Trust Management in Databases

Scott D. Stoller
Department of Computer Science,
Stony Brook University, Stony Brook, NY, USA

Related Concepts

►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.

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 locally in that 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 the 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 reusing 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. [1] proposed the first trust management system with this design.

Several syntactically small but semantically powerful extensions to SQL are needed. 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 [2], 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` 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 the 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 [2], 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 some record in the `Physician` table.

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 site will do this only if the requested information is releasable to the requester according to the site'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 is needed to specify 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

1. De Capitani di Vimercati S, Jajodia S, Paraboschi S, Samarati P (2007) Trust management services in relational databases. In: Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security (ASIACCS), ACM, New York, pp 149–160
2. Stoller SD (2009) Trust management and trust negotiation in an extension of SQL. In: Proceedings of the 4th International Symposium on Trustworthy Global Computing (TGC 2008). Lecture notes in computer science, vol 5474. Springer-Verlag, Berlin, pp 186–200

Trusted Boot

William D. Casper, Stephen M. Papa
High Assurance Computing and Networking Labs (HACNet), Department of Computer Science and Engineering, Bobby B. Lyle School of Engineering, Southern Methodist University, Houston, TX, USA

Synonyms

IC integrated circuit

Related Concepts

►Levels of Trust; ►Trusted Computing

Definition

Trust: (a) assured reliance on the character, ability, strength, or truth of someone or something (b) one in which confidence is placed [1].

Trusted Boot refers to the ability to have confidence or trust in the security of a system startup, beginning with the initial configuration boot of the system.

Background

Computer security is reliant on trust. This trust is composed of several fundamental principles, including confidence that the targeted system is configured as expected, will operate as intended, and has not already been compromised or exploited. A verification strategy with appropriate methodology should be used to validate this trust. The validation can be done with a combination of hardware attestation and software integrity verification.

Validation of hardware and software components is performed by validation hardware and software primitives. At the lowest level, these primitives are controlled by the boot process of the system and are ideally the first items to execute after power-on of a system, but how are these hardware and software primitives validated? In order to trust the hardware and software primitives they themselves must be trusted. If an attacker can undermine the trust of the trusted primitives then the attacker can own the system and control the results of the system validation, thus circumventing the validation process. This initial trust of the system primitives can be ensured through the use of a trusted boot process.

Theory

A trusted boot process is rooted in the ability of the system to securely ensure the integrity of the boot primitives. These boot primitives must not be susceptible to compromise, either by modification or by substitution. In other words, these boot primitives must be exactly what was implemented and cannot have unknown and undetected changes made to them. Once undetected changes are made, the boot primitives lose their trustworthiness.

A trusted boot is usually a process of multiple smaller validated boot iterations. These iterations may be comprised of hardware or software verification stages, depending on the implementation mechanisms utilized. Some mechanisms utilize both hardware and software features during the same verification iteration to cross-check the trustworthiness of both features. Iterations build on the trustworthiness of the previous iterations and thus are allowed to inherit the trust values of the preceding steps in the boot process. This process relies on the ability of a particular iteration to be validated prior to acceptance