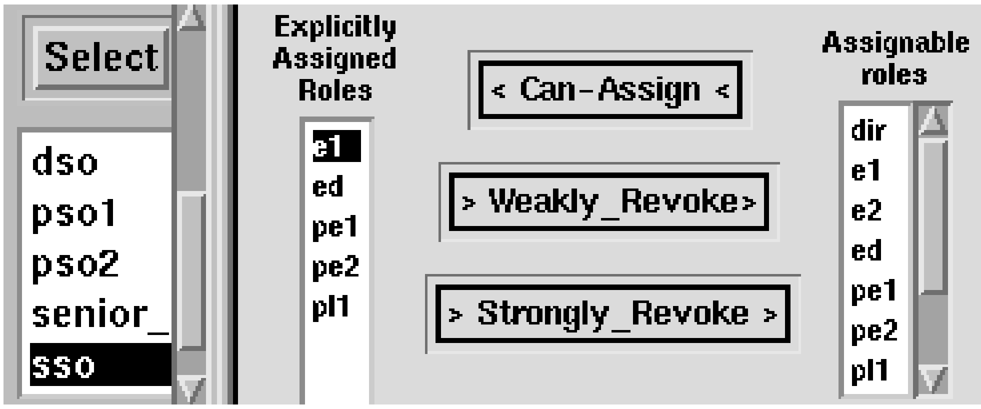

Figure 8: Before Strong Revocation from E1 by SSO

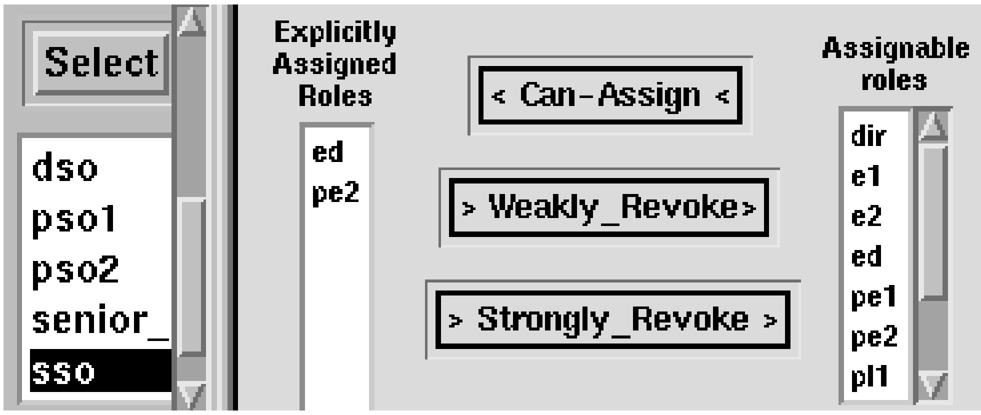

Figure 9: After Strong Revocation from E1 by SSO

privileges to a defined role.

The hyperDRIVE approach does not provide a model or guidance with respect to the system administrators' view to achieve RBAC. In other words, it is hard for system administrators to set the constraints, role hierarchy, and administer many roles in a huge system controlled by hyperDRIVE because the approach focuses on the user's transaction.

On the other hand, [TC97] suggests I-RBAC implementation with agents: coordination agents, task agents, and database agents. The agents are active network objects that implement the different security procedures by checking user's authorizations for using resources within an Intranet.

The main advantage of this approach is that there are local and global network objects in the system. A local object has an identity known only with the corresponding local server, while a global network object has a unique identity known throughout the Internet. Therefore, when a network object with a given role wants to access Intranet resources, the system checks its role in the global-role database. If the object's global role exists in the global-role database, then the system derives its global permissions and uses the permission domain tables in the local-role databases, which contain the access information of the local server's network objects within an Intranet, to verify the global role's derived local authorizations.

However, there is a consistency problem between roles. For instance, if we have hundreds of roles and thousands of permissions in the system, it is very difficult to keep the consistency by reflecting new security requirements between global network objects and local network objects.

Recently, Siemens Nixdorf released TrustedWeb which supports role-based access control for Web contents and applications as well as security services such as mutual authentication, integrity and confidentiality for Intranets. The system which combines elements from both Sieman's SESAME [PP95] and Kerberos [Neu94] provides a single list of users on its central domain security server and assigns roles to users. Therefore, access to the individual Web servers in the Intranet is controlled based on the role of the user rather the identity of the user. However, to use TrustedWeb, the client's browser needs specific software installed in the client's machine to communicate with the TrustedWeb servers in the Intranet while the URA97/WWW requires only Web browsers on the client side.

6 CONCLUSION

In this paper we have described the URA97/WWW extended from RBAC/Web, developed by NIST, to decentralize the details of RBAC system administration on the Web without loosing central control over the system policy. We have implemented the URA97/WWW with NCSA Web server and CGI in Perl for UNIX environment and made the demonstration available on the Web. URA97/WWW focuses on the assignment of users to roles and their removal. For comprehensive administration, we can use other components of ARBAC97 [SBC+97] (administrative RBAC '97) with URA97 such as PRA97 (permission-role assignment '97), and RRA97 (role-role assignment '97).

References

[Bar97] Larry S. Bartz. hyperDRIVE: Leveraging LDAP to implement rbac on the web. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, pages 69–74. ACM, Fairfax, VA, November 6-7 1997.

[BCF+97] John Barkley, Anthony Cincotta, David Ferraiolo, Serban Gavrilla, and Richard Kuhn. Role based access control for the World Wide Web. Technical report, National Institute of Standards and Technology, 1997.

[FB97] David Ferraiolo and John Barkley. Specifying and managing role-based access control within a corporate intranet. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, pages 77–82. ACM, Fairfax, VA, November 6-7 1997.

[FCK95] David Ferraiolo, Janet Cugini, and Richard Kuhn. Role-based access control (RBAC): Features and motivations. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 241–48, New Orleans, LA, December 11-15 1995.

[FK92] David Ferraiolo and Richard Kuhn. Role-based access controls. In *Proceedings of 15th NIST-NCSC National Computer Security Conference*, pages 554–563, Baltimore, MD, October 13-16 1992.

[GI96] Luigi Guiri and Pietro Iglio. A formal model for role-based access control with constraints. In *Proceedings of IEEE Computer Security Foundations Workshop 9*, pages 136–145, Kenmare, Ireland, June 1996.

[Gui95] Luigi Guiri. A new model for role-based access control. In *Proceedings of 11th Annual Computer Security Application Conference*, pages 249–255, New Orleans, LA, December 11-15 1995.

[HDT95] M.-Y. Hu, S.A. Demurjian, and T.C. Ting. User-role based security in the ADAM object-oriented design and analyses environment. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, *Database Security VIII: Status and Prospects*. North-Holland, 1995.

[MD94] Imtiaz Mohammed and David M. Dilts. Design for dynamic user-role-based security. *Computers & Security*, 13(8):661–671, 1994.

[Neu94] B. Clifford Neuman. Using Kerberos for authentication on computer networks. *IEEE Communications*, 32(9), 1994.

[NO95] Matunda Nyanchama and Sylvia Osborn. Access rights administration in role-based security systems. In J. Biskup, M. Morgernstern, and C. Landwehr, editors, *Database Security VIII: Status and Prospects*. North-Holland, 1995.

[PP95] Tom Parker and Denis Pinkas. *SESAME V4 - OVERVIEW*, December 1995. SESAME Technology version 4.

[SA98] Ravi Sandhu and Gail-Joon Ahn. Decentralized group hieraches in unix: An experiment and lessons learned. In *Proceedings of 21st NIST-NCSC National Information Systems Security Conference*, Arlington, VA, October 5-8 1998.

[San97] Ravi Sandhu. Rationale for the RBAC96 family of access control models. In *Proceedings of the 1st ACM Workshop on Role-Based Access Control*. ACM, 1997.

[SB97] Ravi Sandhu and Venkata Bhamidipati. The URA97 model for role-based administration of user-role assignment. In T. Y. Lin and Xiaolei Qian, editors, *Database Security XI: Status and Prospects*. North-Holland, 1997.

[SBC+97] Ravi Sandhu, Venkata Bhamidipati, Edward Coyne, Srinivas Ganta, and Charles Youman. The ARBAC97 model for role-based administration of roles: Preliminary description and outline. In *Proceedings of 2nd ACM Workshop on Role-Based Access Control*, pages 41–50. ACM, Fairfax, VA, November 6-7 1997.

[SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, February 1996.

[TC97] Zahir Tari and Shun-Wu Chan. A role-based access control for intranet security. *IEEE Internet Computing*, pages 24–34, September/October 1997.

[vSvdM94] S. H. von Solms and Isak van der Merwe. The management of computer security profiles using a role-oriented approach. *Computers & Security*, 13(8):673–680, 1994.

[YCS97] Charles Youman, Ed Coyne, and Ravi Sandhu, editors. *Proceedings of the 1st ACM Workshop on Role-Based Access Control, Nov 31-Dec. 1, 1995*. ACM, 1997.