Reasoner-Based Policy Assurance in Databases

José Hiram Soltren <jsoltren@mit.edu>

January 22, 2009

1 Introduction

The widespread use of automated systems to collect, store, and retrieve data in the public, private, and academic sectors has given rise to a large number of databases. Many of these databases contain private, personally identifiable, or sensitive information. In the financial sector, databases store information about bank accounts, trades and company activity that is not suitable for a large audience. In the health services industry, hospitals and insurance companies maintain databases containing confidential patient health data. In the military, databases contain classified and secret data that must remain hidden for reasons of national security. All of these systems share a common need of regulated access to data.

Database systems traditionally use some form of access control to enforce policies regarding the data they contain. These policies, a form of rule-based access control, create a control structure that mimics the structure of the database itself. We can restrict access to tables, enable certain views of data, or prohibit access selectively. These systems have two major points of failure. The first is that they require system designers to think about data and security simultaneously. The second is that they often fall short, creating a substantial "gray area" between unlimited access and highly restricted access.

In this project, we will apply Semantic Web technologies to a new area, that of policy assurance in database systems. We argue that the logical reasoners of the Semantic Web world are well suited for a more abstract form of rule-based access control. We will define a language that allows us to describe database queries, and their semantics, to both human and automated agents. We will construct a policy framework that enables us to create policies from abstract concepts, and use existing logical reasoners to determine whether or not we are in compliance of these technologies. Finally, we will integrate our system into an existing relational database implementation.

Our approach is novel in using Semantic Web technologies for this type of policy checking. The use of access controls in database systems is not new, nor is the use of reasoning systems and forward chaining to make logical deductions. We feel that Semantic Web technologies are well suited to the database access control problem, as they provide a robust, dynamic, and traceable way of implementing access control at a very fine granularity.

The rest of this proposal is structured as follows. First, we describe the problem of policy assurance in databases in more detail, offering sample scenarios. We then discuss the current state-of-the-art, including a discussion of existing Semantic Web technologies, and best practices in database administration using traditional models of access control. We then discuss a sample implementation, and demonstrate how the use of Semantic Web technologies is fundamentally different from prior approaches. We set a framework for future work and discuss challenges before concluding.

2 Problem Description

Our goal is to create a policy assurance system for a database containing sensitive data. Specifically, we wish to encode abstract policies in a reasoning language, and determine if access to our database is in adherence to these policies. Re-using as much of our existing framework as possible, we can inform users of how we derive at the conclusion of whether or not they are acting within the policies we established.

We would like for our system to work in databases with policies that restrict what the database administrator can do. The database administrator may be in charge of a database containing secured, limited-access, or confidential data. The administrator must be assured that users of the system are in compliance of the policies in place, even though the administrator may not have permission to see the queries, or the results of the queries, that the users make. Our system aimes to provide a database administrator with a tool that tells them if users are playing fair, or breaking the rules.

3 Previous Work

3.1 Semantic Web

> "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation."
>
> - Tim Berners-Lee, James Hendler, and Ora Lassila

The Semantic Web is an ongoing endeavor to make information accessible to both human agents, and automated systems. It is an extension of the World Wide Web, a collection of design principles that seek to foster information sharing in new ways through the use of new systems.

At present, Semantic Web technologies specify specific ways of encapsulating information so that it is human and machine readable. The primary method is RDF, an XML specification that binds objects to properties. The Notation 3 (N3) language is a form of "syntactic sugar" for RDF. Accountability in RDF, or AIR, offers a language for expressing policies in RDF languages. [4]

Applications such as the Tabulator browser addition render RDF information in an easily human readable form. The real power in Semantic Web technologies is the way that they easily lend themselves to rule-based systems and reasoners. One existing system, N3Logic, uses the N3 language to perform reasoning. [1] AIR also provided an infrastructure for logical reasoning.

3.2 Policy Awareness

Within the realm of the Semantic Web, there is prior work relating to policy awareness. This is the application of reasoner technologies to rule-based systems, where the rules represent laws, licenses, or policies that relate to the system.

The Information Accountability paper describes the arenas in which policy-aware systems are if interest. [10] Among these are privacy, copyright, surveillance, and data mining. The paper argues that if we can trace who uses data and how they use it, if we have a way of finding accountability, we can begin to move toward policies that are between full disclosure and full control. The paper offers scenarios of how information aware systems would be useful in everyday life.

REIN describes a system offering "policy management". [3] The system is policy-language agnostic, and operates in an independent environment, offering a yes/no opinion of policy compliance to an existing system. The REIN paper defines three possible operating modes. In the server-side rules operating mode, the analogy of choice is an application for a library card. The user does not know the rules, but supplies the information requested on the form to the library for processing. In the client-side rules mode, the onus of rule checking is on the user. The hybrid mode offers a way of sharing this responsibility. At present, REIN is implemented in Python, using the N3 policy language and the CWM reasoner.

There is prior work demonstrating the use of a policy-aware system for checking license information. [6] In this case, the system uses Creative Commons license policies to check for compliance with the content creator's sharing preferences. The author describes this system as a way of "keeping honest people honest." The system, as implemented, converts metadata on an image file to an AIR policy, and checks usage patterns against a set of rules that parallel the Creative Commons license policies. If, for example, an image from the Flickr Web site is used on someone's personal page, in violation of the content creator's wishes as expressed in their choice of Creative Commons licensing, this tool will detect a conflict.

Our logical approach is to use the AIR reasoning language to implement policies. We have yet to specify how we will encode policies, or queries.

3.3 Methodologies of Access Control

We are implementing a form of access control in this project. We thought it would be instructive to look at the existing literature, and the approaches

systems designers take in designing traditional access control systems. Systems designers tend to have a different mind set when approaching security problems than Web designers: systems designers tend to favor closed systems, whereas Web designers tend to favor open systems. Existing relational database systems extend on these technologies in their implementation of access control.

3.3.1 Yesterday: Mandatory and Discretionary Access Control

The traditional models of access control, circa the mid 1980s, consist of mandatory access control, or MAC, and discretionary access control, or DAC. The origins of MAC lie in secure military systems, as a way to enforce permissions on proprietary, secured, confidential data. In an MAC system, the only way to gain access to data is to expressly have an outside authority grant access. There is absolutely no provision for an end user to alter permissions on data. The primitives for enforcing access control extend deep into the design of the operating system. MAC systems explicitly define rings of access; an example of this is the levels of protection used in an x86 processor. Mandatory access control is primary a military security technology, and does not see widespread implementation. Some secure Unix operating systems, including SELinux, utilize MAC design principles.

Discretionary access control is a design principle that allows a user to show that they have the credentials to access data. Users have the capability of modifying their own permissions, or the permissions of others. A widespread example of a DAC system is that of Unix style permissions. On Unix systems, everything, from devices to network sockets to data on a disk, is represented as a file. Every file has three sets of permissions: those for the owning user, for the owning group, and for everyone else. Each of these sets may be granted (or denied) permissions to read, write, and/or execute a file. The operating system checks the user's credentials and the file's permissions before enabling an operation. Access Control Lists (ACLs) are another example of a DAC system. Such a security model works well for general purpose computing, but not in all scenarios.

The excessive rigors in the implementation of a MAC system, and the inadequacy of a DAC system, have led systems researchers to newer design philosophies that allow finer granularity in security settings.

3.3.2 Role Based Access Control

In the early 1990s, systems researchers began to find practical limitations in the expressive power of the discretionary access control model. Researchers presented role based access control as a more secure, easy to implement alternative to discretionary access control. [2] Role based access control is a form of mandatory access control, in that users obtain their permissions from an outside authority. The primary difference is in the structure of permissioning. In role based access control, a user can be assigned to one (or more, in some implementations) "roles". As an example, a user in a large company might wear several

hats throughout their time in the company, from "human resources associate" to "financial associate" to "systems analyst". Each of these roles requires a different set of permissions. The formal description of role based access control, demonstrated in the paper, shows the power and flexibility of assigning roles rather than individual permissions. It is particularly well suited to assigning permissions where a high user turnover rate is present.

3.3.3 Rule- and Policy-Based Access Control

Rule based access control is an extension and a generalization of role based access control. Instead of simple roles, we can use rules and logic to derive whether or not a particular agent should have access to a particular resource. As an example, consider a student trying to access a secured Web page. The student may have credentials saying, "I am a graduate student", "my advisor is Professor Smith", and "I am a member of Research Group X". The page may have a set of rules defining access: it may be restricted to individuals who are members of certain research groups, and also graduate students. Different versions of the page may exist based on the credentials used. REIN [3] and SWRL [5] provide sample implementations of policy based access control.

Our system will likely mimic one of these closely. We will draw from their examples in the way that they define and process rules.

3.4 Access Control in Relational Databases

Modern relational database management systems utilize some form of discretionary or role-based access control to regulate access to tables. The database administrator may restrict database access to particular rows or columns in particular tables, may grant or deny read and write permissions, and may limit the user to executing particular queries. In effect, all of these systems limit the user to seeing a particular subset of the data in the database. It follows that the structure of the access control policy must closely follow the structure of the data itself. The structure of the security policy may divulge secure information, and may not be tolerant of changes to the underlying structure of the database.

Our project seeks to address a particular shortcoming in database systems at present. In order to enforce access control policies, the database must be able to see the queries that a user is making against the database. This poses a problem when the queries themselves may reveal sensitive data.

3.5 Alteration of Data

An orthogonal approach to security of certain kinds of sensitive data is the alteration of the data itself. If we restrict the utility of the data through alteration, we can, in theory, limit the usefulness of the data, its sensitivity, and its potential for damage. The classic example of this approach is the use of a heavy black marker to manually censor sensitive government documents prior to

public release. There has been some research into formalizing and automating this process.

TODO: Talk more about Latanya Sweeney's k-anonymity [9] [8], and algorithms for anonymizing data on the fly, such as Datafly [7]. Talk about the MIMIC II database, and the challenges they face, based on the conversation with Mauricio. Our system could potentially anonymize data, though that does require access to the data.

4 Design Proposal

TODO: Talk about some implementation details. I need more information to make this concrete!

5 Challenges

Our solution does not address the issue of users who conspire together. It may be possible for two users to work as a team, obtaining independent sets of data from a protected database. The users can combine the data offline and break one of our policies.

We have yet to define a method of transforming a SQL query into an object that a reasoner can understand, and it would be impossible to do so without access to the query itself. Since we wish to implement this system as a drop-in addition to an existing system, it is particularly important to work within the existing database query infrastructure. Some languages, inclusing SPARQL and SPASQL, may be well suited to our goals.

6 Conclusion

Our approach to database security differs from previous approaches in that it uses rule-based Semantic Web technologies to offer policy assurance. Our system will integrate with relational database management systems in production, offering a yes/no opinion as to whether or nor a single or series of queries is in violation of a policy. This will allow fine-grained access control to users, and policy assurance for administrators, with no divulgence of information at any stage.

References

[1] Berners-Lee, T., Connolly, D., Kagal, L., Scharf, Y., and Hendler, J. N3logic: A logical framework for the world wide web. *Journal of Theory and Practice of Logic Programming* (2007).

[2] Ferraiolo, D. F., and Kuhn, D. R. Role-based access controls. In *15th National Computer Security Conference* (October 1992), pp. 554–563.

[3] Kagal, L., Berners-Lee, T., Connolly, D., and Weitzner, D. Using semantinc web technologies for policy management on the web. In *21st National Conference on Artificial Intelligence (AAAI)* (July 2006).

[4] Kagal, L., Hanson, C., and Weitzner, D. Using dependency tracking to provide explanations for policy management. *IEEE Policy 2008* (2008).

[5] Li, H., Zhang, X., Wu, H., and Qu, Y. Design and application of rule based access control policies. In *ISWC Workshop on Semantic Web and Policy* (2005), pp. 34–41.

[6] Seneviratne, O., Kagal, L., Weitzner, D., Abelson, H., Berners-Lee, T., and Shadbolt, N. Detecting creative commons license violations on images on the world wide web. In *WWW2009* (April 2009).

[7] Sweeney, L. Guaranteeing anonymity when sharing medical data: the datafly system. In *AMIA Annual Fall Symposium* (1997).

[8] Sweeney, L. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems 10 (5)* (2002), 571–588.

[9] Sweeney, L. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness, and Knowledge-Based Systems 10 (5)* (2002), 555–570.

[10] Weitzner, D. J., Abelson, H., Berners-Lee, T., Feigenbaum, J., Hendler, J., and Sussman, G. J. Information accountability. *Communications of the ACM* (June 2008).