Flexible Team-Based Access Control Using Contexts

Christos K. Georgiadis
Informatics Lab
Faculty of Technology
Aristotle University of Thessaloniki
54006, Greece
chrisgr@eng.auth.gr

Ioannis Mavridis
Informatics Lab
Faculty of Technology
Aristotle University of Thessaloniki
54006, Greece
imav@eng.auth.gr

George Pangalos
Informatics Lab
Faculty of Technology
Aristotle University of Thessaloniki
54006, Greece
gip@eng.auth.gr

Roshan K. Thomas
NAI Labs
Network Associates
8000 Westpark Drive
Suite 600
McLean, VA 22102-3015
rthomas@nai.com

ABSTRACT

We discuss the integration of contextual information with team-based access control. The TMAC model was formulated by Thomas in [1] to provide access control for collaborative activity best accomplished by teams of users. In TMAC, access control revolves around teams, where a "team" is an abstraction that encapsulates a collection of users in specific roles and collaborating with the objective of accomplishing a specific task or goal. Users who belong to a team are given access to resources used by a team. However, the effective permissions of a user are always derived from permission types defined for roles that the user belongs to. TMAC is an example of what we call "active security models". These models are aware of the context associated with an ongoing activity in providing access control and thus distinguish the passive concept of permission assignment from the active concept of context-based permission activation. The ability to integrate contextual information allows models such as TMAC to be flexible and express a variety of access policies that can provide tight and just-in-time permission activation.

Keywords

Teams, access control, contexts, active security.

1. INTRODUCTION

In the past decade, we have witnessed many new emerging trends in computing. These include massive large-scale distribution, automated coordination of tasks using workflow technology and collaborative computing. Accordingly, new models of access control are required to meet the challenges of these new models of computing. Traditional subject-object access control models such as those based on subject-object models typically implemented through access control matrices and access control lists are not capable of embedding the required application-level context information. As such, these models cannot express application-level access control policies in these new application domains.

In this paper, we discuss the integration of team-based access control (TMAC) with contextual information. The TMAC model was formulated by Thomas in [1] to provide a natural way to model access control for collaborative activity best accomplished by teams of users. Thus, central to the TMAC approach is the notion of a "team" as an abstraction that encapsulates a collection of users in specific roles and collaborating with the objective of accomplishing a specific task or goal. TMAC can be distinguished by two key characteristics. First, it is an example of a new breed of access control models called "active security models". Active security models are aware of the context associated with an ongoing activity in providing access control and thus distinguish the passive concept of permission assignment from the active concept of context-based permission activation. Second, when compared to the development of role-based access control (RBAC) models, TMAC is a hybrid access control model. It incorporates the advantages of broad, role-based permission assignment and administration across object types as in RBAC and yet provides the flexibility for fine-grained activation of permissions for individual users on individual object instances.

In TMAC, users are assigned to teams and by virtue of team membership, get access to a team's resources. However, for each user, the exact permissions he/she obtains to a team's resources will be determined by his/her role and the current activity of the team. For example, in a healthcare setting a doctor may have the permission to prescribe certain medications. However, the doctor should not be allowed to prescribe for anyone. Rather, he/she should be allowed to prescribe only for the patient's he/she is taking care of. TMAC can model the above requirement as it sees a doctor as belonging to one or more care teams, where a care team is concerned with the care of a patient. When a doctor joins a patient's care team, he/she will be given access to the patient's medical records. The specific level of access and permissions the doctor can have to these records will be determined by his role in the organization. Thus only a specialist doctor may be allowed to see a section of the records that pertain to the results of very sensitive medical test.

The use of contexts allows team-based access control to be tailored to specify very fine-grained, flexible and context-based access control policies. Thus one can consider such contextual information as the time and shift of a nurse, the location of the patient etc. in modeling access control policies.

Active security models provide very tight, just-in-time permissions so that only the appropriate team members get access to specific records and only when they provide their services, without adding any significant administrative overhead. These permissions are neither granted "too early" nor revoked "too late",

SACMAT'01, May 3-4, 2001, Chantilly, Vriginia, USA.

ensuring in this way a tight matching of permissions to actual usage and need. In other words, the granting, tracking and revoking of permissions are automated and synchronized with the progression of associated tasks. In our healthcare example, as soon as a patient checks out of the clinic or hospital, access to the patient's medical records may be turned off for all or specific members of the patient's care team.

One of the main advantages of TMAC over other access control models such as RBAC is that it is able to leverage the scaleable security administration benefits of role-based permission assignment and yet able to provide fine-grained permission activation and deactivation to individual users and object instances. We can thus assign and administer broad permissions for doctors on object types based on some role definitions and yet activate a doctor's permission to a patient's records (object instances) only when he/she is taking care of the patient.

In this paper, we extend the original TMAC proposal [1] in two key directions. First, we give a framework to integrate TMAC concepts with RBAC. Second, we extend TMAC to use general contextual information. Such contextual information can among others include the time of access, the location from which access is requested, the location where the object to be accessed resides, transaction-specific values that dictate special access policies etc. This allows TMAC to model a richer set of access policies that are more closely tied to application semantics and needs.

2. BACKGROUND

2.1 Role-based Access Control

With role-based access controls, access rights are grouped by role name. This approach offers significant advantages because of scalability. Each user is assigned one or more roles, and each role is assigned one or more permissions that can be given to users in that role [2].

Users are granted membership into roles based on their competencies, credentials and responsibilities in the organization. User membership in roles can be revoked easily and new memberships established as needed. This simplifies the administration and management of permissions since roles can be updated without updating the permissions for every user on an individual basis [3]. Moreover, the use of role hierarchies provides additional advantages since one role may implicitly include the operations that are associated with another role. A recent well-known role-based approach is RBAC [4], which has received considerable attention as a promising way to enhance traditional mandatory and discretionary models.

2.2 Team–based Access Control

The TMAC model was originally proposed by Thomas in [1]. TMAC recognized the importance of context information associated with collaborative tasks and the ability to apply this context to decisions regarding permission activation. The collaboration context of a team contains two pieces: the user context, which could be the current members (users) of a team, and the object context, which could be the set of object instances required by the team to accomplish its task. TMAC allows us to create a general structure (class/definition) of a team with role-based permission assignments to object-types. However, when a team is instantiated, the user context can be used to tailor the role-based permissions defined on object types to user-specific permissions on individual object instances considered to be part of a team's resources.

By aligning access control to the metaphor of teams, TMAC can provide a paradigm for access control that is natural and non-intrusive to the way users work in collaborative environments.

We extend the original TMAC proposal [1] in two key directions. First, we give a framework to integrate TMAC concepts with RBAC. Second, we extend TMAC to use other contextual information besides what is currently used in the user context and object context. This generalized model is referred to in the rest of the paper as C-TMAC (for context-based TMAC). Such contextual information can among others things include the time of access, the location from which access is requested, the location where the object to be accessed resides, transaction-specific values that dictate special access policies etc. This allows TMAC to model a richer set of access policies that are more closely tied to application needs.

3. TEAM-BASED ACCESS CONTROL USING CONTEXTS (C-TMAC)

3.1 Integrating RBAC, TMAC and Contexts

As noted by other researchers [5, 6, 7, 8] a variety of factors and contextual information (like time and location) have to be in considered when influencing the desirable behavior of an access control system during runtime. There are specific application areas, such as the healthcare ones, where it is difficult to define workflow tasks and their access control requirements in a static way. Workflows often tend to be ad-hoc with several users joining and leaving one or more teams in unpredictable way. What is needed is an active access control system that supports context-based permission activation. Our perspective is that all these factors have to be considered in order to formulate the team context associated with a particular task. Context thus identifies the specific need-to-know requirements of each member of the team.

Our proposed Context-based Team Access Control (C-TMAC) approach is based on the integration of RBAC [4] and the TMAC [1] approaches. C-TMAC consists of five sets of entities called users, roles, permissions, teams and contexts, as well as a collection of sessions, which are shown in the diagram of figure 1.

A user (U) is simply a person. A role (R) is a job function within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role.

Permissions (P) are approvals of a particular mode of access to one or more data objects.

User assignment (URS) and permission assignment (PRS) are both many-to-many relations. A user can be a member of many roles, and a role can be assigned to many users. Similarly, a role may have many permissions and the same permission can be assigned to many roles. These relations are the fundamental concepts in RBAC [4].

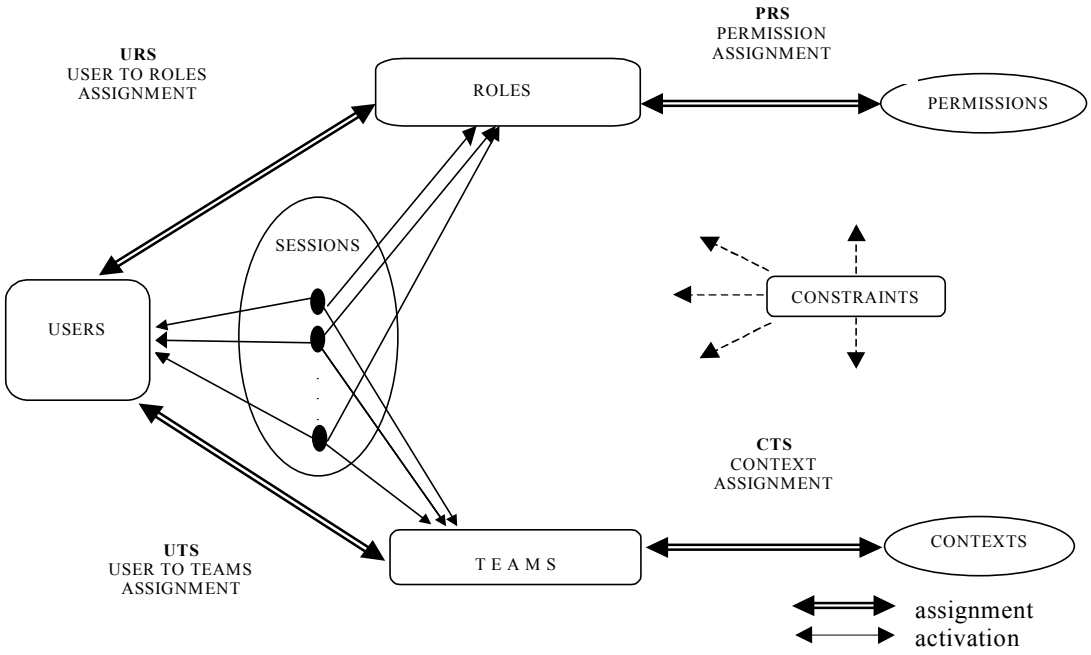

Figure 1 - The C-TMAC approach

An important property of a session (S) is that the user associated with a session, via the session-user function defined below, cannot change. The association remains constant for the life of a session. The permissions available to the user are the union of permissions from all roles activated in that session. In addition, active roles in a session can be changed at the user's discretion.

In context (C) is included information regarding the required data objects for a specific activity, as well as contextual information such as locations and time intervals etc. The team (T) entity is used to represent a group of users having specific roles with the objective of completing a specific activity in a particular context. However, the team concept is used also as a mechanism that associates users with contexts. The use of a team as an intermediary to enable a user to obtain a context is similar to the use of roles as an intermediary between users and permissions. Even when a user is acting alone, we may consider the user as the only member of a private team. During a session, a user can participate in a number of teams. So, each session is also a mapping of one user to a subset of teams that he is a member of. The contexts available to the user are the union of contexts from all teams that he participates in. Moreover, "active" teams in a session can be changed at the user's discretion, just like his active roles. A team can also be seen as a mapping to multiple users. The roles activated by these users identify the permission set available to the team as the combination of permissions from all roles participating in that team.

Context assignment (CTS) and team assignment (UTS) are both many-to-many relations. A team may have many contexts and the same context can be assigned to many teams. Similarly, a user can be a member of many teams, and a team may have many users. However, there are constraints when assigning user to teams. An obvious constraint is related to the roles already assigned to the user. There are mutually exclusive roles and teams, e.g. a user that has been assigned the roles Physician and Director cannot participate in a care-team as a Director.

3.2 Formal Description of C-TMAC

The following definition, which is based on the definition of RBAC0 [4], provides some formalization to the above discussion.

Definition: C-TMAC has the following components:

- U, R, P, S, T, C, stand for users, roles, permissions, sessions, teams and contexts, respectively
- PRS $\subseteq$ P x R, is a many-to-many permission to role assignment relation
- URS $\subseteq$ U x R, is a many-to-many user to role assignment relation
- CTS $\subseteq$ C x T, is a many-to-many context to team assignment relation
- UTS $\subseteq$ U x T, is a many-to-many user to team assignment relation
- session-user : $S \rightarrow U$, is a function mapping each session s_i to the single user user(s_i) that is constant for the session's lifetime
- session-roles : $S \rightarrow 2^R$, is a function mapping each session s_i to a set of roles roles(s_i) $\subseteq \{r \mid (user(s_i), r) \in URS\}$, which can change with time, and session s_i has the permissions $\cup r \in$

roles(s_i) {p | (p, r) ∈ PRS} referred to as Session-Roles Permissions

- session-teams : $S \rightarrow 2^T$, is a function mapping each session s_i to a set of teams teams(s_i) $\subseteq$ {t | (user(s_i), t) ∈ UTS}, which can change with time, and session s_i has the contexts $\cup$ t ∈ teams(s_i) {c | (c, t) ∈ CTS} referred to as Team-Context.
- team-users : $T \rightarrow 2^U$, is a function mapping each team t_i to a set of users users(t_i) $\subseteq$ {u | (u, t_i) ∈ UTS} $\wedge \exists$ s_j : user(s_j) = u}, which can change with time
- team-roles : $T \rightarrow 2^R$, is a function mapping each team t_i to a set of roles roles(t_i) $\subseteq$ {r | (users(t_i), r) ∈ URS}, which can change with time, and team t_i has the permissions $\oplus$ r ∈ roles(t_i) {p | (p, r) ∈ PRS}, referred to as Team-Roles Permissions, and where $\oplus$ means "combination". We may consider different ways in which team-roles permissions could be combined as follows:
  - Aggregation: The set of access permissions of the team is the sum-up (union) of the individual assigned role-based access permissions of all team members.
  - Maximum/Minimum: The set of access permissions is considered to be equal to the maximum or minimum permissions sets of the individual members of the team.
  - Current team structure: The structure (formation) of the team is used to determine the credentials held by the team members. According to a team template, a certain number of members of the team is required. Individual users are not permitted to perform actions on their own but only in the presence of the remainder participants of the team.

3.3 Activation of Final User Permissions

In order to access specific objects using contextual information such as time intervals or trusted locations, we approach access control using role-based permission assignment and team-based permission activation. At first (during the login phase), a user has to complete the identification and authentication procedure, presenting suitable credentials (such as user-id and password information for local networks, or present digital certificates for internet/intranet environments). Then, the user has to select a subset of roles from the set of roles assigned to him/her. According to this selection, a particular set of role-based permissions is granted and these are called session-roles permissions. Note that up to this point, no permissions are available to any specific object instances.

After the role selection, the user has to select a subset of teams to participate. It is worth mentioning that certificate-based credentials, such as attribute certificates [9, 10] could be used for both role and team membership user verifications. After the team selection procedure is completed, the permission set of the user is combined with the permission set available to the team.

As we have mentioned before, teams can be seen as groups of current task contexts. This means that when a user participates in a team he gains also the context of his task. The team context is expressed in terms of ranges of values. For every team, there are a variety of system variables that can hold sets of values for chosen contextual information (factors). The binding of these variables to actual values is accomplished during the runtime by the administration subsystem of the organization.

Team contexts can be seen also, as restrictions on objects and/or on conditions pertaining to the filtering of the access request in such a way as to select appropriate result sets. Thus, the final permission set of a user is filtered using the current context of his team. Any subsequent user access request is allowed only when the necessary role-based permissions have already been granted and only when current values of context variables are contained in the ranges of his team's context.

The activation of user permission is accomplished in accordance with the following two-step procedure:

Step 1: Considering a user who has activated a subset of roles and participates in a subset of teams, initially the role-based permissions of this user are derived with the following definition, where the symbol $\oplus$ stands for "combined with":

- Role-based Permissions =

  Session-Roles Permissions $\oplus$ Team-Roles Permissions

Step 2: The final permissions activated are the context-based permissions, which are derived from role-based permissions (step 1) with the following definition, where $\otimes$ means "filtered by":

- Context-based Permissions =

  Role-based Permissions $\otimes$ Team-Context

The filtering process makes the C-TMAC access control very dynamic. It is simple as a rule in order to determine the final permission set of the user. We may consider it as a mechanism of extracting meaningful subsets of the role-based permission set based on the values of a team's contextual variable such as user location, time, and patient to be billed. This requires that we define valid ranges of acceptable context values for every team.

4. APPLICATIONS OF C-TMAC

We now discuss the applications of C-TMAC with an example from the healthcare domain based on a prototype implementation.

The health care setting is an example where a variety of teams may be involved in a task. Tasks also tend to very dynamic and often ad-hoc. For example, a patient may be transferred from the general wards to the coronary care unit (CCU) as a result of a heart attack and the members of the cardiology team now have to provide care for the next few hours with their specific services. The cardiology team in this example is a group of users with specific roles and contextual variables. In this example, the task has a scope (i.e., taking care of cardiology patients, is executed in a specific location, namely the CCU unit and by specific roles/users, and has a start and a finish point). Therefore, tasks in healthcare environments could be defined at runtime (dynamically) on the basis of the following contextual variables:

- patient: a user (doctor, nurse) gains additional permissions for a specific patient he is in care of.
- location: the collaborative activity depends on the specific area wherein the users (members) of a particular team are working.
- time: all permissions are valid during a certain time (periodic) interval.

Our example uses a backend relational database management system. Thus, the objects of protection are relations, tuples, attributes and views using modes of access operations such as SELECT, INSERT, DELETE and UPDATE.

The permissions of users can be defined as data views (e.g. by using the SELECT statement of the Structured Query Language - SQL). Using views of the relational model results in a view-based protection model [11]. A significant advantage of this definition is the use of flexible granularities to define the objects to be protected [12]. So, it is easy to introduce detailed specifications of specific items (e.g. fields), as well as, more general declarations for coarser groups of data sets (e.g. tables) in order to save storage space.

In the following example, we intend to demonstrate the way C-TMAC active security concepts can be implemented on top of well-accepted passive security models in a collaborative environment. For this reason, we use a relational database access control system that has been defined according to the previously proposed eMEDAC (enhanced Medical Environment Database Access Control) security policy [13] and the corresponding definition methodology of its security mechanisms.

The eMEDAC (enhanced Medical Environment Database Access Control) security policy is based on both mandatory and discretionary security models. It also exploits the RBAC security features that have been tailored to meet the specific needs of a healthcare information system.

Discretionary security is implemented by using user roles that are authorized to execute specific database operations on predefined sets of data on behalf of users that activated them. Those sets of data are defined by using the concept of a view. As a result, instead of authorizing users to have access to the base relations of a database system, user roles are permitted access only to the virtual view relations.

eMEDAC utilizes the notion of a Hyper Node Hierarchy (HNH). This is a mechanism for inheriting permissions (discretional control) and deriving security labels (mandatory control). HNHs are used to construct User Role Hierarchies (URH) and Data Set Hierarchies (DSH) and to derive the security labels (consisting of a security level and a category set) of user roles and data sets. The construction of HNH hierarchies for different administrative domains conforms to a number of constraints. Access control data are centrally defined and stored separately from the application data holders. A flexible number of refinement levels can be specified and these do not have to be strictly equal to the number of mandatory security levels. However, the HNH mechanism specifies (in a mandatory way) a certain number of security levels that cannot be overridden.

In general, the definition of various configurations (role and data-set hierarchies, user to role assignments and session-roles permissions) are accomplished for a particular application during the design phase and is based on static conditions. So, it has to be executed once initially and subsequently requires minimal modification.

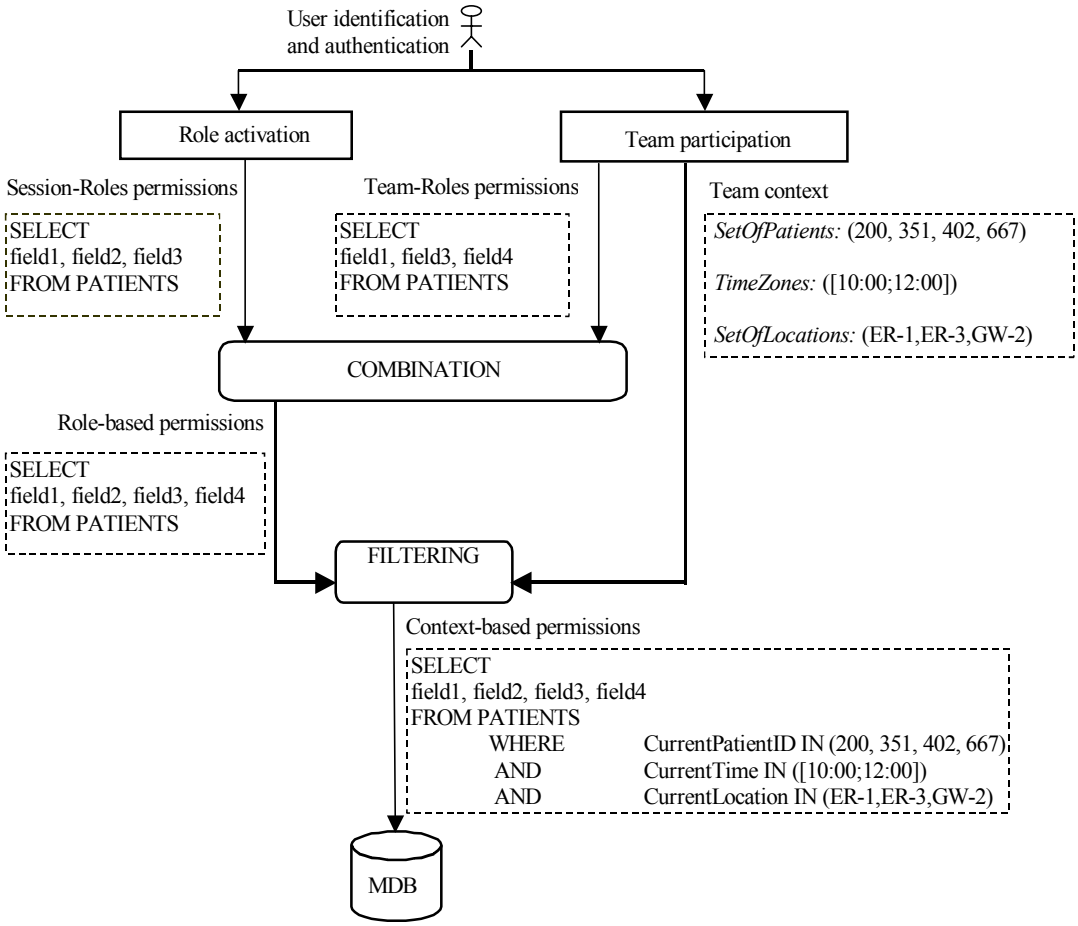

Figure 2 - Context-based permissions activation in healthcare domain.

Our objective is to focus on a simple case in a healthcare environment and define a set of representative factors that constitute the active access control mechanisms. Subsequently, these mechanisms are used to manipulate a given user access request and determine the final access decision. For this purpose, we assume the presence of the following table in the medical database: PATIENTS (PatientID, field1, field2, field3, field4, field5).

For roles Doctor, Head Nurse and Nurse, the sets of role permissions for this table, with view-based protection expressed in SQL form, are as follows:

- Permissions (Doctor): SELECT field1, field2, field3 FROM PATIENTS
- Permissions (Head Nurse): SELECT field1, field3, field4 FROM PATIENTS
- Permissions (Nurse): SELECT field1, field4 FROM PATIENTS

Let assume as context parameters for clinical tasks are the patient-ids, the current time and the current location of the users. This means that the context for every team can be expressed with the corresponding variables SetOfPatients, TimeZones and SetOfLocations, wherein are placed the sets of actual values regarding charged patient-ids, work time zones and responsibility locations, respectively.

In our example, we assume an Emergency Room care-team (ER-Team). Possible locations of users could be ER-1, ER-3 for emergency rooms 1 and 3 respectively, as well as GW-2 for the general ward. Then the context assigned to the ER-Team could be as follows:

Context (ER-Team): SetOfPatients: (200, 351, 402, 667)
TimeZone: [10:00;12:00]
SetOfLocations: (ER-1, ER-3, GW-2)

This context can be expressed as a WHERE clause that is going to be added subsequently to the view definition of the role-based permissions of the user in order to filter them:

WHERE CurrentPatientID IN SetOfPatients
AND CurrentTime IN TimeZone
AND CurrentLocation IN SetOfLocations.

We assume that users Mary and Helen have already started their sessions s1 and s2 and activated their roles HeadNurse and Nurse respectively, and they are participating in the ER-Team.

Session 1:

session-user(s1) = ‘Mary’
session-roles(s1) = [HeadNurse]
session-teams(s1) = [ER-Team]

Session 2:

session-user(s2) = ‘Helen’
session-roles(s2) = [Nurse]
session-teams(s2) = [ER-Team]

Team-users (ER-Team) = [Mary, Helen]
Team-roles (ER-Team) = [Head Nurse, Nurse]

The permissions of the ER-Team are determined as the union of the permissions of the participating roles:

Team-Roles Permissions (ER-Team) =
SELECT field1, field3, field4 FROM PATIENTS

Continuing our example, we assume that user Chris is starting his session s3 and is activating the role Doctor and is participating in the ER-Team.

Session 3:

session-user(s3) = ‘Chris’
session-roles(s3) = [Doctor]
session-teams(s3) = [ER-Team]

Team-users (ER-Team) = [Chris, Mary, Helen],
Team-roles (ER-Team) = [Doctor, Head Nurse, Nurse].

According to the proposed procedure for defining the team-roles permissions, we have chosen to use the aggregation (union) as the combination method for role-based permissions:

Step1: Role-based Permissions(Chris) =

= Session-Roles Permissions (Doctor) $\oplus$ Team-Roles Permissions (ER-Team) =

= {SELECT field1, field2, field3 FROM PATIENTS} $\cup$
$\cup$ {SELECT field1, field3, field4 FROM PATIENTS}
= SELECT field1, field2, field3, field4 FROM PATIENTS

Step2: Context-based Permissions (Chris) =

= Role-based Permissions (Chris) $\otimes$ Team-Context (ER-Team) =
= SELECT field1, field2, field3, field4 FROM PATIENTS
WHERE CurrentPatientID IN (200, 351, 402, 667)
AND CurrentTime IN [10:00;12:00]
AND CurrentLocation IN (ER-1,ER-3,GW-2)
(view-1)

CurrentPatientID, CurrentTime and CurrentLocation are variables that are bound to concrete values during the access request. In order to make this clear, we assume Chris places the following access request:

SELECT field1, field4 FROM PATIENTS
WHERE PatientID = CurrrentPatientID
(view-2)

at 11.30 (CurrentTime = [11:30], from Emergency Room Nr.1 (CurrentLocation = ER-1) and for patient with ID = 351 (CurrentPatientID = ‘351’).

This action has two effects. Firstly, view-1 expression is evaluated, since all variables contain specific values. Secondly, based on the user access request, the system evaluates and compares view-1 and view-2. The access control system examines whether the set of results from access request (view-2) is included (IN clause of SQL) in the set of results of context-based permissions (view-1).

In summary, Chris indeed gets access and his request (view-2) is satisfied, as all relative parameters of his specific access request belong to the contextual information of ER-Team. If at least one of the parameters CurrentTime, CurrentLocation, CurrentPatientID, has a value that is outside the range of ER-Team context (e.g. Chris requested access from Emergency Room Nr. 2), then the request would have been denied.

In summary, we used a relational database management system to model all role-based permissions of a healthcare organization as views (SELECT statements) according to the eMEDAC security policy. We then exploited the advanced characteristics of Dynamic SQL in modern relational DBMSs. Dynamic SQL statements are stored in character strings that are input to or built by the program at runtime. We store in appropriate data structures the context values of every team. Naturally, these values can be changed at runtime by the administration of the organization. It is important to note that these values are implemented by using Dynamic SQL statements. These contextual values are capable of implementing the filtering process of C-TMAC model. The character strings that contain the role-based permissions are extended in order to attach additional WHERE clauses to the initial SELECT statements. When these extended strings are executed as dynamic SQL statements, the newly added WHERE clauses provide selections of rows of data objects according to the team context.

From our development and implementation experience we are convinced that C-TMAC provides significant capabilities to model and implement permission activation mechanisms in a flexible manner, so as to meet the needs of collaborative environments.

5. CONCLUSION

We have presented an approach to integrating team-based access controls with RBAC and contexts. TMAC preserves the advantages of scaleable security administration that RBAC-style models offer and yet offers the flexibility to activate permissions for individual users and to specific object instances. The C-TMAC model developed in this paper allows the use of general contextual information and gives TMAC the capability to model a rich set of security policies and to tune permission activation and deactivation in very flexible ways. We have also shown in this paper, how C-TMAC concepts can be implemented over passive role-based security policies and mechanisms such as that found in relational database systems and eMEDAC. We believe that TMAC and its variations will prove to be an interesting starting point for further investigations of security models for next-generation collaborative applications.

6. REFERENCES

[1] Thomas R.K. Team-Based Access Control (TMAC): A Primitive for Applying Role-Based Access Controls in Collaborative Environments, Proceedings of the Second ACM workshop on Role-based Access Control, Fairfax, VA USA, 1997.

[2] NIST. Role Based Access Control, National Institute of Standards and Technology, 1999, available in URL: http://hissa.ncsl.nist.gov/rbac

[3] NIST. An Introduction to Role-based Access Control, NIST CSL Bulletin on RBAC, National Institute of Standards and Technology, 1995, available in URL: http://csrc.nist.gov/nistbul/csl95-12.txt

[4] Sandhu R. Role-Based Access Control, Advances in Computers, Vol.46, Academic Press, 1998.

[5] ISO. ISO / IEC 10181-3 Model of Access Control, X/ Open Guide Basic Security Facilities - Authorization in Distributed Security Framework, 1994.

[6] Beznosov K. Requirements for Access Control: US Healthcare domain, Proceedings of the Third ACM Workshop on Role-Based Access Control, October 1998, Fairfax, VA, USA, 1998.

[7] Lupu E. and Sloman M. Reconciling Role Based Management and Role-based Access Control, Proceedings of the Second ACM Workshop on RBAC, Fairfax, VA, USA, 1997.

[8] Giuri L. and Iglio P. Role Templates for Content-Based Access Control, Proceedings of the Second ACM Workshop on RBAC, Fairfax, VA, USA, 1997.

[9] Farrel S. and Housley R. An Internet Attribute Certificate Profile for Authorization, Internet Draft: draft.ietf.pkix.ac509prof-03.txt, work in progress, May 2000.

[10] Mavridis I., Georgiadis C., Pangalos G. and Khair M. Using Digital Certificates for Access Control in Clinical Intranet Applications. Journal Technology and Health Care, Vol. 8, Nos. 3, 4 (2000), ISSN 0928-7329, p. 173-174, IOS Press, 2000.

[11] Pernul G. Database Security, Advances in Computers, Vol.38, M.C. Yovits (Ed.), Academic Press, 1994.

[12] Pangalos G. and Khair M. Design of a Secure Medical Database Systems, in IFIP/SEC'96, 12th international information security conference, 1996.

[13] Mavridis I., Pangalos G. and Khair M. eMEDAC: Role-based Access Control Supporting Discretionary and Mandatory Features, Proceedings of 13th IFIP WG 11.3 Working Conference on Database Security, Seattle, Washington, USA, 1999.