CS4440
Project Proposal – Group 2

Dustin Burke, Bryan Davidson, Arcadiy Kantor, Adam Leonard

‘

Motivations and Objectives

Often times when creating a database, there are certain datasets that need to be exclusive to either the database designer or to an administrator. Moreover, there is often a division between sets of data that a power user can access and modify and other sets that a regular user can work with. Finally, there can even be aspects of the database that guests can access. Taking this idea, you can extend it to other application such as software design. User types can include Developer, Designer, Engineer, Tester, and Requirements. Each of those users needs to see specific tables in a database; no more and no less. By implementing Role Based Access Control, you can handle what specific users are able to see and what they are not.

There are many ways to tackle this problem, but we think a useful way to go about it is to write a PHP library that will first examine a query that would be sent to the database to see if the user creating the query has access to the data he or she is requesting. If the user has sufficient privileges, the query will be passed to the database, but, if not, the query will be rejected. For this to be even more useful the PHP library we write will be largely database-agnostic; as long as the database implements the ODBC API, our library can be used.

Related Work

The field of Role Based Access Control (RBAC) has been extensively researched since its introduction in the early 1990s in the appropriately titled paper by David Ferraiolo and Richard Kuhn, “Role Based Access Control.” This paper built on the work done at the Department of Defense to develop two basic types of access control: Mandatory Access Control (MAC) and Discretionary Access Control (DAC). MAC was substantially more secure, but DAC was considered to be more appropriate for industry use. RBAC, then, built upon the basics of MAC but, as Ferraiolo and Kuhn noted in their original paper, adapted the controls to be “more central to the secure processing needs of non-military systems.”

This research has been expanded significantly since with a number of publications from a variety of researchers, culminating in the publishing of a book by Ferraiolo, Kuhn

and Ramaswamy Chandramouli in 2003. Most significantly, Kuhn, Ferraiolo and Cugini formally outlined the RBAC model and several variations thereof in the paper "Role Based Access Control: Features and Motivations" in 1995. This was the first major paper allowing for implementations of the access control technology to follow a certain set of guidelines and to allow researchers to concisely explain what subset of RBAC functionality they are implementing. The National Institute of Standards and Technology published a standard (again with the involvement of Kuhn and Ferraiolo) in 2000, and finally an ANSI publication recognized the method in 2004.

While the concept of RBAC has become more notable in recent years, the extent of its implementation and therefore use in many major systems remains inconsistent. According to the 1998 paper, "Role-Based Access Control Features in Commerical Database Management Systems" by Ramaswamy and Sandhu, although all three systems they evaluated had support for role hierarchies, only one of the three systems had support for the mutual exclusion of roles. Even today, MySQL, an open source relational database management system, has only marginal support for RBAC.

The concept has also been implemented in a number of software systems beyond the database level. Many systems, including the Windows administrative control, now use some form of role-based access management. In addition, basic access control list management has been extended to use role-based means of authentication in a number of frameworks for web development. For instance, the Zend Framework includes a component called Zend_Acl which provides for role-based "lightweight and flexible access control list functionality and privileges management." The Zend Framework is a set of modules for developing flexible PHP applications. However, this component focuses on permissions for actions a user might take in a PHP application, rather than database-specific constructs like tables.

We were unable to find PHP implementations of role-based permission controls that were fine-grained enough to work on a database table level rather than query actions. Furthermore, many implementations of RBAC in PHP are specific to a certain database system. This is the area we seek to address with our work.

Proposed Work

Our proposed project is to create a PHP library that enforces role based access

control policies in database agnostic environments. Current role based access control policies are built directly into the database engine and are only available in a limited number of database systems. Our new approach to role based access is beneficial because it will facilitate faster implementation and enforcement of role based policies than before through the combined use of a newly developed PHP library and any database that supports ODBC. Users will interface with their database through our library (as opposed to the current MySQL and ODBC functions found in PHP) in order to enforce role based access policies when querying, updating, and maintaining their database. Because we will use ODBC functions in our new library, our implementation will be database agnostic and have no problems running with any ODBC compliant database beneath it.

In order to evaluate the success of the project, one must look at a few criteria. First and foremost, the solution should correctly enforce role based access control policies in databases. We plan to demonstrate this functionality by building a sample web application that makes use of the new PHP library. Secondly, the library should work correctly with any SQL-based underlying database. This will be accomplished through the use of standardized ODBC wrappers throughout the new library to query and update the underlying database. To verify that the new library is indeed database agnostic, we will show that it works when using different underlying database systems, such as MySQL, Microsoft SQL Server, and Oracle.

A rough system overview needed to accomplish these goals simply includes the new PHP library and any database that is supported by ODBC. In order to enforce the role based access policies, users will interface with their database through the PHP library. When a user attempts to issue an action on the database, our library will first identify and verify the user's role(s). From there, we will determine what actions the user is able to perform on the database. This role and permission data will be stored in metadata tables directly in the user's database. If the user is allowed to perform the requested action, their SQL statement will be passed on to the database. Otherwise, the library will not allow the statement to execute and return an error to the user.

Plan of Action

CS4440
Project Proposal – Group 2

For our project, one of our members has a PHP webserver that will run our code once it is written. The PHP code can be written in any text editor like Notepad or JEdit. Once we have written our library, we plan to test it on an SQL Server, MySQL database, and Oracle database. MSNDAA will provide access to a SQL server, MySQL databases are open-source and free, and Prism has Oracle databases for every Georgia Tech student.

Weekly Schedule/Milestones:

October 15 – Get databases set up

October 22 – Have structure of user tables set up to query

October 29 – Decide on syntax for checking users privileges and begin writing code

November 5 – Continue to write library

November 12 – Finish writing library and begin testing

November 19 – Finish working out bugs and testing

Evaluation and Testing

As noted above, we will have access to three databases for testing purposes. If time permits at the end of the project, we may be able to perform further tests on any other available database(s). Our testing plan is to create an example application in PHP which uses our library for its database access methods. In this example, we will define various roles and then create several users, each with a different role. We will then attempt to perform several operations on the database through this application (e.g. creating a new table, deleting tuples, updating columns, etc.) with each user. Some of the users will have the access privileges to perform these actions and others will not. None of the users who do not have the correct privileges for an operation should be able to execute that operation.

Bibliography

Comparing Simple Role Based Access Control Models and Access Control Lists, J. Barkley, (1997), Second ACM Workshop on Role-Based Access Control.

Formal Specification for Role Based Access Control User/Role and Role/Role Relationship Management, S. Gavrila, J. Barkley, Third ACM Workshop on Role-Based Access Control.

An Introduction to Role Based Access Control NIST CSL Bulletin on RBAC (December, 1995).

Role Based Access Control: Features and Motivations, D.F. Ferraiolo, J. Cugini, D.R. Kuhn, Computer Security Applications Conference.

Role Based Access Control, D.F. Ferraiolo and D.R. Kuhn (1992), 15th National Computer Security Conference.

Role Based Access Control , D.F. Ferraiolo, D.R. Kuhn, R. Chandramouli, Artech House, 2003.

RBAC References – Zend Framework Wiki. <http://framework.zend.com/wiki/display/ZFUSER/RBAC+(Role+Based+Access+Control)+References>. September 2, 2007.

Role Based Access Control. <http://csrc.nist.gov/rbac/>. September 2, 2007.

Zend Framework Documentation: Zend_Acl. <http://framework.zend.com/manual/en/zend.acl.html>. September 2, 2007.