Extended Privilege Inheritance in RBAC

M.A.C. Dekker
Security group, TNO ICT
The Netherlands

J.G. Cederquist
SQIG-IT, IST, TU Lisbon
Portugal

J. Crampton
ISG, Royal Holloway
United Kingdom

S. Etalle
DIES, University of Twente
The Netherlands

ABSTRACT

In existing RBAC literature, administrative privileges are inherited just like ordinary user privileges. We argue that from a security viewpoint this is too restrictive, and we believe that a more flexible approach can be very useful in practice. We define an ordering on the set of administrative privileges, enabling us to extend the standard privilege inheritance relation in a natural way. This means that if a user has a particular administrative privilege, then she is also implicitly authorized for weaker administrative privileges. We prove the non-trivial result that it is possible to decide whether one administrative privilege is weaker than another and show how this result can be used to decide administrative requests in an RBAC security monitor.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls

Keywords

Access control, RBAC, Administrative privileges

1. INTRODUCTION

Role-based access control (RBAC) [6] is a non-discretionary access control mechanism that simplifies the assignment of access rights to users. The basic idea is that while there are many access rights and users, rights and users can be grouped using a relatively small number of roles, ordered in a *role hierarchy*. In practice however, an RBAC system in a large enterprise may involve thousands of roles [4]. Keeping the access rights and roles up to date with changes in the enterprise may be a too big task for a single administrator. The usual approach to this problem is to divide the work and to allow delegation of part of the administrator's authority to other users. It is convenient to make this delegation mechanism flexible, in order to reduce the likelihood of bottlenecks and (administrative) users sharing keys or passwords that should remain secret. On the other hand, the delegation mechanism should be *safe*: users should not obtain rights other than those explicitly delegated by the administrator.

Several lines of research address the problem of delegation of administrative privileges in RBAC systems; Ferraiolo et al. [4] compare some of the different approaches. The main issue in these lines of research is how to model *administrative privileges*, as opposed to the ordinary *user privileges*, and to decide who should have them. In ARBAC [5] administrative privileges are assigned to a separate hierarchy of administrative roles and defined by specifying a range of roles that can be changed. Crampton and Loizou [3] take a more general approach, by using the same hierarchy for both the administrative privileges and the ordinary user privileges. Using the concept of *administrative scope*, they define which roles should have administrative privileges over other roles. In the Role-Control Center [4], administrative privileges over roles are defined in terms of *views*, which are subsets of the role-hierarchy, and they can only be assigned to users assigned to these roles.

In existing RBAC literature [2, 3, 4, 5, 7], administrative privileges are inherited just like ordinary user privileges. We argue that this approach is more restrictive than necessary for safety: consider a simple setting where an administrator has delegated the authority to some user u to assign a user u' to a *high* role in the role-hierarchy. When user u uses this authority, user u' becomes assigned to the high role, and consequently u' can also play *lower* roles. There is no security motivation for not letting user u assign user u' directly to (one of) the lower roles. User u' could play the lower roles anyway. However, in standard RBAC, administrative privileges are not interpreted in this way.

In this paper, we define an ordering on the administrative privileges, enabling us to extend the standard privilege inheritance relation in a natural way. This means that if a user has a particular administrative privilege, then she is also implicitly authorized for weaker administrative privileges. Basically, this allows for a more flexible use of administrative privileges. We argue that decentralized management of RBAC becomes more flexible with this extension.

ASIACCS'07, March 20-22, 2007, Singapore.

2. ADMINISTRATIVE PRIVILEGES

Privileges can be divided into *user privileges* and *administrative privileges* [6]. While user privileges allow actions on objects (such as printing files or viewing records), administrative privileges, allow actions on the RBAC state itself, e.g. adding an edge from one role to another. Here we assume that user privileges form a *finite* set of atomic privileges, denoted by Q, that corresponds to a finite set of actions on objects. On the other hand, the set of administrative privileges is necessarily infinite, because privileges about administrative privileges are included as well. We formalize the full set of privileges by defining a *grammar* that encompasses both user privileges and administrative privileges.

Definition 1 (Privilege Grammar). *Given the sets U of users, R of roles and Q of user privileges, the set of all privileges P is defined by the following grammar:*

$$p \ ::= \ q \ | addUser(u, r) \ | \ addEdge(r, r') \ | \ addPrivilege(r, p),$$

where $u \in U$, $q \in Q$ and $r, r' \in R$.

Each administrative privilege corresponds to an administrative action. The privilege $addUser(u, r)$ allows the action of adding a member u to the role r. The privilege $addEdge(r_1, r_2)$ allows the action of adding an edge from role r_1 to r_2. The construct *addPrivilege* is a grammatical connective and consequently - as mentioned above - the set P is infinite despite the fact that the sets of users, roles and user privileges are all finite. In the existing literature the number of administrative levels (in other words, the number of nestings of the *addPrivilege* connective) is sometimes restricted to one [6] or to two levels [7]. We agree that administrative privileges with multiple levels of administration might not be useful in some implementations. However, here we take a general approach, and we let the administrators choose which administrative privileges to use. We should mention also that most existing models constrain the roles that can have administrative privileges, for example to prevent low roles obtaining privileges to change membership of higher roles [3]. We do not make choices with respect to such constraints.

2.1 Extended Privilege Inheritance

An RBAC state is denoted by a triple (UA, RH, PA), containing the user-role assignments, the edges between roles and the privilege assignments to roles. A well-known feature of RBAC is *privilege inheritance* [6], by which a role has the privileges to which it is explicitly assigned, and additionally, the privileges of lower roles.

Definition 2 (Standard Privilege Inheritance). *Let (UA, RH, PA) denote an RBAC state and let $\geqslant$ denote the reflexive transitive closure of RH, we say that a role r has the privilege p, denoted by $r \rightsquigarrow p$, iff*

$$r \geqslant r' \text{ and } (r', p) \in PA \text{ for some } r' \in R.$$

We argue that the standard privilege inheritance is inadequate for administrative privileges. Take for example a role r with the privilege to add an edge e from r_2 to r_3. The role r does not have the privilege to add an edge from r_2 to any role below r_3, nor the privilege to add an edge from any role above r_2 to r_3. However, from a security point of view this makes no sense, because with edge e in place there would be anyway a path to roles below r_3, or a path from roles above r_2. The standard RBAC privilege inheritance however does not capture this, and treats administrative privileges like ordinary user privileges. In Figure 1 we show the six different cases where administrative privileges yield weaker administrative privileges. For example, in Figure 1a, the (administrative) privilege to assign user u to role r_1 (the dashed edge) is stronger than the privilege to assign user u to role r_2. In Figure 1b, the privilege to add an edge between r_1 and r_2 (the dashed arrow) is stronger than the privilege to add only user u to role r_2. And so on. We formalize this ordering by the following definition.

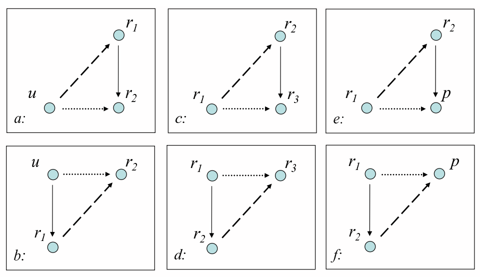

Figure 1: The right to add the dashed edge is stronger than the right to add the dotted edge.

Definition 3 (Privilege Ordering). *Let (UA, RH, PA) be an RBAC state, let p, p_1, p_2 be privileges in P, let Q be the subset of user privileges in P, and let r_1, r_2, r_3, r_4 be roles in R. We define the relation $\rightarrow$ as the smallest relation satisfying:*

1. *$p \rightarrow p$, if $p \in Q$*
2. *$addUser(u, r_1) \rightarrow addUser(u, r_2)$, if $r_1 \geqslant r_2$*
3. *$addEdge(r_1, r_2) \rightarrow addUser(u, r_3)$, if $r_2 \geqslant r_3$ and $(u, r_1) \in UA$*
4. *$addEdge(r_2, r_3) \rightarrow addEdge(r_1, r_4)$, if $r_1 \geqslant r_2$ and $r_3 \geqslant r_4$*
5. *$addEdge(r_2, r_3) \rightarrow addPrivilege(r_1, p_2)$, if $r_1 \geqslant r_2$, $r_3 \geqslant r_4$, $(r_4, p_1) \in PA$ and $p_1 \rightarrow p_2$*
6. *$addPrivilege(r_2, p_1) \rightarrow addPrivilege(r_1, p_2)$, if $r_1 \geqslant r_2$ and $p_1 \rightarrow p_2$*

The ordering $\rightarrow$ is both reflexive and transitive.

The ordering of privileges yields an extension of the standard privilege inheritance relation.

Definition 4 (Extended Privilege Inheritance). *Let (UA, RH, PA) be an RBAC state, let r be a role in R and p be a role in P, and let $\rightsquigarrow$ denote the* standard *privilege inheritance, reported in Definition 2. We say that the extended privilege inheritance $r \rightsquigarrow^* p$ holds iff*

$$r \rightsquigarrow p' \text{ and } p' \rightarrow p, \text{ for some } p' \in P.$$

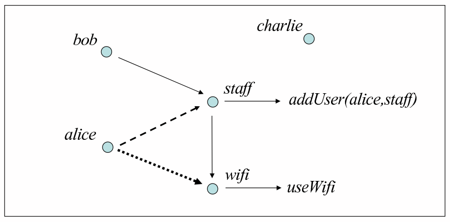

Figure 2: A practical example of the use of the extended inheritance relation.

The extended privilege inheritance relation is useful because it allows users, with administrative privileges, to be implicitly authorized for weaker administrative privileges. Thereby, it gives administrative users the possibility to perform safer administrative operations than the ones originally allowed. We now give a practical example of its usage.

Example 1 (Visiting Researcher). *Charlie, the security administrator, gives the staff the privilege to add visiting researchers to the staff role. There is also a role below staff called wifi, with the privilege to use the wireless network. Alice is a visiting researcher and Bob is a member of the staff. Alice only needs access to the wifi network, so Bob would like Alice to use the wifi role. Charlie (who just left) did not provide this privilege explicitly to the staff. This scenario is illustrated in Figure 2.*

In the standard RBAC model, Bob can only assign Alice to the staff role. Given the fact that Alice only needs wifi access, Bob urges Alice to apply the principle of least privilege, and to activate only the wifi role. However, Bob can only hope that Alice does so. With the extended privilege inheritance relation Bob can assign Alice to the wifi role because *of his privilege to add users to the staff role. In a way, instead of preaching the principle of least privilege to Alice, Bob applies it for her.*

2.2 Tractability

We now show that the extended privilege inheritance relation (Definition 4) is tractable. This result is not immediate, since the full set P of privileges is infinite. For instance, a naive forward search does not necessarily terminate. The proof also indicates how a decision algorithm, deciding which privileges are to be given to which roles, can be implemented at an RBAC security monitor.

Theorem 1 (Decidability).
Given an RBAC state, a role r and a privilege p in P, there is an algorithm to determine whether $r \rightsquigarrow^* p$.

Proof. (Sketch) The standard privilege inheritance $\rightsquigarrow$ is decidable, yielding a finite set of privileges p' inherited by r. Now for each privilege p' we need to check whether $p' \rightarrow p$. It can be proven that, given two privileges p', p, it is decidable whether $p' \rightarrow p$. The proof is by structural induction over p. □

We would like to mention here that it is not decidable to determine, given p', the set of all privileges p, such that $p' \rightarrow p$, despite the fact that the UA, RH and PA are finite. Because of space limitations, we refer to an extended paper[1] for these details and the details of the proof of Theorem 1.

3. CONCLUSION

With this work we make a contribution to the design of flexible administration models for RBAC. Flexible administration is important to cut the cost of maintenance and to enable the RBAC system to adapt to changing circumstances. Concretely, our contribution is an extension of the standard RBAC privilege inheritance relation. We defined an ordering on administrative privileges, that enabled us to extend the standard privilege inheritance relation in the natural way. This means that if a user has a particular administrative privilege, then she is also implicitly authorized for weaker administrative privileges. We showed that this relation is tractable. Our extension can be seen as an application of the *principle of least privilege* at the level of administration.

A number of improvements and additions can be made. We do not express privileges such as $\forall r.addEdge(r', r))$, which may be useful in practice, but we believe that special care is needed to deal with quantifiers. Finally, as we do not make a particular choice regarding constraints on the administrative privileges, it would be interesting to investigate how our results can be combined with, for example, the work by Crampton and Loizou [3] or that of Bandmann et al. [1].

4. REFERENCES

[1] O. L. Bandmann, B. S. Firozabadi, and M. Dam. Constrained delegation. In M. Abadi and S. M. Bellovin, editors, *Proc. of the Symp. on Security and Privacy (S&P)*, pages 131–140. IEEE Computer Society Press, 2002.

[2] E. Barka and R. S. Sandhu. Framework for role-based delegation models. In J. Epstein, L. Notargiacomo, and R. Anderson, editors, *Annual Computer Security Applications Conference (ACSAC)*, pages 168–176. IEEE Computer Society Press, 2000.

[3] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *Transactions on Information System Security (TISSEC)*, 6(2):201–231, 2003.

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

[5] R. S. Sandhu, V. Bhamidipati, and Q. Munawer. The ARBAC97 model for role-based administration of roles. *Transactions on Information and System Security (TISSEC)*, 2(1):105–135, 1999.

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

[7] X. Zhang, S. Oh, and R. S. Sandhu. PBDM: a flexible delegation model in RBAC. In D. Ferraiolo, editor, *Proc. of the Symp. on Access Control Models and Technologies (SACMAT)*, pages 149–157. ACM Press, 2003.

[1]An extended version of this paper is available from http://eprints.eemcs.utwente.nl