Trust Management in Databases

Scott D. Stoller
Department of Computer Science
Stony Brook University

Cross-References

Access control, SQL access control model, Trust management

Definition

Trust management in databases refers to access control models that support the characteristic features of trust management—notably, decentralization of security policies and information used by the policies—and are seamlessly integrated with a database management system (DBMS).

Background

Traditional database access control models, such as the SQL access control model, are *static*—the assignment of privileges to users and roles is defined manually by administrators, and does not depend on or vary with the contents of the database or other information sources—and *centralized*—the access control policy for a database is stored in the same DBMS as the database and does not depend on any external information sources.

Such models are not well suited to distributed computing systems. Access control models better suited to such systems are *attribute-based*—the assignment of privileges is based on attributes of (and relations between) users and resources and varies automatically when this information changes—and *decentralized*—the policy and information used by the policy may come from multiple sources. To support this, all information is labeled with its source, and policies specify which sources are trusted for which kinds of information. Access control models with these characteristics are often called *trust management* models.

Trust management policy languages are fundamentally *relational*: they define an *authorization* relation, which relates users with their privileges, in terms of relations describing the attributes of users and resources. Policies may also define and use auxiliary relations; this makes policies more modular and easier to read. Most trust management policy languages define these relations using *rules*, similar to rules in logic programming languages.

Theory

A general-purpose trust management system can, in principle, be used to control access to any resource, including a database, but trust management systems designed specifically for databases offer greater efficiency, security, and ease of use, by re-using existing functionality in the database. A key observation is that the most widely-used databases are based on a relational language, namely the Structured Query Language (SQL). Thus, the relations defined and used in trust management policies can be represented by tables in the database, and the policy language can be based on SQL, instead of rules. di Vimercati *et al.* (2007) proposed the first trust management system with this design.

Several syntactically small but semantically powerful extensions to SQL are needed to realize this approach. One extension allows specification of trusted sources for the information in each table. Specifically, the definition of a database table T may include an optional clause that specifies a table or view S that contains a record for each trusted source for T; concretely, a designated

column in S contains the source's name (e.g., public key or X.509 distinguished name). Only information from those sources may be inserted in T. For example, using syntax similar to that proposed by Stoller (2009), the statement `create certtable Patient (name varchar(30), id varchar(9)) check (issuer in (select subject from Physician))` defines a table named `Patient` with columns `name` and `id` and whose trusted sources are principals named in the `Physician` table. Specifically, data in an X.509 attribute certificate C can be inserted in `Patient` only if (1) C's issuer is named in the `subject` column of some record in `Physician` and (2) C contains an attribute corresponding to (i.e., with the same name as) each column of `Patient`.

A second extension connects attribute information stored in tables and views with access privileges and role memberships. A new form of the SQL grant statement specifies a table or view T and a privilege P (e.g., permission to update a specified table). Each user for which T contains a record is granted privilege P. This invariant is maintained whenever the contents of T changes. A similar variant of the grant statement specifies a role R instead of a privilege P: each user for which T contains a record is granted membership in R. For example, using syntax similar to that proposed by Stoller (2009), the attribute-based grant statement `ab_grant delete on Patient to (select subject from Physician)` grants the privilege to delete records from the `Patient` table to principals whose name appears in the `subject` column of the `Physician` table.

di Vimercati *et al.* (2012) designed a trust management system for databases that features an efficient algorithm for verifying whether the attribute information in given certificates should be trusted, based on a set of supporting certificates and the database's trust management policy.

Advanced trust management features, such as credential discovery and trust negotiation, also fit naturally in this framework. For example, suppose a user tries to insert a certificate C into a table T, but C's issuer I does not appear in the table or view S of trusted sources for T. The system might automatically ask trusted sources for S to send certificates about I, which could then be inserted in S, allowing C to be inserted in T. This is an example of *credential discovery*. Sources for S can create such certificates from information stored in tables. However, a source will do this only if the requested information is releasable to the requester according to the source's *trust negotiation* policy, which might specify that the information is releasable only to requesters with certain attributes. Another small extension to the syntax of database table definitions allows specification of trust negotiation policies.

Although database systems do not currently support trust management, it seems likely that they will support it in the future, because of the importance of trust management in large-scale systems, because current database security models can be extended seamlessly to support trust management, and because these extensions require only localized changes to the database implementation.

Recommended Reading

S. D. C. di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati, Trust management services in relational databases, *Proceedings of the 2007 ACM Symposium on InformAtion, Computer and Communications Security (ASIACCS)*, ACM, 2007, pp. 149-160.

S. D. Stoller, Trust management and trust negotiation in an extension of SQL, *Proceedings of the 4th International Symposium on Trustworthy Global Computing (TGC 2008)*, volume 5474 of Lecture Notes in Computer Science, Springer-Verlag, 2009, pp. 186-200.

S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Psaila, and P. Samarati, Integrating trust management and access control in data-intensive Web applications, *ACM Transactions on the Web* 6(2), pp. 6:1–6:43, May 2012.