POSTER: Security Enhanced Administrative Role Based Access Control Models

Rajkumar P.V.
Department of Information Systems
Texas Southern University
Houston, TX, USA.
rajkumar.pv@gmail.com

Ravi Sandhu
Institute for Cyber Security &
Department of Computer Science
University of Texas at San Antonio, TX, USA.
ravi.sandhu@utsa.edu

ABSTRACT
Role Based Access Control (RBAC) is a widely implemented protection mechanism in operating systems, software applications, and cloud platforms. The existing administrative models of RBAC provides system administrators the privilege to change users roles and roles permissions that are under their administrative limits. While such privileges are indispensable part of the system, monitoring the execution of administrative privileges are often necessary to protect and ensure system security. However, the current administrative models does not have sufficient monitoring features within the model. In this work, we present a preliminary idea of integrating obligations into an Administrative RBAC model to regulate the usage of administrative privileges. Obligations would serve as an accountability mechanism within the administrative RBAC models. We believe that this work would initiate further discussions and development of RBAC administrative models that would improve the accountability of system administrators.

Keywords
Operating System; Security; Access Control; RBAC; Administration.

1. INTRODUCTION
Role Based Access Control (RBAC) model associates system permissions to roles and assigns users to appropriate roles according to their functions and responsibilities within the organization. If a users role is active, then he/she will get the permissions associated with the role. Thereby the users and permissions are decoupled through the roles. The RBAC is policy neutral model, further, it closely matches with the functional structure of organizations [11]. These feature made the RBAC as a prominent security mechanism in various Database Management Systems, Operating Systems, and number of application software used in large organizations like banks. The RBAC model is also implemented in various cloud services like Microsoft Azure and OpenStack. Today, the RBAC remains as an important mechanism to provide application security as well as to protect user privacy. Therefore administration of RBAC is an important practical issue.

Main tasks of RBAC administrators are associating permissions with roles, assigning user to roles, and defining hierarchy of roles as per the organizations policies. These administrative rights are different from typical administrative privilege and root users in commodity operating systems. The RBAC administrators are not necessarily permitted to read or write files within the system. Application user rights are often referred as inert rights. Such segregation of inert rights and administrative rights improves system security. However, usage of administrative rights in RBAC are not fully subjected to accountability.

Organizations with hundreds of roles and thousands of permissions are not uncommon in banking and finance sectors. Therefore, most of the RBAC administrative models support decentralized delegation of administrative rights to multiple administrative users. In general, the administrative rights of large RBAC systems are distributed among multiple administrators and same rights may be given to multiple administrators. For example, multiple administrators may have the right to add users to Backup and Recovery role which has rights to read and write sensitive files. If an administrator has such a right he can add/delete users to the roles without subjecting himself for any accountability in the future. Such provisions have potential security risk. In this work, we present our preliminary idea of associating obligations with execution of administrative rights such that accountability can be brought into the context where ever it is needed.

General logging and auditing mechanisms currently available in most of the applications helps in enforcing certain level of accountability of all users including system administrators. Such mechanisms can be configured to log user actions at various levels of granularity. If any untoward event or breach is identified then the analysis of past logs in the system would provide certain level of details about the breach. On the other hand, integration of obligations with the execution of administrative rights would provide a more systematic approach to manage the usage of RBAC administrative rights. Obligations are the actions that are often need to be completed before execution of rights and they would work as a proactive protection mechanism and may prevent certain breaches before they occur in the system. Sections 3.1, 3.2 and 3.3 provides the details on administrative obligations.
Section 2 presents the brief overview of administrative models available for RBAC. Section 3 presents the Security Enhanced Administrative RBAC model. Section 4 presents a discussion and conclusion.

2. RELATED WORK

The Administrative Role Based Access Control (ARBAC97) model introduced the concept of administrating RBAC [10]. The ARBAC97 defined the set of administrative roles and the set of administrative permissions which are functionally different from application permissions and roles. The ARBAC97 model uses the notions of user role assignment, permission role assignments, and role hierarchies defined in [11]. The ARBAC99 model introduced concept of mobile and immobile users [12]. The ARBAC02 model decoupled the administrative user permissions from roles and role hierarchy [8]. This model takes the organizational structure as a basis and develops the administrative roles and permissions as per the structure. The SARBAC model defined the concept of administrative scope and used the scope as a precondition to grant administrative rights [2]. Other Administrative RBAC models [5], [7], [1], and [3] also define who can get admin permissions. To the best of our knowledge, accountability and obligation aspects of admin users are not addressed in the ARBAC literature. However, obligations are well studied concept [4], [6] and the authorization of obligations using RBAC model is studied [9]. In this work, we present the notion of administrative obligations and their integration into administration of RBAC model to enhance the system security.

3. SECURITY ENHANCED ARBAC

The Administrative RBAC model has three components: (1) user role assignment, (2) permission role assignment, and (3) role hierarchy or role-role assignment. A change in any one of the above relation enables or disables application user rights. We propose to integration obligations such that if an administrative action changes user permissions the model will take notify or have to take an approval from other administrators beforehand.

3.1 Administrative Obligations

Administrative obligations are set of actions that the administrators have to perform before executing admin rights. We define three obligatory actions that we deem relevant for ARBAC97 model. The Report action is designed to notify a specified set of co-admins before executing an admin right. The Log action is meant to store the admin, users, and roles affected due to execution of the right. The Seek Approval action requires the admin user to obtain approval from a specified set of co-admins. The first two obligations are passive and they do not obstruct the execution of right. Whereas the third one is an active obligation which requires an approval from co-admins before executing the right.

Definition 3.1 The Report is defined as a relation of the form Report ⊆ AU × {RL} × U × {CAU} where, (1) AU is the administrative user who executes the right, (2) RL is the set of relations that the execution of the admin right would change in the RBAC system, (3) L is the secure location where the log will be entered, and (4) T is the time stamp.

Definition 3.3 The Seek Approval is defined as a relation of the form Seek Approval ⊆ AU × {RL} × U × {CAU} where, (1) AU is the administrative user who executes the right, (2) RL is the set of relations that the execution of the admin right would change in the RBAC system, and (3) CAU is the set of co-administrators whose approval is needed before executing the right.

3.2 Security Enhanced User Role Assignment

Modifications in the user role assignments would directly change the set of permissions given to the individual application user; other users’ permissions would remain intact. The URA97 model defines can_assign and can_revoke relational construct that assigns administrative rights to system admins to add/remove users into roles. These constructs specify the set of permitted administrative rights to the admins. We define assign and revoke relations that specify the administrative obligations required to be fulfilled while executing the admin rights permitted by can_assign and can_revoke.

Definition 3.4 The assign is defined as a relation of the form assign ⊆ AU × U × {R} × {AO} where, AU is the administrative user who tries to add the user U to the set of roles R and AO ⊆ {Report, Log, Seek Approval}.

Definition 3.5 The revoke is defined as a relation of the form revoke ⊆ AU × U × {R} × {AO} where, AU is the administrative user who tries to remove the user U to the set of roles R and AO ⊆ {Report, Log, Seek Approval}.

3.3 Security Enhanced Permission Role Assignment

The permission role assignment model PRA97 is a dual of user role assignment URA97. The PRA97 defines can_assignp and can_revokep constructs to associate and disassociate set of permissions with roles. We define the assignp and revokep relations to integrate the obligations with execution of admin rights permitted by can_assignp and can_revokep.

Definition 3.6 The assignp is defined as a relation of the form assignp ⊆ AU × P × R × {AO} where, AU is the administrative user who tries to add the set of permissions P to the role R and AO ⊆ {Report, Log, Seek Approval}.

Definition 3.7 The revokep is defined as a relation of the form revokep ⊆ AU × P × R × {AO} where, AU is the administrative user who tries to remove the set of permissions P from the role R and AO ⊆ {Report, Log, Seek Approval}.

Change in a permission role relation would impact the permissions of all users who are assigned with the role. Such changes require certain level of organizational policy decisions, therefore, Seek Approval type obligations seems more useful than others. However, it requires further study with respected to established security practices and usability aspects. Execution of admin rights that change the role permission relation would affect large number of users’ permissions, therefore, administrative obligations for such admin rights require further analysis on deciding the appropriate set of obligation types as well as selecting the set of co-admins for each type.
3.4 Example
Let us assume that the RBAC system is administrated using ARBAC97 model and the set of system administrators \( SA = \{ A_1, A_2, A_3 \} \). The following example shows the specification of administrative obligation policy “whenever an administrator adds new users to Backup and Recovery role he must report to his co-admins”

\[
\text{assign} = \langle \text{au}, u, \text{Backup and Recovery}, \\
\text{Report}(\text{au}, u \times \text{Backup and Recovery}, \\
\forall au': au' \in (SA - au)) \rangle
\]

In the above specification, the parameter \( au \) denotes an administrative user who execute the assign right; \( u \) denotes the application user who gets the Backup and Recovery role. The Report obligation sends the message that “the admin \( au \) has successfully modified the user permission relation \( u \times \text{Backup and Recovery} \)” to all his co-admins. Set of receipts of the report may various depends on the context and sensitivity of the role.

4. DISCUSSIONS AND CONCLUSION
Administrative rights are more powerful permissions and checking accountability of execution of admin rights is an important security measure. Most of the administrative RBAC models distribute rights to multiple administrators. Though such decentralized security management has difficulties in checking admin accountability, it is more efficient compared to centralized approach, particularly in large organizations. We introduced administrative obligations in ARBAC as a way to improve the accountability of admin users in the decentralized systems. The proposed approach would reduce the potential of security risk and improve accountability of security administrators. As the cloud and mobile applications are becoming integral part of business information systems, ensuring the accountability of admins play a vital role in system security. Obligations are well studied feature in the security literature and adding them into security administration would open up many possibilities for future developments in this direction.

ACKNOWLEDGMENTS
This research is partially supported by NSF Grants CNS-1111925 and CNS-1423481.

5. REFERENCES