TRDBAC: Temporal Reflective Database Access Control

Zahid Rashid, Abdul Basit and Zahid Anwar
School of Electrical Engineering and Computer Science (SEECS)
National University of Science and Technology
Islamabad, Pakistan
{09msitzrashid, 09msitabasit, zahid.anwar}@seecs.edu.pk

Abstract—**Database access control policies can become extremely complicated and complex in large databases such as hospital medical systems, banks and enterprise resource planning systems of large enterprises etc. The complexity in access control policies may results in security breaches if the policies are ambiguous, not well defined and implemented incorrectly. e.g. HSBC database security breach reported in year 2006 in which an ex-employee swiped away almost 24,000 customers accounts due to incorrect access policies. The access control policies define the rights and privileges of users on database objects. In order to keep these database systems secure, the database security should provide controlled, protected access to the contents of a database as well as preserve the integrity, consistency, and overall quality of the data. In order to implement the consistent database access control policies, a number of models have been developed by the database security community such as, discretionary (DAC) and mandatory (MAC) access control models, role-based access control model (RBAC), reflective database access control (RDBAC). RDBAC is a relatively new and more expressive access control model that provides a more fine-grained level control than the previous models. Move over database privilege is expressed as a database query itself, rather than as a static privilege contained in an access control matrix.**

In this paper, we propose Temporal Reflective Database Access Control (TRDBAC)- a new access control policy designed to address a limitation of RDBAC: the inability to express time-constraints, just as TRBAC extends RBAC to incorporate the notion of time. To show how our new policy works we have demonstrated a case study on students result information system, in which policies are written in a time based extension of reflective database access control (RDBAC) and converted to SQL queries. Finally we analyze the behavior of our new model.

Index Terms*—*Access control policies; Controlled access; Reflective database access control; RDBAC; Temporal Reflective Database Access Control; TRDBAC

I. INTRODUCTION

Database systems are one of the core components in any organization. As the size of the organization grows the complexity and data in the databases also grows rapidly. Besides other database security features, the access control mechanism should be flexible enough to implement different types of access control policies periodically. Furthermore, it is also important to maintain the consistency in the access control mechanism of the database system.

This work was a part of research conducted for ISE854/EC840 at the National University of Sciences and Technology (NUST), Islamabad, Pakistan

While talking about the database access control mechanisms, the database systems faces many threats i.e. internal threats (threats from inside an organization) and external threats (threats from outside the organization). These threats then leads to different security breaches such as unauthorized data observation, incorrect data modification and data unavailability etc. [1]. Many organization of varying domain such as healthcare, banking systems etc. suffers significant loss in terms of both financial and resources as a consequence of unauthorized data observation [1]. The ideal database system should be secure enough so that it can protect the important data of an organization from any type of threats. The data in the database systems should be exposed to the users only who need that information and their nature of work requires that data. Consider the example of student result information system the information of grades should be modifiable by the instructor or by the exam coordinator only. Similarly consider health information system of a hospital. The information of patients should only be viewable to the concerned doctor or physician only, and patients may view their own data, if it has been released for view.

In database systems, the access control mechanism enables controlled access to subjects on objects. The main objective of the access control mechanism ensures confidentiality of data. The rights of the subjects are defined and checked against the set of authorization by the access control mechanisms. An authorization states whether a subject can perform a particular action on an object. Authorizations are stated according to the access control policies of the organization [1]. Until now the research community has described many access models for relational database system. Some of these are

- *Discretionary access control (DAC)*
- *Mandatory access control (MAC)*
- *Role-based access control model (RBAC)*
- *Temporal role-based access control (TRBAC)*
- *Reflective database access control (RDBAC)*

The above mentioned options are available to implement the access control policies. The selection of appropriate access model is a critical decision made by the database administrators. The administrator has to keep in view a number of characteristics of the environment as well as the sensitivity of the data in the databases.

In this paper, we implement the access control policies using RDBAC for student results information system. Transaction Datalog (TD) [13] is used to write the access control policies. Furthermore, these policies are implemented in the SQL. We observed that RDBAC does not consider the time constraint during expressing the access control policies e.g. in many organizations, functions may have limited or periodic temporal duration. Consider, for instance, the instructor can also act as an exam coordinator in case of unavailability of designated exam coordinator i.e. the instructor is acting as instructor (primary function) as well as exam coordinator (temporal based function) for a specific duration. Consider another example of medical database systems in hospitals; the doctor-on-night-duty role is enabled during the night. Since doctors may need the assistance of a nurse, the access policy should make sure that the corresponding function i.e. nurse-on-night-duty is enabled whenever doctor-on-night-duty is [12].

In order to cope with these types of policies we have to consider the time constraint. As RDBAC does not explicitly considers the time constraint therefore, we extended the RDBAC and presented the idea of Temporal Reflective Database Access Control (TRDBAC). This model will allow the policy makers and database administrators to design and implement time based database access control policies.

The remainder of the paper is organized as follows: In Section 2, we have discussed the characteristics of DAC, MAC, RBAC, TRBAC and RDBAC. In section 3, we discussed how the access control policies be implemented using the reflective access control model for the student results information system. Section 4 describes the implementation of access control policies for student results information system using the TRDBAC. Finally section 5 presents some concluding remarks.

II. ACCESS CONTROL MODELS

A. Discretionary Access Control (DAC)

In DAC, restrictions on access to database system is based on the identity of subjects and/or groups to which they belong to and authorization rules. DAC is discretionary in the sense that it allows subjects to grant database access control to other subjects. Due to the flexibility of granting permission, discretionary access control model is adopted by many commercial DBMSs i.e. Oracle9i, MySQL etc. The most common implementation of DAC is through ACL's which are dictated and set by the owners. The DAC has the flexibility of granting access but this benefit results in the drawback of illegal access to confidential information due to inappropriate passing of access controls.

B. Mandatory Access Control (MAC) [10]

MAC is a mean of restricting access to objects based on sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity. The implementations of MAC in RDBMSs have focused solely on Multilevel Security (MLS). Multilevel Security (MLS) is a capability that allows information with different classifications to be available in an information system, with users having different security clearances and authorizations, while preventing users from accessing information for which they are not cleared or authorized. Due to this restriction MAC has been used in few governmental organizations where the levels of authorization can be identified but in most commercial application this is not easy to identify such type of levels. A few commercial RDBMSs offering MAC based access control e.g. Trusted Oracle, Informix OnLine/Secure, and Sybase Secure SQL Server. Besides the limitations of MAC an important advantage of such label security products is their ability to offer a centrally managed tool for defining label access policies and for assigning access labels to users.

C. Role Based Access Control (RBAC) [11]

In RBAC, policies are described in terms of users, subjects, roles, role hierarchies, operations, relationships, and constraints. The users are granted membership into roles based on their competencies and responsibilities. The users must be active in a role and authorized as a member of the role by a security administrator for performing operations. The main theme of RBAC is that it provides the capability to administrators to place constraints on role authorization, role activation, and operation execution. One of the constraints in RBAC is the non-discretionary nature of access controls that simplifies the understanding and management of access policies. Furthermore, it facilitates the administrator to control the access at a level of abstraction.

D. Temporal Role Based Access Control (TRBAC) [12]

TRBAC is an extension of RBAC which supports temporal constraints on the enabling/disabling of roles. Such constraints are expressed by means of role triggers (rules that are automatically executed when the specified actions occur) can also be used to constrain the set of roles that a particular user can activate at a given time instant. The firing of a trigger may cause a role to be enabled/disabled either immediately, or after an explicit specified amount of time. The more power of expressiveness of TRBAC leads to the drawback of ambiguity in understanding the status of the user's role at any particular time e.g. the user may belong to more than one role instead of basic role because firing of some triggers.

E. Reflective Database Access Control (RDBAC)

RDBAC is newly emerging access control model. RDBAC tries to overcome the difficulties to implementing the complex access control model. As mentioned previously current databases use a conceptually simple model for access control such as access control matrix and definition of roles etc. and based on this the users are allowed to perform operation i.e. read, insert, update, or execute on database resources i.e. tables, views, and functions etc. But these techniques face weakness in granting access control at more fine-grained level i.e. access control at certain portions of database table. If such level of access needs to be granted than a separate view is to be created, which contains required data and then the user is granted access on that view. These models provide a handsome flexibility in some cases such as users are allowed to define access privileges for their own tables, without requiring super user privileges [7]. However, these models are limited to

expressing the extent of the policy, such as "Alice can view data for Alice," "Bob can view data for Bob," etc., rather than the intent of the policy, such as "each employee can view their own data." But the policy administration becomes more tedious in changing nature of databases where data is changing rapidly, modification in database schema and new users may came at a faster pace.

In other types of access controls where the policy implementation is achieved by defining roles in addition to ACMs to group together common sets of privileges, the same problem of fine-grained policies does not fully addressed. For example in a scenario such as the policy of "each employee can view their own data in the table," each user requires an individually-defined view of a table as well as a separate role to access each view, which yields no benefit over a standard ACM-based policy[5][7].

In order to overcome these problems and for the administration of address control policies in an effective manner RDBAC is proposed. RDBAC is a model in which a database privilege is expressed as a database query itself, rather than as a static privilege contained in an access control matrix [7]. In this model access control policy decisions can depend on data contained in other parts of the database, such as attributes of the user, attributes of the data being queried, or relationships between the user and the data [5]. In this model efforts are made to remove the above mentioned restrictions and allow policies to refer to any part of the database. One of the main advantages of RDBAC is that it aids in improving the expressiveness of access control policies. In [5] a policy is presented, "employees who are registered as managers may view contact data for the employees they manage" which illustrates the benefits of RDBAC i.e. suppose we have a database that contains a table listing a company's employees, along with their position in the company and the department in which they work. Suppose we want to grant all employees having manager role to access the data of the other employees in their department. When a manager queries this table, the policy will first check that the user is indeed a manager, then retrieve the manager's department, and finally return all employees in that department. This approach has at least two benefits. First, the policy leverages data already being stored in the database. Second, the policy describes its intent rather than its extent; thus, privileges are automatically updated when the database is updated (for instance, when an employee receives a promotion to manager), preventing update anomalies that leave the database in an inconsistent state.

One of the lacking of RDBAC is that, there is no efficient way for formal mathematical molding of access control policies for practical database systems. For this purpose transaction Datalog [8] language was used which provides a powerful syntax and semantics for expressing RDBAC policies but it is not implemented and tested for large scale implementation and also lacks syntax and semantics that correspond to certain commonly-used operations in SQL. However a strategy was demonstrated in [7] for compiling policies in Transaction Datalog into standard SQL views that enforce the policies, including overcoming significant differences in semantics between the languages in handling side-effects and evaluation order.

Some products such as Oracle's Virtual Private Database technology (VPD) is a significant implementation of RDBAC, in which queries on a resource are rewritten based on a policy function that can filter the results of the query. The policy function may in turn contain other queries, and these queries may in turn be rewritten based on other policy functions. One other example is Sybase's Adaptive Server Enterprise database similarly uses query rewriting based on logical conditions, including arbitrary logic written in user-defined Java functions. But these implementations lack the formal mathematical modeling.

III. CASE STUDY: IMPLEMENTATION OF RDBAC FOR STUDENTS RESULTS INFORMATION SYSTEM

In this case study we implemented the privacy policies of students result information system. This system allows the *students* to view their results as well as facilitates the *instructors* to upload the results and also the *Exam Branch coordinators* to manage the results. The *administrators* have access of the whole system for monitoring and management. Let's have some detailed overview of the students result system case study.

The basic purpose of implementing the access control policies is to enforce the privacy policy in the results information system. The following points are necessary to be discussed in order to design the access control policies to prevent unauthorized access. The basic policy is that the students can view the results of other students enrolled with him/her only before the final term exams results. But the students are restricted only to view their own final results instead of viewing the results of other students.

The data of results is stored in *Results* table and the users of the system like *Instructors*, *Students* etc. can access the data according to the permissions granted to the user type. The instructor can only upload the results of the students that are enrolled with the course taught by the same instructor. Once the results are uploaded and finalized by the instructor, the instructor will not be able to change the result of any student. If some changes are required, a request will be sent to the *Exams Coordinator* for any type of alteration.

Final result showing policy also allows the parents/guardian of the students to view the final results and the grading. The mechanism to implement this policy will ensure that parents/guardian will see the results of their own children.

The *Exam Branch Coordinator* has the capability of viewing/altering final results of all the students. For alteration in the student's results, it is required to provide the reason of change in the record. These actions are recorded and will be audited later in order to prevent unauthorized data alteration to maintain the data integrity.

The *Administrator* is the only authority capable of changing the status (Active/Inactive) of other users including *student*, *instructor*, *exam coordinator* etc. The administrator can also view the system log. The following policies are defined keeping in view the above mentioned rules:

A. Policies

1) General Purpose

- User can only login to the system if his/her status is "Active".
- Users can view their own personal information only.
- Users can only update their own personal and contact information.
- Deletion of any record from the database will not actually delete the record physically from database rather its status would be marked as Inactive.

2) Policies for Instructor

- The instructor can only upload the result information of the students that are enrolled with his/her course.
- The instructor can only view the results data and the contact information of the students that are enrolled with the course taught by the same instructor.
- An instructor can only update, delete the result data of the students registered with his/her own course before the results are marked as finalized by the instructor.
- Once the students' results are finalized by the instructor, the instructor will not be able to change the results later.
- The instructor can also change the status of the student's results information like "Result Late" to "Announced" for the students enrolled with his/her course only.
- Only the instructor and the Administrator can view the contact information of the exam coordinator.
- The instructor can update his/her own contact information
- The instructor can only send the result information of a student via email on the student request, provided the student email is there and it is correct.

3) Policies for Students

- A student can only view the results of his/her own subject.
- Student can view the contact information of the instructor of the course with which he/she is enrolled.
- Student would be capable of updating his/her own contact information
- Student can only contact to the instructors of the subjects they have taken regarding any results related issues.
- Students can view the detailed results of other students before the final term exams. These details include the marks obtained in midterm, assignments, quizzes, projects etc.

4) Policies for Exam Coordinator

- Exam coordinator can only view the results of all the students that are currently enrolled with any subject.
- Exam coordinator can view the contact information of the instructor associated with a course for any type of grading issue.
- The exam coordinator can contact to the administrator in case of any issue in the system.
- Only the exam coordinator can change the student results after it is marked as finalized by the instructor, keeping in mind the actions are recorded and will be audited later.

5) Policies for Parent/Guardian

- Parents/Guardians can only view the final result and grades of their own children, published at the SEECS website.
- Parents/Guardians can only view the result information of their own child, published at the website.
- Parent/Guardian can also view the contact information of the instructors having the courses with which their children are enrolled.

6) Policies for Administrator

- The administrator can view all the data from any table
- The administrator and the instructor can only view the contact information of the exam coordinator.
- Only the administrator can change the status of the users from Active to Inactive and vice versa.
- Only the administrator can view the change logs generated during various actions performed in the system.
- Only the administrator can add a new user in the system.
- The administrator can't change the contact information of any other user in the system.
- The administrator can only view the students' results but can't change the students' results.

B. Policies Implementation in RDBAC

1) Basic Status Policy (Active)

We have to first define that the user of the system is Active then the user will be able to access the information.

a) Instructor is Active

Instructor (Id, Name, Contact, .. , .. , .. , IsActive) e.g.
Active instructor has record like (5, ABC, 234, .. , .. , .. , 1)

b) Student is Active

Student (Id, FName, LName, .. , .. , IsActive) e.g.
Active student has record like (3, AB, CD, .. , .. , 1)

c) Exam coordinator is Active

ExamCo(Id, Name, .. , .. , IsActive) e.g.
Active exam coordinator has record like (2, XYZ, .. , .. , 1)

2) Basic Access Policy (HasAccess)

Now while implementing the access policies that which user of the system can access what system resources.

Instructor wants to see the students' information:

hasAccess(IID, SID):-
Instructor(IID, Name, , , 1)
Student(SID, FName, LName, , , 1)
Courses(CID, Code, Title, , , IID, 1)
StudentCourses(SID, CID, , ,)

Exam Coordinator wants to see the student information:

hasAccess(ECID, SID):-
ExamCo(ECID, Name, , , 1)
Student(SID, FName, LName, , , 1)

3) Contact Policy

As described earlier, a user can view the contact information of other users as according to the policy enforced. Consider the case that the instructor wants to see the contact information of the student, so the policy implementation will be as following:

View.ContactInfo(FName, LName, Email, Contact, , ,):-
Instructor(IID, Name, , , , 1)
Student(SID, FName, LName, Email, Contact, , , 1)
Courses(CID, Code, Title, , , IID, 1)
StudentCourses(SID, CID, , ,)

4) View Result Policy

A student can only view his/her own results; also the instructor can view the results of all the students that are enrolled with the subject taught by the same instructor. The Exam Coordinator can view the results of all the active students enrolled in any subject. Parents/Guardians can also view the result of their own child. So following are some policies implementation described above:

a) Student Results View

```
View.viewResults(FName, LName, CourseName, Marks, Grade, CGPA):-
        Student(SID, FName, LName, 1)
        Courses(CID, Title,1)
        StudentCourses(SID,CID,1)
        Results(SID, CID, Marks, Grade)
```

b) Instructor Results View

```
View.viewResults(FName, LName, CourseName, Marks, Grade, CGPA):-
        Instructor(IID,1)
        Student(SID, FName, LName, 1)
        Courses(CID, Title, IID,1)
        StudentCourses(SID,CID,1)
        Results(SID, CID, Marks, Grade)
```

c) Exam Coordinator Results View

```
View.viewResults(FName, LName, CourseName, Marks, Grade, CGPA):-
        ExamCo(ECID,1)
        Student(SID, FName, LName, 1)
        Courses(CID, Title, 1)
        StudentCourses(SID,CID,1)
        Results(SID, CID, Marks, Grade)
```

d) Parent/Guardians Results View

```
View.viewResults(FName, LName, CourseName, Marks, Grade, CGPA):-
        ExamCo(ECID,1)
        Student(SID, FName, LName, PID, 1)
        Courses(CID, Title, 1)
        StudentCourses(SID,CID,1)
        Parent(PID, 1)
        Results(SID, CID, Marks, Grade)
```

e) Administrator Results View

```
View.viewResults(FName, LName, CourseName, Marks, Grade, CGPA):-
        Admin(AID,1)
        Student(SID, FName, LName, 1)
        Courses(CID, Title, 1)
        StudentCourses(SID,CID,1)
        Results(SID, CID, Marks, Grade)
```

5) Insert Result Policy

Only the instructor of the course is allowed to insert the result of the students that are enrolled with his/her course. But once the instructor has submitted the results to the examination branch then the instructor can't insert or update the results. In this case the instructor will request to the Exam Coordinator to add/update the result of any student. Following are the policies implementation of insertion of the results:

a) Insert Results View by Instructor

```
View.Ins.sResults(CID, SID, Marks,Grade,1)
        Results.IsInActive(SID, CID) || Result.NotExist(SID, CID)
        Instructor(IID, 1)
        Student(SID, 1)
        Courses(CID, IID, 1)
        StudentCourses(SID, CID, 1)
        Ins.sResults(SID, CID, Marks, Grade,1)
```

b) Insert Results View by Exam Coordinator

```
View.Ins.sResults(CID, SID, Marks,Grade,1)
        Results.IsInActive(SID, CID) ||
        Result.NotExist(SID, CID)
        ExamCo(ECID, 1)
        Student(SID, 1)
        Courses(CID, 1)
        StudentCourses(SID, CID, 1)
        Ins.sResults(SID, CID, Marks, Grade,1)
```

6) Update Result Policy

Only the instructor of the course is allowed to update the result of the students that are enrolled with his/her course and additionally the Exam Coordinator can also update the result of any student upon the request from the instructor after the result was submitted finally.

a) Update Results View by Instructor

```
View.Upd.sResults(CID, SID, RID, Marks, Grade, 1)
        Results.IsActive(SID, CID,RID,1) || Result.Exists(SID, CID, RID)
        Instructor(IID, 1)
        Student(SID, 1)
        Courses(CID, IID, 1)
        StudentCourses(SID, CID, 1)
        Results(RID, CID, SID, 1)
        Upd.sResults(SID, CID, RID, Marks, Grade, 1)
```

b) Update Results View by Exam Coordinator

```
View.Upd.sResults(CID, SID, RID, Marks, Grade, 1)
        Results.IsActive(SID, CID,RID,1) || Result.Exists(SID, CID, RID, 1)
        ExamCo(ECID, 1)
        Student(SID, 1)
        Courses(CID, 1)
        StudentCourses(SID, CID, 1)
        Results(RID, CID, SID, 1)
        Upd.sResults(SID, CID, RID, Marks, Grade, 1)
```

7) Delete Result Policy

If the results are not finally submitted, then only the instructor of the course is allowed to delete the result of the students that are enrolled with his/her course. If the results are submitted in Examination Branch then only the Exam Coordinator can delete the student result on the request of instructor.

a) Delete Results View by Instructor

```
View.Del.sResults(CID, SID, RID, Marks, Grade, 1)
        Results.IsActive(SID, CID,RID,1) || Result.Exists(SID, CID, RID,1)
        Instructor(IID, 1)
        Student(SID, 1)
        Courses(CID, IID, 1)
        StudentCourses(SID, CID, 1)
        Results(RID, CID, SID, 1)
        Del.sResults(SID, CID, RID, 1)
```

b) Delete Results View by Exam Coordinator

```
View.Del.sResults(CID, SID, RID, Marks, Grade, 1)
        Results.IsActive(SID, CID,RID,1) || Result.Exists(SID, CID, RID,1)
        ExamCo(ECID, 1)
        Student(SID, 1)
        Courses(CID, 1)
        StudentCourses(SID, CID, 1)
        Results(RID, CID, SID, 1)
        Del.sResults(SID, CID, RID, 1)
```

C. Implementation of Policies in SQL

Policies that are defined using the Transaction datalog can be expressed in standard query language (SQL) that is understandable for the DBMSs. Here we will transform some of the RDBAC policies to the SQL for better understanding the actual implementation of the RDBAC at database level, assuming that the user executing these is Active.

a) Instructor wants to see the students' contact information

```
Create View View_Students_Contact AS (SELECT s.FName, s.LName,
s.Address, s.Cell FROM Students s INNER JOIN StudentCourses sc ON s.SID
= sc.SID INNER JOIN Courses c ON sc.CID = c.CID WHERE c.InstName =
CURRENT_USER)
```

b) Student wants to view the instructors' contact information

Create View View_Instructor_Contact AS (SELECT i.Name, i.Cell, i.Email FROM Instructor i INNER JOIN Courses c ON i.IID = c.IID INNER JOIN StudentCourses sc ON sc.CID = c.CID WHERE sc.SID = CURRENT_USER)

c) Exam Coordinator wants to view the students' information

Create View View_Students AS (SELECT s.FName, s.LName, s.Email, s.Address, s.Cell FROM Students s INNER JOIN StudentCourses sc ON sc.SID = s.SID)

d) Student wants to view the Results

Create View View_Results AS (SELECT r.sTitle, r.Marks, r.Grade FROM Results r INNER JOIN Courses c ON c.CID = r.CID INNER JOIN StudentCourses sc ON sc.CID = c.SID INNER JOIN Students s ON s.SID = sc.SID WHERE s.SID = CURRENT_USER)

e) Instructor wants to view the students' Results

Create View View_Students_Results AS (SELECT r.sTitle, r.Marks, r.Grade FROM Results r INNER JOIN Courses c ON c.CID = r.CID INNER JOIN Instructor i ON i.IID = c.IID WHERE i.IID = CURRENT_USER)

f) Parents view of the students' Results

Create View View_Results AS (SELECT r.sTitle, r.Marks, r.Grade FROM Results r INNER JOIN Courses c ON c.CID = r.CID INNER JOIN StudentsCourse sc ON sc.SID = c.CID INNER JOIN Students s ON s.SID = sc.SID WHERE s.PID = CURRENT_USER)

IV. Implementation of TRDBAC for Students Results Information System

The following policies need to be implemented using the time constraints:

If the Exam Coordinator is not available due to certain reason and a necessary change in the results is required then we may allow the instructor to update the results for a certain time period. For instance, the instructor is allowed to update the results from June 28, 2010 9:00 AM to June 30, 2010 5:00 PM. Following is the description of this policy in transaction Datalog (TD)

View.Upd_Temporal.sResults(CID, SID, RID, Marks, Grade, 1)
(Results.IsActive(SID, CID, RID, 1) ||Results.Exists(SID, CID, RID, 1)) && !DateExpired(IID, Current_DateTime)
Instructor(IID, Additional_Role, '6/28/2010 9:00:00 AM', '6/30/2010 5:00:00 PM', 1)
Student(SID, 1)
Courses(CID, IID, 1)
StudentCourses(SID, CID, 1)
Results(RID, CID, SID, 1)
Upd.sResults(SID, CID, RID, Marks, Grade, 1)

SQL Implementatio

Create View Upd_Temporal_Results AS (UPDATE RESULTS r, Instructor i SET r.sTitle='Title', r.Marks='Marks', r.Grade='Grade' WHERE CURRENT_DATETIME BETWEEN i.FromDate AND i.ToDate AND i.Additional_Role=1 AND i.IID = CURRENT_USER)

Another scenario where we need to implement the TRDBAC is to allow the students to view the results for some specific time period then the result will go offline. For instance, we say that the students are allowed to view the results from July 1st 2010 12 AM to July 5th 2010 5:00 PM, and then we have the following policy description:

View.view_Temporal.sResults(SID, RID, Marks, Grade, 1)
(Results.IsActive(SID, CID, RID, 1) || Results.Exists(SID, CID, RID, 1)) && !DateExpired(SID, CID, Current_DateTime)
Student(SID, Additional_Role, '7/1/2010 00:00:00 AM', '7/1/2010 5:00:00 PM')
Courses(CID, 1)
StudentCourses(SID, CID, 1)
Results(RID, CID, SID, 1)
View.sResults(SID, CID, RID, Marks, Grade, 1)

SQL Implementation

Create View View_Temporal_Results AS (SELECT r.sTitle, r.Marks, r.Grade FROM Results r INNER JOIN Courses c ON c.CID = r.CID INNER JOIN StudentCourses sc ON sc.CID = c.SID INNER JOIN Students s ON s.SID = sc.SID WHERE CURRENT_DATETIME BETWEEN s.FromDate AND s.ToDate AND s.Additional_Role=1 AND s.SID = CURRENT_USER)

It is clearly described in the above scenarios that when we need TRDBAC and how it will be implemented in the SQL.

V. Conclusion

RDBAC is strong enough to implement the access control policies at the database level instead of implementing at application level. In some scenarios we are not only interested to implement the policies at database level but also interested to have some time constraints for the subjects. For that purpose we introduced TRDBAC which is the extension of RDBAC and proved a good temporal model that allows us to have some time constraints in our policies.

References

[1] Elisa Bertino, Fellow, IEEE, and Ravi Sandhu, Fellow, IEEE, "Database Security—Concepts, Approaches, and Challenges", IEEE Transactions on Dependable and secure computing, vol. 2, NO. 1, January-March 2005

[2] Meg Coffin Murray, Kennesaw State University, Kennesaw, GA, USA, "Database Security: What Students Need to Know", Journal of Information Technology Education: Innovations in Practice Volume 9, 2010

[3] Min-A Jeong', Jung-Ja Kim, and Yonggwan Won, "A Flexible Database Security System using Multiple Access Control Policies"

[4] Elisa Bertino and Elena Ferrari, "Data Security"

[5] Lars E. Olson, Carl A. Gunter, Sarah Peterson Olson, M.D. "A Medical Database Case Study for Reflective Database Access Control"

[6] D. Ferraiolo and R. Kuhn. Role-based access controls. In Proceedings of the 15th NIST-NCSC National Computer Security Conference, October 1992

[7] L. E. Olson, C. A. Gunter, W. R. Cook, and M. Winslett. Implementing reflective access control in SQL. In DBSec'09, Montreal, QC, Jul. 2009.

[8] Bonner, A.J.: Transaction datalog: A compositional language for transaction programming. In: Cluet, S., Hull, R. (eds.) DBPL 1997. LNCS, vol. 1369, pp. 373–395. Springer, Heidelberg (1998)

[9] W. Rjaibi, P. Bird. A Multi-Purpose Implementation of Mandatory Access Control in Relational Database Management Systems. In Proc. of the 30th International Conference on Very Large Databases, Toronto, Canada, 2004

[10] E. Bell, L. J. LaPadula. Secure computer systems: Unified exposition and multics interpretation. Tech-nical Report MTR-2997, The Mitre Corporation,Burlington Road, Bedford, MA 01730, USA.

[11] Ferraiolo, D., Cugini, J., and Kuhn, R. 1995. Role-based access control: Features and motivations. In Proceedings of the Annual Computer Security Applications Conference, IEEE Press, Los Alamitos, Calif.

[12] Elisa Bertino , Piero Andrea Bonatti , Elena Ferrari, TRBAC: a temporal role-based access control model, Proceedings of the fifth ACM workshop on Role-based access control, p.21-30, July 26-28, 2000, Berlin, Germany

[13] A. J. Bonner. Transaction Datalog: A compositional language for transaction programming. Lecture Notes in Computer Science, 1369:373–395, 1998.