Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments

Roshan K. Thomas
Odyssey Research Associates
Cornell Business and Technology Park
33 Thornwood Drive
Ithaca, NY 14850-1350
(607)257-1975
rthomas@oracorp.com

ABSTRACT

In this paper, we introduce the notion of **T**ea**M**-based Access **C**ontrol (TMAC) as an approach to applying role-based access control in collaborative environments. Our focus is on collaborative activity that is best accomplished through organized teams. Thus, central to the TMAC approach is the notion of a "team" as an abstraction that encapsulates a collection of users in specific roles with the objective of accomplishing a specific task or goal. We were led to the idea of TMAC when our investigations revealed two interesting requirements for certain collaborative environments. The first was the need for a hybrid access control model that incorporated the advantages of broad, role-based permissions across object types, yet required fine-grained, identity-based control on individual users in certain roles and to individual object instances. The second was a need to distinguish the passive concept of permission assignment from the active concept of context-based permission activation. It remains to be seen whether these requirements should lead to yet another variation of one or more models of RBAC, or whether such requirements and TMAC concepts should form another access control model layered on top of RBAC. It is hoped the RBAC workshop will help researchers advance discussions on this issue.

1. Introduction

The notion of role-based access controls (RBAC) is increasingly being incorporated into the security features found in many operating systems and database management systems [1,2]. With RBAC, permissions are partitioned and specified by roles rather than individual users. There are many advantages to using RBAC. First, the notion of roles is an enterprise or organizational concept. As such, RBAC allows us to model security from an enterprise perspective since we can align security modeling to the roles and responsibilities in the enterprise. Second, RBAC is more scaleable than user-based security specifications since security can be administered as a whole for all users belonging to a role. This reduces the cost and administrative overhead associated with fine-grained security administration at the level of individual users, objects, and permissions.

In this paper, we introduce the notion of **T**ea**M**-based Access **C**ontrol (TMAC) as an approach to applying role-based access control in collaborative environments such as those involving workflows [11, 12]. Our focus is on collaborative activity that is best accomplished through organized teams. Thus, central to the TMAC approach is the notion of a "team" as an abstraction that encapsulates a collection of users in specific roles with the objective of accomplishing a specific task or goal.

We were led to the formulation of TMAC during the course of our investigations on a recent DARPA funded research project [14]. The focus was on security issues for clinical workflows associated with patient care. Our goal was to come up with a security paradigm that recognized collaborations in clinical workflows in order to meet three objectives. The first was to provide a security environment that was nonintrusive to clinical staff. The second objective was to provide very tight, just-in-time permissions so that only the appropriate clinical staff could get access to a patient's records and *only* when they were providing care for the patient. The third objective was to design a security framework that did not add any significant administrative overhead and was therefore self-administering to a great extent.

The clinical setting is generally characterized by users with a diverse set of qualifications and responsibilities that can naturally be mapped to various roles. As such, it appeared

RBAC 97 Fairfax Va USA
Copyright 1997 ACM 0-89791-985-8/97/11..$3.50

that RBAC was a good candidate to provide access control. However, closer examination revealed that although RBAC was a good start, additional notions were necessary to effectively apply RBAC in a collaborative setting. Our first observation was the need for a hybrid access control model that incorporated the advantages of having broad, role-based permissions across object types, yet required fine-grained control on individual users in certain roles and on individual object instances. A second requirement was the need to recognize the context associated with collaborative tasks and the ability to apply this context to decisions regarding permission activation. This can be better understood by drawing a distinction between active and passive security models. We consider a *passive security model* to be one that primarily serves the function of maintaining permission assignments, such as in RBAC where permissions are assigned to roles. An *active security model*, on the other hand, distinguishes task- and context-based permission activations from permission assignment.[1] In particular, after a permission is assigned, it may be turned on (activated) and off (deactivated) several times in accordance with the evolving context associated with progressing tasks. Only when a permission is activated will the corresponding operation succeed (i.e., only when a read permission is activated will the user be able to read a document). In a passive model, once a permission is assigned to a user, it is always assumed to be activated independent of any other considerations such as context. For example, this is typically the case with a permission on an access control list (ACL).

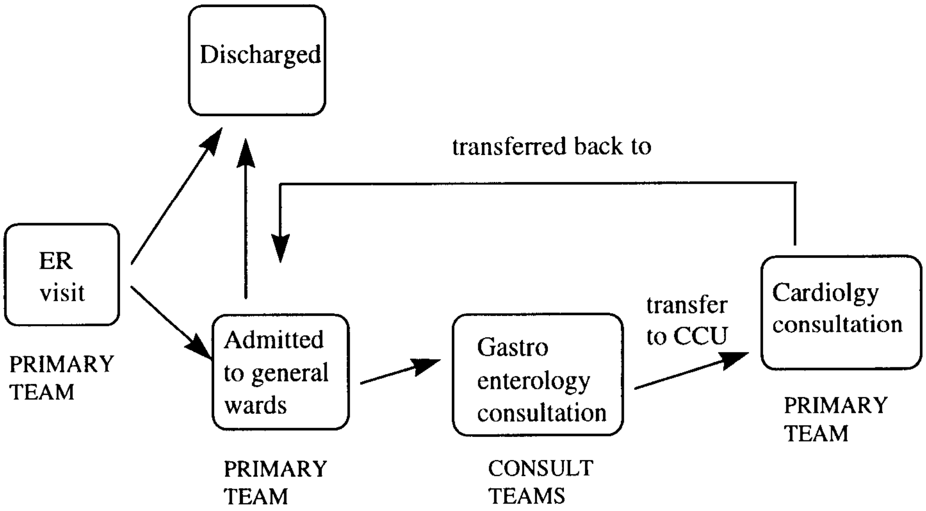

Figure 1. An inpatient scenario

The RBAC96 family of models introduced in [1] supports the notion of role activations within sessions. This clearly allows RBAC96 to distinguish permission activation from permission assignment. Hence one may consider RBAC96 as an active security model. However, the notion of sessions in RBAC96 does not go far enough in encompassing the overall context associated with any collaborative activity.

It is important to point out that our goal in this paper is not to present a comprehensive or formal model of TMAC. Doing so would be premature. Rather, our intention is to present some preliminary ideas at the workshop to gain vital feedback that could lead to further advances. It remains to be seen whether these requirements should lead to yet another variation of one or more models of RBAC, or whether such requirements and TMAC concepts should form another access control model layered on top of RBAC. It is hoped that the RBAC workshop will help researchers advance discussions on this issue. If appropriate, we will publish a more formal model of TMAC after the conclusion of the workshop.

The rest of this paper is organized as follows. Section 2 discusses how RBAC interacts and meets access control requirements in collaborative environments calling for active security. Section 3 discusses TMAC in detail and Section 4 concludes the paper. Section 5 contains various references to the literature.

[1] A similar distinction exists in the field of database management where an *active* database management system is one capable of recognizing and reacting to external conditions and events through mechanisms such as event monitors, evaluation rules, and triggers. On the other hand, a passive database is merely an efficient data storage and retrieval engine.

2. RBAC and Active Security for Collaborative Environments

In this section, we motivate the need for active security models. We begin by illustrating with a clinical workflow an example of how RBAC interacts with the requirements for access control in collaborative applications.

2.1 An illustrative Scenario

The workflow scenario begins with a visit to the emergency room (ER) by a patient who is suffering from pneumonia (see Figure 1). Upon arrival, the patient is quickly screened by a triage nurse who determines that the patient needs to be admitted to the general medicine ward. The primary team that admitted the patient subsequently requests a gastroenterology consultation because the patient is beginning to develop gastrointestinal bleeding. The gastroenterologist who is consulted decides to investigate further and proceeds to do an upper GI endoscopy procedure. While undergoing this procedure the patient has a heart attack and is immediately transferred to the coronary care unit where a cardiology team takes over the care of the patient, becoming the primary team. Eventually the patient's heart condition is stabilized and the patient is transferred back to the general medicine ward. After spending a few more days in the hospital, the patient recovers, is discharged, and is told to see his/her primary care doctor for follow-up care.

At any given time, there exists a primary team that is a single point of contact for the patient and takes overall responsibility for the patient's care. The primary team may, of course, change during the course of care (such as when the patient was transferred from the general ward to the coronary care unit).

We notice the following about this scenario.

- A number of clinical staff (users) was involved in various roles in providing care at various points in the workflow. These included general physicians, specialists such as cardiologists, residents and interns, nurses, etc.
- The staff was organized into care teams and each team was associated with a single department or unit in the organization.
- Care teams were often dynamically formed. For example, when the gastroenterologist joined the care team as the result of a request for a gastroenterology consultation. This dynamic formation can be distinguished from the case in which a staff member is preassigned by the scheduling department to be on a particular team.

From an access control standpoint, we notice the following requirements:

1. The permissions a clinical staff member has to clinical records (documents) should reflect his/her role in providing care. For example, only the cardiologist may prescribe a certain drug for cardiac-related illness and only physicians, not nurses, may order lab tests.
2. Only members of a patient's team should be able to get access to the patient's records. Thus, although a physician, P, may have the right to order a lab test by virtue of the qualifications and responsibilities that determine his role, P should have the right to do so for Patient A's record *only* when P is part of A's care team.
3. Depending on the workflow, various clinical staff needs access to patient records at different points in the overall workflow. Therefore the primary care team in general wards should be given access to a patient's records only *after* the ER unit has requested transfer of the patient to one of those wards.
4. Requirements 1 and 2 above should hold for any staff member who dynamically joins a team.
5. When a patient is transferred from one unit to another, the members of the primary care team of the second unit should be given access to the records of the patient (according to their roles) and no one else.
6. Certain team members may delegate duties and associated permissions to other team members. Thus a physician may authorize a resident or nurse to order a specific lab test or referral for a specific patient. The resident would, in this situation, would need limited (probably one-time) permissions to complete this order, even though under normal circumstances, he/she would not be able to give the order directly since their role is not endowed with such privileges.
7. Once the patient is discharged, all permissions to the patient's medical records should be deactivated.

Let us now explore how RBAC can be used to handle the above requirements. Requirement 1 clearly calls for RBAC and can indeed be handled by RBAC in a straightforward manner. However, as we will see later, the exact nature of permissions in the RBAC model used has certain ramifications. In other words, are permissions specified on classes (types) of objects, or are they specified on individual objects?[2] We will come back to this question later.

Requirement 2 cannot be met completely by RBAC. If we examine this requirement closely, we see that it has three parts:

2.1 The type of permissions allowed to the team members should be determined by their role.

[2] Note that RBAC96 leaves this issue of object types vs. object instances open and flexible.

2.2 Permissions should be restricted to individual user instances of roles belonging to A's team. Thus in the role of physician, only patient A's physician P, and no other physicians, should be given access to A's record.

2.3 Permissions should be restricted to specific instances of objects (records) that pertain to patient A.

We see that requirement 2.1 can be met by RBAC and benefits from the advantages of RBAC-based approaches. Most significant is the reduced security administration costs in medium to large health care settings with hundreds of users. Requirements 2.2 and 2.3 call for access control to be specified at the level of individual users and object instances; unfortunately they cannot be met *simultaneously* by RBAC. Note that RBAC with roles being assigned permissions to individual object instances can meet requirement 2.1 in isolation. However, specifying permissions at the level of individual objects adds to security administration overhead, confusion, and error--precisely the problem RBAC is trying to avoid.[2]. In summary, requirement 2 calls for a hybrid model of access control that incorporates aspects of RBAC as well as user identity and object-based access control. In other words, we want RBAC-based permission assignment on object types at an enterprise level and yet able to activate and control permissions on individual users and object instances at a later time.

Requirement 3 calls for the collective runtime activation of permissions for a set of users (in various roles) in a coordinated fashion and in a manner that can handle requirements such as 2.2 and 2.3. Permission activation is possible in RBAC96 with sessions. However, a session in RBAC96 is a concept that is bound to a single user and allows the user to activate the permissions of a subset of roles to which he/she belongs. To meet requirement 3, we need an abstraction that encapsulates specific instances of various roles. This is precisely what the notion of a team does in TMAC, and in some ways can be seen as a logical extension of the session concept to multiple users. We will discuss this in more detail in the next section.

Requirement 5 calls for permission activation to progress from one team to another for a specific patient. Thus the progression of permission activations would have to maintain some sort of object context that enables instances of roles to point to the records of a specific patient. As we discussed for requirement 2, this is not very straightforward with RBAC.

Requirement 6 calls for the ability to delegate a permission to a specific instance of a role. This essentially means that a permission needs to be assigned and activated for the role instance. Further, this permission has to be on an object instance (i.e., the patient's record) and not on an object type. It is important to note that this cannot be accomplished in RBAC by redefining a role to have the additional delegated permission. Why? Because doing so would give the additional permission to all instances of the role.

Finally, requirement 7 calls for deactivation of all permissions that various team members may hold on a patient's record. This again would not be possible in typical RBAC models because if we remove a permission from a role in RBAC, all users (instances) in this role will lose this permission. If such a permission is defined at the granularity of an object type (class), then all users in this role will lose this permission to all instances of objects in the system of this object type. Neither of these options would be satisfactory as either one would mean a patient's discharge would have a ripple effect resulting in all physicians being unable to access relevant medical records.

In summary, the above scenario would require role-based permission assignment for users and object types, and team-based activation of permissions for individual users and object instances. In the next section, we will discuss how TMAC can be used to accomplish both of these goals.

Although the example given above involves clinical workflows, we believe the same access control issues arise in other environments and applications involving collaboration among groups of users. Examples include collaborative authoring, computer-aided design (CAD/CAM), etc., where teams exchange document and designs while working towards a goal.

2.2 Active vs. Passive Security

In this subsection we briefly discuss active vs. passive security models. From an access control standpoint, we can consider the majority of well-known models to be passive ones. These include typical subject-object models for access control, which are often implemented using access control lists (ACLs) or access control matrices [7, 8], as well as lattice-based access controls such as the Bell-LaPadula Model [6]. These models do not distinguish between permission assignment and activation, and further, are not capable of representing or considering any levels of context when processing an access operation on an object. In these models, some basic definitions (such as a specification of which subject can access which object) are stored and access control requests are validated against these definitions. However, these definitions represent independent and primitive access control information distanced both from application logic as well as from any emerging context associated with ongoing tasks, work units, and processes.

More recently, some work has surfaced that addresses the need for active security models. These include task-based authorization controls (TBAC) [3, 4, 5], which are used to manage authorizations that encapsulate a group of related permissions as well as a workflow authorization model reported in [9]. These models consider the overall context associated with tasks before activating permissions. We expect active security concepts to be an important area of research and believe they will influence the evolution of RBAC.

3. Team-based Access Control

The workflow example presented earlier surfaced two key requirements for access control in collaborative activities.

> R1. The need for role-based, scaleable permission assignment as in RBAC.
>
> R2. The need for fine-grained, run-time permission activation at the level of individual users and objects.

Our basic premise is that RBAC, as we understand it today, cannot be used to enforce these requirements simultaneously. This is because if (R1) above is enforced, then we lose the flexibility required to meet (R2). Enforcing (R2) in isolation does not require RBAC, but we lose the scaleability and administration benefits of RBAC.

Thus the challenge is to come up with an access control approach where (R1) and (R2) can be met concurrently. To do this, we need the following:

- 1. An abstraction to encompass and model a set of users, and the roles these users belong to.
- 2. Some recorded memory of the overall collaboration context for a set of users.

In RBAC models, the only group of users recognized by the model is the one belonging to the *same* role. This limitation was what initially led us to the notion of a team as one that models a set of users in various roles. Once we stumbled on the idea of a team, we observed that a team in an enterprise implicitly carries with it a collaboration context that contains information about the overall mission and task to be accomplished. From an access control standpoint, the collaboration context of a team should contain two pieces of vital information: (1) *user context*, i.e., the specific users comprising a team at any given moment (2) *object context*, i.e., the set of object instances required by the team to accomplish its task. Hence, if we know the basic structure of a team in terms of its various roles, we can meet requirement R1 above. If we know the collaboration context, we can meet R2.

We can now informally discuss the main ideas for a team-based access control approach. A team consists of the following:

- A team name, *t*.
- A set of team members/users, TU.
- A set of team roles, TR, that restricts the roles which members of the team can belong to.
  $TR \subseteq R$, where R is the total set of roles in the information system.
- A special role called *team head (h)*, where $h \subset TR$.. Only one user can be the team head at any given time.
- A set of object types, OT.
- A set of object instances, O.
- A set of team permissions TP, defined across TR and OT, i.e., $TP \subseteq TR \times OT$
- A collaboration context that consists of the following two components.
  - ♦ A user context (UC), where UC : TR x TU
  - ♦ An object context (OC), where OC : OT x O

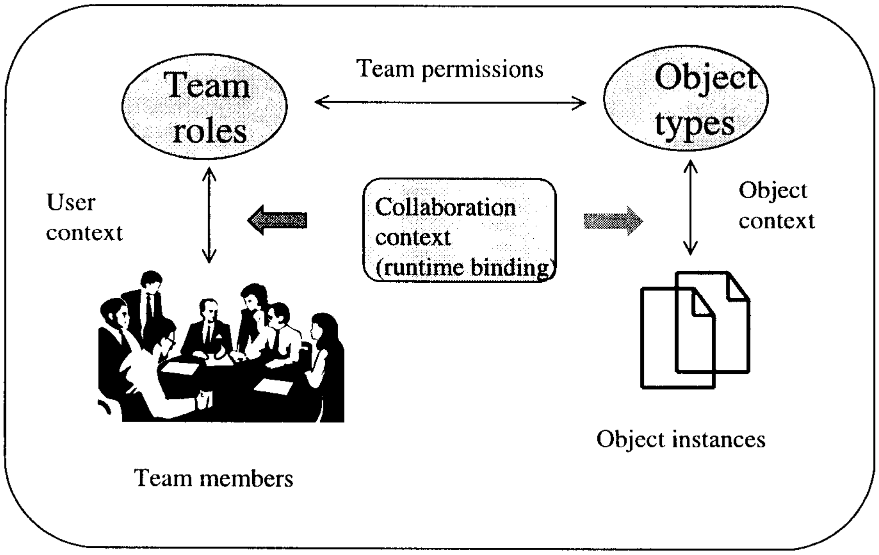

Figure 2. Illustrating concepts in team-based access control

Figure 2 illustrates the main concepts in TMAC graphically.

The basic idea in TMAC is to use RBAC to define a set of permissions P across the domains of R and OT. Individual teams of the same structure (type/class) will encompass the same subset of roles, TR of R, and will thus inherit the same subset TP of P. However, TMAC calls for the run time binding of the TP for each team to the sets TU and O of the team. This allows run-time activation of permissions at the level of individual user and objects.

From an operational and implementation standpoint, TMAC should support the following primitives to enable access control on the team as a whole:

- User_assign (user, team): assigns a user to a team.
- User_deassign (user, team): removes a user from a team.
- Team_activate(team): This binds the team permissions to the team members and the objects they need (TU and O).
- Team_deactivate (team): This deactivates the permissions for the entire team.

In our clinical workflow scenario, these primitives can be used to activate permissions when a patient is transferred to a new team for care, and to deactivate permissions to the patient's record when the patient is discharged from the hospital and exits a workflow instance. In many applications, it may be necessary to activate and deactivate permissions on a user-by-user basis. In this case, similar primitives, User_activate and User_deactivate, need to be supported.

In our initial discussion, we mentioned the need for a security framework that was self-administering, to a great extent, in order to reduce security administration overhead. In other words, as teams are collaborating and as the workflow progresses, permission assignment and deassignment, as well as activation and deactivation, should be achieved without the manual intervention of a human security administrator. A system requiring manual intervention to support active security controls will be error-prone, inefficient, and will be rejected by any enterprise.

TMAC can be made self-administering by trapping basic calls issued by the host information system to assign and deassign team members, as well as by trapping at run-time when workflow instances are invoked. These can then be synchronized with the user assignment and activation primitives to automate security administration. To preserve access control to individual objects across teams, the object context can be passed from one team to another.

4. Conclusions

TMAC allows us to formulate a security model that dovetails the team-based nature of access and work in collaborative settings. TMAC has the advantage of being able to offer the administrative and modeling advantages of RBAC and yet provide fine-grained control over permission activation to individual users and objects. By distinguishing permission assignment from context-based, run-time permission activation, TMAC can be considered an active model of access control. As such, it is able to provide just-in-time permissions and support to a higher degree the principle of least privilege in comparison to passive security models.

It remains to be seen whether the requirements for access control highlighted in this paper should lead to yet another variation of one or more models of RBAC or whether such requirements and TMAC concepts should belong to an access control model layered on top of RBAC. We are hoping the workshop will shed light into this matter and that it will also help us assess the practical significance of the concepts in TMAC.

Acknowledgement

This research was partly funded by the Defense Research Projects Agency Small Business Innovation Research (SBIR) program and under contract # DAAH01-96-C-R093. We are grateful to Teresa Lunt and Gary Koob for their encouragement of this work.

5. References

[1] R.S. Sandhu. Rationale for the RBAC96 Family of Access Control Models. In Proceedings of the First ACM Workshop on Role-based Access Control, Gaithersburg, MD, Nov. 30 - Dec. 1, 1995, ACM .

[2] V. Gligor. Characteristics of Role-based Access Control. In Proceedings of the First ACM Workshop on Role-based Access Control, Gaithersburg, MD, Nov. 30 - Dec. 1, 1995, ACM.

[3] R.K. Thomas and R.S. Sandhu. Towards a Task-based Paradigm for Flexible and Adaptable Access Control in Distributed Applications. In proceedings of the Second New Security Paradigms Workshop, Little Compton, Rhode Island, IEEE Press, 1993.

[4] R.K. Thomas and R.S. Sandhu. Conceptual Foundations for A Model of Task-based Authorizations. In proceedings of the IEEE Computer Security Foundations Workshop, New Hampshire, IEEE Press, 1994.

[5] R.K. Thomas and R.S. Sandhu. Task-based Authorization Controls (TBAC): Models for Active and Enterprise-oriented Authorization Management.

To appear in the 1997 Proceedings of the IFIP WG 11.3 Workshop on Database Security.

[6] D.E. Bell and L.J. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. EDS-TR-75-306, Mitre Corporation, Bedford, MA, March 1976.

[7] M.H. Harrison, W.L. Ruzzo, and J.D. Ullman. Protection in Operating Systems. Communications of the ACM, 19(8), pages 461-471, 1976.

[8] R.S. Sandhu. The Typed Access Control Model. In proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1992, pages 122-136.

[9] V. Atluri and W. Huang. An Authorization Model for Workflows. In Proceedings of the Fourth European Symposium on Research in Computer Security, Rome, Italy, September pages 25-27, 1996.

[10] R.S. Sandhu. Transaction Control Expressions for Separation of Duties In proceedings of the Fourth Computer Security Applications Conference, pages 282-286, 1988.

[11] M. Rusinkiewicz and A. Sheth. Specification and Execution of Transactional Workflows, In Modern Database Systems: The Object Model, Interoperability, and Beyond. W. Kim, Ed., Addison-Wesley / ACM Press, 1994.

[12] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, Distributed, and Parallel Databases. Vol. 3, pages 119-153, 1995.

[13] M. Abrams, K. Eggers, L. LaPadula, and I. Olson. A Generalized Framework for Access Control: An Informal Description. In proceedings of the 13th NIST-NCSC National Computer Security framework, 1990, pages 135-143.

[14] R. Thomas, Security in Workflow Processes: Dynamic Security Models for Clinical Workflows, Odyssey Research Associates Technical report, TM 97-004.