Visualizing Access Control Policies in Databases

Jared Chandler

Comp 116
Fall 2015

To the Community

I chose this topic because it is something I have an interest in improving. SQL databases are widespread and are responsible for storing some of the most important information used on a daily basis. The personal details of our lives reside in these systems. Financial, healthcare, educational and government organizations all rely on SQL databases as the central repository for facts about their operations and the people they serve. Naturally these systems are targets for attack. Despite having industry standard security mechanisms such as role based access control, SQL databases continue to be compromised due to lax security policies. As a community we must consider why and seek a solution.

Abstract

Role Based Access Control (RBAC) is a part of the SQL specification. All major relational database management systems (RDBMS) support RBAC. However even with these security mechanisms, SQL Injection attacks exploiting weak permissions and gaps in the access control policies remain pervasive and show no signs of going away. This paper examines obstacles to utilization of RBAC in SQL databases. Building on this examination we present a system designed to overcome these obstacles by presenting RBAC to the user in a visual context.

Introduction

Role Based Access Control (RBAC) is the industry standard for implementing security. Its defining characeristic is placing users into roles which they perform and then applying security policy to those roles [2]. This allows for central administration of security policy by promoting a single point of security truth for each role. It facilitates separation of duties between users by forcing security to be abstracted from a user into a role or group. Finally it encourages the principle of least privilege, namely that a user should only have the rights required for the roles they occupy, nothing more.

Given that such a powerful security model has been adopted and formally implemented with multiple versions of the SQL standard, why does database security continue to prove so difficult?

Related Work

Turning to Landwehr's taxonomy of computer program security flaws [4] shows that this problem is not easily classified. The industry standard and standards compliant database systems have a RBAC security mechanism. Is the user's failure to use it effectively a flaw we can attribute to the program? Landwehr

doesn't address this. Bertino [1] examines a variety of access control models and their relationship to database systems, among them RBAC. She focuses on the implications of different risks and mechanisms from the perspective of which type of access control system a database employs. The user is mentioned as a component only in passing. It is notable that while security architectures are explored from a system perspective, the human role is minimized.

The field of Human Computer Interaction (HCI) offers insight into the interactions between humans and security systems [5]. In "Security and human computer interfaces." Johnston [3] writes "The interface informs the user of the security functions that are available and how to use them."

Smith [9] adds to this that "many widespread security problems arguably might stem from bad interaction between humans and systems". The standard interface for most RDBMS is text input and output via a command prompt.

Sasse [8] goes a step further and calls for security designers to identify the causes of undesirable user behavior and address them to design effective security systems. A system which is secure from a theoretical perspective fails if it encourages bad actions by the user. Sasse focuses on password systems as an example, but many of the ideas can easily be extended. When a password complexity requirement becomes onerous, a user will choose something memorable at the expense of something strong. Sasse points to human memory as a limitation which drives poor security choices, not by intent or laziness rather by simple cognitive limitation.

Lax database security is not a result of poor or flawed systems, rather a poor security interface between the user and the system. An interface which does not convey information about security functions and the context in which they apply places a greater burden on the working memory of the user.

The intersection of the access control and visualization is very much an open area of exploration [6,10]. Ray's "Using UML to visualize role-based access control constraints." [7] is an example of using visualization to ease the burden of recognizing access control issues.

According to Wickens [11] the essential idea in the study of Human Computer interfaces is that an interface should replace memory with visual information and in doing so lower the burden to the user.

We suggest that the following are the key ways in which access control in database systems place burdens on the working memory of the user.

First, access control policy is dynamic. The effective result of the application of a policy is specific both to the state of the system at a point in time, and to the activity or query the user is executing.

Second, Database Manipulation Language is declarative. A user queries the database and awaits the results. Access control is a binary operation on this activity. If the query is permitted, it is executed. If it is not, the database notifies the user that they have insufficient permissions. There is no representation of access control in the query the user writes to retrieve data.

Third, RDBMS lifecycles often exceed those of applications built on top of them. A database and its access control policies may have a lifecycle approaching a decade or more. Consequently, the resulting architecture often is neither optimized for database security, nor application security. Legacy objects and access control policies can be persisted long after the applications which necessitated them have been retired.

Fourth, data-modeling may be done independent of security modeling. Security policies and role definitions may be developed only as an afterthought, not as an integral and ongoing part of the architecture of the database.

Fifth, and finally the database and its access control policies may predate the user, outlast the user, or both. The person making decisions about security often isn't the person who designed the database.

Problem Statement

These burdens on the working memory of the user can be condensed down into three problems a solution should address.

1. It can be difficult for a user to predict what the effect of an access control policy will be.
2. The result of a policy application can only be understood in context.
3. The action necessary to achieve a desired security outcome may not be obvious.

To address these problems, we turn to a combination of visualization and HCI.

Solution

SQL Grant Revoke Visualization (SuGaRVIZ) [12] is a prototype system for the visualization of SQL RDBMS access control policies within a particular activity context. It deals with three separate components of applying an access control policy to an activity. First, SuGaRVIZ allows the user to input objects as native SQL table DDL definitions in the Objects input. Second, SuGaRVIZ takes an access control policy as a

series of SQL grants and revokes in the Grants input. Third and finally, SuGaRVIZ takes an activity as a set of SQL Select DML queries in the Activities input.

The system renders these three components in a single visualization. Objects (tables) are rendered in the style of entity relational diagrams with relationship identifying links omitted. SQL grants are visually represented as one of four distinctly colored boxes on each column of a table. There is one box for each of the primary table actions: SELECT, INSERT, UPDATE, and DELETE. The user input grants are visualized in order. If an action is granted on an object, the corresponding box in the visualization is colored. If a previously set grant is revoked, the box is left unfilled. If there is no grant specified for an object it is similarly left unfilled. The activity input is parsed and a set of columns required for execution of the query or queries is computed. These columns are colored in the visualization with a color corresponding to the action of the query.

SuGaRVIZ solves three problems central to a user's ability to understand the access control policy within a specific RDBMS context.

Problem 1: Working Memory

By providing a visual representation of the three components along with the native SQL representation, SuGaRVIZ lowers the burden on the user's working memory. Objects can be manipulated in their native form and the change in visualization observed directly. Following the principles of effective interface design, color coding is used to indicate commonalities.

Problem 2: Context

By providing a way to observe the interaction between the activity query and the applicable access control policies, the user is given a representation of the activity within the security context it takes place. This allows them to understand whether their activity context and the security context are aligned. Objects which are common to the two contexts are easily seen, as are objects belonging to one or the other. In this fashion the visualization makes it easy for the user to see if something has been left out (insufficient permissions) or something included which ought not be (excessive permissions).

Problem 3: Actionable

By visualizing native SQL, the tool inherently produces an actionable output. Alterations to the grants and revokes can be applied with minimal effort to a SQL RDBMS. Similarly, so to can changes to activity queries and the underlying tables given as objects input.

Future Work

SuGarVIZ is well positioned for evaluation in an experimental setting. Assessing the quantitative impact of visualization in terms of real world security scenarios is one possibility.

Currently SuGaRVIZ displays all grants as belonging to a single role. Extending it to allow roles to be turned on and off with the results visually displayed is another possibility.

Other areas of security visualization such as firewall rulesets or file access permissions may benefit from visualization similarly to database system security.

Conclusion

The importance of the interface is understood within the HCI community. Its role is less clear in security research. The security of a system is only as strong as its weakest link. It many cases that link is the method of interaction between the user and the system. Johnston [3] says "The easier a system is to use, the less likely the user will be to make a mistake or to try to bypass the security feature." SuGaRVIZ proposes that the easier a system is to use, the more likely the user will implement good security policy.

References

[1] Bertino, Elisa, and Ravi Sandhu. "Database security-concepts, approaches, and challenges." Dependable and Secure Computing, IEEE Transactions on2.1 (2005): 2-19.

[2] Ferraiolo, D.F. and Kuhn, D.R. "Role-Based Access Control." 15th National Computer Security Conference (1992): 554–563.

[3] Johnston, J., Jan HP Eloff, and Les Labuschagne. "Security and human computer interfaces." Computers & Security 22.8 (2003): 675-684.

[4] Landwehr, Carl E., et al. "A taxonomy of computer program security flaws."ACM Computing Surveys (CSUR) 26.3 (1994): 211-254.

[5] Nielsen, Jakob. Usability engineering. Elsevier, 1994.

[6] Osborn, Sylvia, and Yuxia Guo. "Modeling users in role-based access control." ACM Workshop on Role-Based Access Control. 2000.

[7] Ray, Indrakshi, et al. "Using UML to visualize role-based access control constraints." Proceedings of the ninth ACM symposium on Access control models and technologies. ACM, 2004.

[8] Sasse, Martina Angela, Sacha Brostoff, and Dirk Weirich. "Transforming the 'weakest link'—a human/computer interaction approach to usable and effective security." BT technology journal 19.3 (2001): 122-131.

[9] Smith, Sean W. "Humans in the loop: Human-computer interaction and security." Security & Privacy, IEEE 1.3 (2003): 75-79.

[10] Tidswell, Jonathon E., and Trent Jaeger. "An access control model for simplifying constraint expression." Proceedings of the 7th ACM conference on Computer and communications security. ACM, 2000.

[11] Wickens, Christopher D., et al. "Introduction to human factors engineering." (1998).

[12] SuGaRVIZ, http://www.eecs.tufts.edu/~chandler/sgrviz 2015