International Journal of Research in Science & Technology http://www.ijrst.com/
(IJRST) 2011, Vol. No. 1, Issue No. I, Apr-June **ISSN: 2249-0604**

Security Issues in Relational Databases Management System

ROHIT KUMAR
Working as Asstt. Prof. Computer Science at IIPM, Ghaziabad

INTRODUCTION

Security models, developed for databases, differ in many aspects because they focus on different features of the database security problem or because they make different assumptions about what constitutes a secure database. This leads to disjointed and incomplete understanding of the organizational security strategy. This makes it difficult to reconcile different security requirements Access control mechanisms of current relational database management systems are based on discretionary policies governing the accesses of a subject to data based on the subject's identity and authorization rules. Common administration policies include centralized administration, by which only some privileged subjects may grant and revoke authorizations, and ownership administration. Ownership-based administration is often provided with features for administration delegation, allowing the owner of a data object to assign other subjects the right to grant and revoke authorizations. More sophisticated administration mechanisms can be devised such as joint administration, by which several subjects are jointly responsible for authorization administration [1]. A number of extensions have been proposed with the goal of enriching the expressive power of the authorization languages in order to address a large variety of application requirements. A first extension deals with negative authorizations [2]. In many cases, authorizations given to users must be given access authorizations to data only for the time periods in which they are expected to need the data. A context-based access control model is able to incorporate into access control decision functions a large variety of context-dependent information, such as time and location. Content-based access control is an important requirement that any access control mechanism. Relational DBMSs support content-based access control based on the use of views. Fine-grained mechanisms supports access control at the tuple level. Role based access control has been motivated by the need to simplify authorization administration and to directly represent access control policies of organizations. As database technology moves into new applications areas, the requirements changes and become more demanding. Object model provides a superset of the functionalities of relational database management system. Many of the commercial relational database management systems are incorporating object-oriented concepts to facilitate database design and development in the increasingly object-oriented world [3][4]. The incorporation of object-oriented concepts offers new tools for securing the data stored in the object databases. Security in relational databases is achieved relatively with ease as compared to object databases since it is based on some formal mathematical model [5][6].

RELATED WORK

Existing database security models, defined for relational DBMSs, are not suitable for an object-based database system because of the wide differences in data models. Unlike traditional RDBMSs, secure OODBMSs have certain characteristics that make them unique. The object-oriented database model permits the classification of an object's sensitivity through the use of class (of entities) and instance. When an instance of a class is created, the object can automatically inherit the level of sensitivity of the superclass. Although the ability to pass classifications through inheritance is possible in objectoriented data bases, class instances are usually classified at a higher level within the object's class hierarchy. This prevents a flow control problem, where information passes from higher to lower classification levels [7]. OODBMSs also use unique characteristics that allow these models to control the access to the data in the database. They incorporate features such as flexible data structure, inheritance, and late binding. Access control models for OODBMSs must be consistent with such features. Users can define methods, some of which are open for other users as public methods. Moreover, the OODBMS may encapsulate a series of basic access commands into a method and make it public for users, while keeping basic commands themselves away from users. Various security models have been designed specifically for object databases. These security models make use of the concepts of encapsulation, inheritance, information hiding, methods, and the ability to model realworld entities that are present in OO environments. These models differ in many aspects because they incorporate different security features, or because they focus on different security issues, or because they make different assumptions about what a secure database should be or about the object-oriented model itself

Security Issues of RDBMS and ODBMS

Early research efforts focused on defining a proper security policy [8] in the area of database security requirements. Earlier research gives different features of database security policy ncluding user identification/authorization policy, access control policy, inference policy, accountability policy, audit policy and consistency policy. Some important principles were introduced in the security policy development to design a good database security policy; minimum vs. maximum principle, open vs. closed system principle, centralized vs. decentralized administration principle, granularity principle and access privilege principle. To enforce multiple access control policies within a single unified system [9] presented a flexible authorization manager. It contains three components: the data system that includes file system, relational databases and object oriented databases; the user hierarchy and the authorization specification language. The emergence of object-oriented database management system extended the early approach to access control for relational database systems. They incorporate features such as flexible data structure, encapsulation, composite objects, versions, inheritance, information hiding and late binding. Various security models have been designed specifically for object databases. These security models make use of the concepts of encapsulation, inheritance, information hiding, methods, and the ability to model realworld entities that are present in OO environments. These models differ in many aspects because they incorporate different security features, or because they focus on different security issues, or because they make different assumptions about what a secure database should be or about the object-oriented model itself. Security policies and security models are implemented by security mechanisms, which can be either discretionary or mandatory.

Some Security Problems In Object-Oriented Database Systems

A. Polyinstantiation

This problem arises when users with different security levels attempt to use the same information. The variety of clearances and sensitivities in a secure data base system result in conflicts between the objects that can be accessed and modified by the users. Through the use of polyinstantiation, information is located in more than one location, usually with different security levels. Obviously, the more sensitive information is omitted from the instances with lower security levels. Although polyinstantiation solves the multiparty update conflict problem, it raises a potentially greater problem in the form of ensuring the integrity of the data within the data base. Without some method of simultaneously updating all occurrences of the data in the data base, the integrity of the information quickly disappears. In essence, the system becomes a collection of several distinct data base systems, each with its own data. In an object-oriented world, the different views could relate to different object values, different class structures, different class methods and different method definitions. I will describe each type of polyinstantiation below:

Object Value polyinstantiation: an unclassified user views a specific object different from a Secret subject views.

Class Structure polyinstantiation: an Unclassified subject views a specific class consisting of the instance variables different from a Secret subject views in terms of instance variables.

Class method polyinstantiation: an unclassified user views EMP as having the methods get-name, change-name while the Secret subject views EMP as having methods get-name, change-name, get-salary, change-salary.

Method polyinstantiation: an unclassified user views a method update-salary to have one parameter which is the amount by which the salary should be increased. A Secret user views this method as having two parameters; one is the amount and the other is the new salary value which is returned to the user.

A polyinstantiated view-name may arise whenever an unclassified (or conditional) user requests to use the same name for defining a view which is already used by a secret user. [26] suggested that a view name associated with the security-level of the user is used to name the view.

B. Aggregation

The aggregation problem occurs when a user can from aggregates of related items, all of which are classified at some level, that deduce classified data [26]. The higher level information (which may be thought to be subject to a higher level of security clearance) may be inferred from a large number of lower level data items. A collection of information items may be required to be classified at a higher security level than any of the individual items that comprise it. The aggregation problem occurred when a subject's right to individual pieces of information results in knowledge to which it does not have a right. The aggregation problem prevents the subject from gaining access to information of higher sensitivity by aggregating lower sensitivity data. This is usually addressed by restricting combinations of accesses in certain ways.

C. Inference problem

The word ―inference‖ means ―forming a conclusion from premises‖ . Users of any database can draw inferences from the information they have obtained from the database and prior additional information (called supplementary knowledge) they have. The inference can lead to information disclosure if the user is able to access to information they are not authorized to read. This is the inference problem in the database security [26]. Inference problem occurs when a user can deduce (or infer) information from a collection of individual accesses against a database [8] summarized different approaches to handle the inference problem: (1) place restrictions on the set of allowable queries that can be issued by a user; (2) add noise to the data; and (3) augment a database with a logic-based inference engine to modify queries before the database processes them. Security violations via inference occur when users pose multiple queries and acquire unauthorized information. A solution to handling the inference problem in relational systems is to augment a relational DBMS with a logicbased inference engine and a knowledge base. The inference engine will detect security violations via inference when processing queries. Two approaches to implementing such an inference controller are as follows: In the first approach, the databases as well as the security constraints are expressed in a logic programming language with support for representing and manipulating objects. An example of such a language is object-prolog. In the second approach, an object-oriented database system is augmented with an inference engine and a rule base. The inference engine is based on an extension to first order logic. The queries are modified first by the inference engine before the object-oriented DBMS processes them. The techniques proposed in this second approach can be used to augment a SORION-based object-oriented database system with a logic-based inference engine which will detect security violations.

Motivation

Users have been demanding information ―anytime, anywhere‖ . The diversity in the range of information accessible to a user at any time is growing rapidly. Accessing diverse and autonomous information repositories with different APIs (Application Program Interfaces) is not accepted since the user has to be retrained to ―learn a new API and its usage on accessing different information sources‖ . As a result, a need to integrate diverse information from different sources and provide a user with common APIs has become a necessity. Adding further complexity, the user mobility and privacy concerns prevent gaining knowledge about exact user location, which otherwise could aid in reducing the access latency to the required information repository. Thus, APIs need to:

• Locate the requested information transparently and intelligently based on the user's security level,

• Access, process, and integrate information repository/repositories intelligently and efficiently based on the user's access right,

• Locate the current location of the user efficiently without violating user's privacy,

• Reliably transport resultant information to the user efficiently and securely, and

• Display the information to the user according to his/her display category.

It should be reemphasized that the information requested might not be confined to a single information repository and distributed across numerous repositories at different geographic locations. Hence, in some cases, additional phases of decomposition of requests and integration of intermediate results would be required.

To gain further insight regarding the global information processing, APIs, and limitations of infrastructure several related issues and their possible solutions are studied. The information repositories could include legacy systems, database systems, data warehouses, information services, (i.e. stock quotes, news, airline information, weather information, etc), and the almost limitless information available on the Internet and the World Wide Web. The APIs include traditional systems to access data in the form of relational, object-oriented, distributed, federated, or multidatabase management systems. The expanding technology is making available a wide breadth of devices to use these APIs. Potential devices include desktop computers, laptops, PDAs (Personal Digital Assistants), and cellular phones. The access mechanisms vary according to the access devices and the environments in which data is being accessed. For example, data can be accessed from a desktop workstation connected to a LAN (Local Area Network), or from a laptop computer via a modem, or from a PDA via a wireless communication medium.

Access devices have different memory, storage, network, power, and display requirements [76]. Within the scope of traditional multidatabases, as a solution to the integration of information from a collection of autonomous and heterogeneous sources, researchers have effectively addressed issues such as local autonomy, heterogeneity, transaction management, concurrency control, transparency, and query resolution in a ―sometimes, somewhere‖ environment. Consequently, the aforementioned issues and solutions are defined based on fixed clients and servers connected over a reliable network infrastructure. A mobile data access system (MDAS) relies on the information stored in multidatabases. A MDAS is distinguished from a multidatabase by the communication links between nodes. Clients (and possibly even servers) are no longer fixed – they are mobile and can access data over wireless links. Client mobility may apply in both idle (not actively accessing the system) and active modes. The concept of mobility, however, introduces additional complexities and restrictions in multidatabase systems. A user accessing data through a remote connection with a portable device has a high probability of facing problems such as reduced capacity network connections, frequent disconnections, and processing and resource restrictions. Research on providing solutions in a wireless medium is underway. The trade offs are being identified gradually and prototypes accordingly are being developed, tested, and commercialized. Some current systems have the means to provide timely and reliable access to remote1 data in wired and wireless media. However, this alone is not sufficient to ensure content users. Users desire secure communication sessions. Confidentiality, integrity, and authenticity are required to meet user's security expectations. Within the scope of the global information sharing process, the security aspect has received less attention. Trust is a major factor that needs to be established when creating a secure environment that allows remote data access to users. Violating trust would result in havoc in the system. Providing security in distributed systems is, however, a difficult task. The wireless medium further imposes lower bandwidth, frequent disconnections, higher error rates, and non-secure links in the face of autonomy and heterogeneity. Site autonomy is a measure of enforcing local security by the local administrator. Whether security is enforced globally and/or locally (due to local autonomy), a secure global information system must address issues such as access control, authorizations and counter-measures for inferential security, and accountability. Users should be given access to information based on their role or access rights in the global system. Pervasive computing systems, which are just beginning to appear, extend the scope of MDAS. The security issues in both multidatabases and MDAS still apply, but are further complicated by user privacy concerns and resource-poor devices.

The remainder of this chapter is organized in the following manner. Section 2 looks at security issues in centralized database systems. This discussion provides the reader with the necessary background information on security methods, which are then discussed within the context of multidatabases and Mobile Data Access Systems (MDAS).

Security Issues of Centralized Database Systems

Three interrelated technologies are used to achieve information confidentiality and integrity in traditional DBMSs: authentication, access control, and audit [1]. Figure 1 logically illustrates how these three services interact with each other to ensure the security of the DBMS. Authentication identifies one party to another and it can be performed in both directions. It ensures the true identity of a user, computer, or process. Once the identity is established, the party can access data under the guidance of the access control service. Access control regulations are set by the system security administrator and define who can access what data with which privileges. However, authentication and access control do not comprise a complete security solution — they must be complemented by an auditing service. An auditing service records important user activities in system logs for real-time or a posteriori analysis. Real-time auditing is often referred to as intrusion detection. An audit service protects a system in three ways: detecting actual security violations; assisting the security administrator in discovering attempted attacks by recognizing abnormal activity patterns; and detecting possible system security flaws.

Characteristics of Relational Database Model

Insulation between Programs and Data

In traditional file processing, the structure of data files is embedded in the access programs, so any changes to the structure of a file may require changing all programs that access this file. By contrast, RDBM access programs are written independently of any specific file. The structure of data files is stored in the Database Management System (DBMS) catalog separately from the access programs. This property is known as program-data independence.

Self-describing Nature of Database System

This kind of database system is not only contains the database itself, but also a complete definition of the database. In conventional file processing, data definition is typically part of the application programs themselves. Hence, these programs are constrained to work with only one specific database, whose structure is declared in the application programs. For example, a PASCAL program may have record structures declared in it; a C++ program may have "struct" or "class" declarations. However, RDBM can access diverse databases by extracting the database definitions from the catalog.

Support of Multiple Views of the Data

A database typically may have many users, each of whom may require a different perspective of the database. The RDBM can contain virtual data that is derived from the database files based on the users requirement, but the file is not explicitly stored. Thus, multiple views of the same data is possible.

Controlling Redundancy

In traditional software development utilizing file processing, every user group maintains its own files for handling its data-processing application. Much of the data is stored twice, once in the files of each user group. This redundancy in storing the same data multiple times lead to duplication of effort, wasting the storage space and files that represent the same data may become inconsistent. In the RDBM, the views of different user groups are integrated during database design. For consistency, the database design stores each logical data item in only one place in the database

References

[1] E. Bertino and E. Ferrari, ―Administration Policies in a Multipolicy Authorization System,‖ Proc. 10th Ann. IFIP Working Conf.

Database Security, Aug. 1997.

[2] E. Bertino, S. Jajodia, and P. Samarati, ―An Extended Authorization Model,‖ IEEE Trans. Knowledge and Data Eng.vol. 9, no. 1, pp. 85-101, 1997.

[3] Mansour Zand, Val Collins, Dale Caviness, ―A Survey of Current Object-Oriented Databases,‖ ACM SIGMIS Database, Volume 26 Issue 1, February 1995.

[4] Elisa Bertino, ―Data Hiding and Security in Object-Oriented Databases,‖ In proceedings Eighth International Conference on Data Engineering, 338-347, February 1992.

[5] Martin S. Olivier, Sebastian H. Von Solms, ―A Taxonomy for Object-Oriented Secure Databases,‖ ACM Transactions on Database Systems, Vol. 19, No. 1, Pages 3-46, March 1994.

[6] Fausto Rabitti, Elisa Bertino, Won Kim, Darrell Woelk, ‖ A Model of Authorization for Next-Generation Database Systems,‖ ACM Transactions on Database Systems (TODS), Volume 16 Issue 1, March 1991.

[7] Pierangela Samarati, Elisa Bertino, Alessandro Ciampichetti, Sushil Jajodia, ―Information Flow Control in Object-Oriented Systems,‖ IEEE Transactions on Knowledge and Data Engineering, vol.9, no.4, pp.524–538, July-August 1997.

[8] Ahmad Baraani-Dastjerdi, Josef Pieprzyk, Reihaneh Safavi-Naini, ―Security In Databases: A Survey Study,‖ Department of Computer Science, The University of Wollongong, Wollongong, Australia, February 7, 1996.

[9] Sushil Jajodia, Boris Kogan, ―Integrating an Object-Oriented Data Model with Multi-Level Security,‖ Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, 7-9, May 1990.

[10] D. Elliott Bell, Leonard J. La Padula, ―Secure Computer System -Unified Exposition and Multics Interpretation,‖ Report, No. MTR- 2997, MITRE, 1976.

[11] E.B. Fernandez, R.C. Summers, and C. Wood, ―Database Security and Integrity,‖ Addison-Wesley, February 1981.

[12] Elisa Bertino, Ravi S. Sandhu, ―Database Security - Concepts, Approaches, and Challenges,‖ IEEE Transactions on Dependable and Secure Computing, Volume 2, Issue 1, Page(s):2 –19, March 2005.

[13] James M. Slack, Elizabeth A. Unger, ―A Model of Integrity for Object-Oriented Database Systems,‖ Proceedings of the 1992 ACM/SIGAPP Symposium on Applied computing: technological challenges of the 1990's, April 1992.

[14] R. Fagin, ―On an Authorization Mechanism,‖ ACM Trans. Database Systems, vol.3, no.3, pp.310-319, 1978.

[15] A. Eisenberg and J. Melton, ―SQL:1999, Formerly Known as SQL3,‖ SIGMOD Record, 1999.

[16] Elisa Bertino, Pierangela Samarati, Sushil Jajodia, ―High Assurance Discretionary Access Control for Object Bases,‖ Proceedings of the 1st ACM conference on Computer and communications security, December 1993.

[17] C. Wood and E.B. Fernandez, ―Decentralized Authorization in a Database System,‖ Proc. Conf. Very Large Databases, 1979.

[18] E. Bertino, P. Bonatti, and E. Ferrari, ―TRBAC: A Temporal Role-Based Access Control,‖ ACM Trans. Information and System Security, vol. 4, no. 3, pp. 191-233, 2001.

[19] Elisa Bertino, C. Bettini, Pierangela Samarati, ―A Discretionary Access Control Model with Temporal Authorizations,‖ in Proc. Of IEEE Int. Workshop on New Security Paradigms, Little Compton, Rhode Island, 1994.

[20] E. Bertino and L.M. Haas, ―Views and Security in Distributed Database Management Systems,‖ Proc. Int'l Conf. Extending Database Technology, Mar. 1988.

[21] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, ―Extending Query Rewriting Techniques for Fine-Grained Access Control,‖ Proc. ACM Sigmod Conf., June 2004.

[22] Joel Richardson, Peter Schwarz, Luis-Felipe Cabrera, ―CACL: Efficient Fine-Grained Protection for Objects,‖ ACM SIGPLAN Notices, conference proceedings on Object-oriented

programming systems, languages, and applications, Volume 27 Issue 10, October 1992.

[23] US Dept. of Defense, Trusted Computer System Evaluation Criteria, DOD 5200. 28-STD, Dept. of Defense, Washington, D.C., 1975.